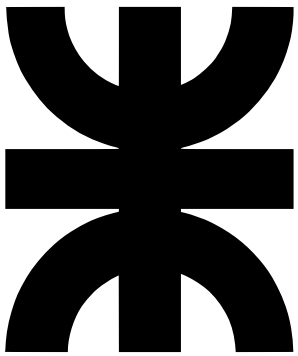
Plataforma de Hardware Reconfigurable



Luis Alberto Guanuco

Departamento de Ingeniería Electrónica

Universidad Tecnológica Nacional – Facultad Regional Córdoba

Este trabajo otorga el grado de

Ingeniero Electrónico

Cátedra Proyecto Final 2011

A toda mi familia y seres queridos que me apoyaron en todo momento.

Agreadecimientos

Se agradece principalmente a la comunidad de *Hardware y Software Libre* ya que sin el aporte cooperativo y desinteresado de ellos sería imposible acceder a un sin fin de herramientas informáticas que son de gran importancia en el desarrollo territorial de cualquier comunidad. También a los estudiantes y docentes investigadores del *Centro Universitario de Desarrollo en Automoción y Robótica (CUDAR)*, quienes con su basta experiencia han brindando soporte técnico en el desarrollo del proyecto. Agradezco el apoyo recibido por el *Departamento de Ingeniería Electrónica* de la *Universidad Tecnológica Nacional – Facultad Regional Córdoba.* Como así también a el *Laboratorio de Técnicas Digitales e Informática (LTDI)* por participar del proyecto, ofreciendo sus recursos humanos y físicos para la recepción del proyecto. A la *Agencia para el Desarrollo Económico de la ciudad de Córdoba (ADEC)* quienes nos han permitido participar de su programa *Córdoba Innovadora - Desarrollo Territorial en el Área Metropolitana de Córdoba.* En donde se ha podido adquirir experiencia en el manejo de proyecto y sobre todo la posibilidad de solventar económicamente todo el desarrollo para su transferencia al LTDI. Y por último agradecer a todas las personas que en el transcurso del desarrollo han participado institucionalmente como así también en forma desinteresada.

Resumen

El proyecto denominado *Plataforma de Hardware Reconfigurable*, ofrecer los recursos de *hardware* y *software* necesarios en el diseño de sistemas digitales reconfigurables de alta complejidad. El objetivo principal del proyecto es su implementación en el ámbito educativo, pero su aplicación puede extenderse a la industria ya que posee flexibilidad y una gran variedad de recursos (periféricos).
El desarrollo se encuentra publicado bajo condiciones legales que permite el *uso* y *modificación* de todo el proyecto por cualquier diseñador que así lo desee, lo que asegura la total *libertad* a quienes se estén formando en el diseño de sistemas electrónicos digitales. La *Plataforma de Hardware Reconfigurable* ofrece un desarrollo alternativo a las plataformas digitales comerciales ya que se encuentre adaptada a las necesidades requeridas por el ámbito educativo regional. En el desarrollo han participado activamente docentes e investigadores, quienes aportaron información que en parte han definido el perfil técnico y académico del proyecto.

Contenidos

Lista de Figuras

Lista de Tablas

Nomenclatura

Acrónimos / Abreviaturas

ASIC *Application-Specific Integrated Circuit*

BGA *Ball Grid Array*

CLB *Configurable Logic Block*

CPLD *Complex Programmable Logic Device*

DLL *Delay-Locked Loop*

DSP *Digital Signal Processor*

EEPROM *Electrically Erasable Programmable Read-Only Memory*

FF flip-flop, circuito que tiene dos estados estables y puede ser usado para almacenar información

FPGA *Field Programmable Gate Array*

GNU *GNU's Not Unix*

HDL *Hardware Description Language*

I2C *Inter-Integrated Circuit or IIC*

IEEE *Institute of Electrical and Electronics Engineers*

INTI Instituto Nacional de Tecnología Industrial

JTAG *Joint Test Action Group*

MSI *Medium Scale Integration*

PALCE *PAL CMOS Electrically erasable/programmable*

PAL *Programmable Array Logic*

PCI *Peripheral Component Interconnect*

PHR Plataforma de Hardware Reconfigurable

PLD *Programmable Logic Device*

PLL *Phase-Locked Loop*

SPI *Serial Peripheral Interface*

SRAM *Static-RAM*

SSI *Small Scale Integration*

UML *Unified Modeling Language*

USB *Universal Serial Bus*

Capítulo 1

Introducción

1.1 Motivación y objetivos

En el proceso de aprendizaje de las *Técnicas Digitales* se tiene un eslabón importante que es la implementación de estos sistemas a la práctica. Teorías como el *Álgebra de Boole* con operaciones digitales simples, hasta la síntesis de *microcontroladores* son prácticas comunes en la formación del profesional en el área de la Ingeniería Electrónica y resulta fundamental su ejercitación para concluir el ciclo de enseñanza. Actualmente existen nuevas herramientas al alcance de la mano que favorecen a la formación del estudiante de ingeniería, sobre todo las denominadas *Plataformas o Kit Educativos* que cuentan con una enorme complejidad y recursos en su diseño pero no así para el usuario final. Además es importante aclarar que, en décadas pasadas, las industrias e instituciones académicas de países desarrollados eran quienes contaba con las tecnologías más modernas.

En el avance tecnológico exponencial que se dio en las últimas décadas del siglo XX se podrían destacar varios logros pero quizá el que toma gran relevancia es el acceso a la información (*Internet*). Esta herramienta permite que regiones en desarrollo puedan llevar adelante estudios de nuevas tecnologías, comparando lo dicho con años anteriores donde no se contaba con la masificación de la información. En la mayoría de los casos las instituciones académicas resultan llevar la bandera en estas búsquedas del conocimiento. En búsqueda de lograr nuestro objetivo principal, ofrecer una herramienta personalizado a las necesidades de los estudiantes que se *inician* en el área del diseño de sistemas digitales basados en Dispositivos Lógicos Programables (*Programable Logic Devices*), se ha trabajado tomando como referencias proyectos universitarios en nuestra Facultad como también de otros países además de los perfiles

de diseño que tienen las placas comerciales.

1.2 Transferencia del proyecto

Como se dijo en la sección anterior, el objetivo del proyecto es diseñar una plataforma que pueda ser útil para los estudiantes iniciales en el diseño de sistemas digitales. En el análisis de la implementación del proyecto en el ámbito académico se optó en coordinar un trabajo conjunto con el *Departamento de Ingeniería Electrónica* y el *Laboratorio de Técnicas Digitales e Informática (LTDI)*, ambas instituciones pertenecientes a la *Universidad Tecnológica Nacional – Facultad Regional Córdoba*. El LTDI es el principal laboratorio informático con el que cuentan los estudiantes de Ingeniería Electrónica. Este laboratorio dispone de recursos físicos necesarios para cubrir la demanda de las distintas Cátedras que allí se dictan, es decir, cuenta con computadoras, placas evaluadoras para diseño digital, instrumentación, *software's*, etc. Miembros del LTDI han formado parte de la generación de documentación necesaria para el presente proyecto y la intensión a futuro es que dicho personal sea quienes definan mejoras/modificaciones a versiones futuras de la *Plataforma de Hardware Reconfigurable*.

1.3 *Hardware* y *Software*

Las herramientas de *hardware* y *software* son comunes en el campo laboral de un Ingeniero Electrónico. Si bien el profesional Electrónico puede ejercer su actividad en diferentes ámbitos industriales, siempre requerirá el conocimientos de diseño de sistemas físicos como también interactuar con programas informáticos. Seguramente el desenvolvimiento del Ingeniero Electrónico de décadas anteriores contaba con otras herramientas y lograba desarrollarse con éxito en su profesión pues así lo requería la Industria. Hoy la situación es distinta y se podría citar diversos materiales bibliográficos que detallen con mayor claridad esta observación. No es nuestra intensión comparar el perfil del profesional de unas décadas atrás con el profesional de nuestro tiempo, simplemente nos interesa poner en contexto la importancia del vínculo entre el mundo físico y los sistemas informáticos.

El nombre del proyecto, *Plataforma de Hardware Reconfigurable*, hace referencia al diseño de placas electrónicas que presentan la posibilidad de reconfigurar su estructura interna y así sintetizar diferentes arquitecturas diseñadas por el usuario, aquí es donde toma importancia en definirla *Hardware Reconfigurable*. Pero a medida que se avance

en la lectura del presente informe se podrá notar la importancia que representa el *software* en el proyecto.

Actualmente en el mercado de las plataformas educativas de sistemas embebidos existe verdaderamente una enorme variedad de excelentes productos. Cada uno de estos desarrollos se encuentran orientados a un determinado grupo de usuarios pero la mayoría de estos diseños tienen recursos de *hardware* en común:

- Dispositivo principal de proceso (*hardware*)
- Puerto de programación y depuración (*debugging*)
- Periféricos
- Herramientas de *software*

A la hora de determinar que plataforma se pretende adquirir se debe realizar un análisis de los requerimientos de la implementación y tener una referencia del presupuesto con el que se dispone. La mayoría de las plataformas comerciales son adquiridas con la finalidad de realizar *prototipos* que permitan clarificar y definir un desarrollo final funcional. Persiguiendo el mismo fin pero en el ámbito académico, el perfil de la plataforma debe ofrecer los recursos físicos estratégicamente necesarios para el avance tecnológico de la región. De esta forma permitirá que los profesionales formados puedan implementar nuevas tecnologías en la industria local.

1.4 PHR (Plataforma de Hardware Reconfigurable)

La plataforma PHR se presenta como una herramienta para las prácticas en las Cátedras del área de Técnicas Digitales. Su estructura está basada en la caracterización planteada en la Sección 1.3. Es decir, la PHR presenta básicamente un Dispositivo Lógico Programable (PLD, siglas en inglés) al cual se tendrá acceso para sintetizar arquitecturas digitales implementadas mediante el uso de Lenguajes Descriptivos de Hardware (HDL, siglas en inglés). Anexo a este dispositivo central se dispone de una memoria de programación donde se almacenará la arquitectura implementada. Estos se acceden mediante un puerto de programación estándar denominado JTAG. Además se ofrecen distintos periféricos que permiten la comunicación con el exterior de la placa mediante: puerto serial (RS-232), pulsadores, llaves, indicadores LEDs y display de 7 segmentos de 4 dígitos y entradas/salidas de propósitos generales. Toda la energía

necesaria es proporcionada por una fuente de alimentación capaz de ofrecer tanto los niveles de potencia necesarios para el dispositivo central y sus periféricos, como así también controlar los requerimientos de encendido del dispositivo central (cuestión que se verá en detalle en Capítulos posteriores). La principal herramienta de *software* que complementa a la plataforma PHR es la encargada de reconfigurar el dispositivo central mediante el puerto JTAG.

1.5 Licencia Libre

El proyecto se encuentra distribuido en con licencias libres, para ser más exactos *GNU General Public License Version 3 (GPLv3)* y *Creative Commons Attribution 4.0 International*. Esto permitirá que cualquier persona pueda acceder a la documentación como así también los esquemáticos y códigos (*scripts*) que se han desarrollado. La mayoría de las referencias utilizadas en el desarrollo de la plataforma PHR han sido de otros proyectos que cuentan con licencias similares. La decisión de especificar la licencia del desarrollo no solo es por una cuestión legal, sino también para darle *valor* y *divulgación* a estos diseños que tienen objetivos similares a lo que aquí se pretende lograr con la PHR. De esta manera el proyecto formará parte de otros diseños similares que se encuentran distribuidos en la comunidad electrónica en forma libre o abiertas. Si bien hay desarrollos similares en otras Universidades del mundo, se pretende incentivar a que los estudiantes de esta región no solo tengan acceso a esta tecnología, sino también que se atrevan a realizar modificaciones y aportes a la plataforma PHR.

Capítulo 2

Conceptos Teóricos

2.1 Dispositivos Lógicos Programables

Los *Dispositivos Lógicos Programables* (PLDs) fueron introducidos a medidos de 1970s. La idea era construir circuitos lógicos combinacionales que fueran *programables*. Contrariamente a los microprocesadores, los cuales pueden *correr* un programa sobre un hardware *fijo*, la programabilidad de los PLDs hace referencia a niveles de *hardware*. En otras palabras, un PLD es un chip de *propósitos generales* cuyo *hardware* puede ser reconfigurado dependiendo de especificaciones particulares del programador[1].

EL primer PLD se llamaba PAL (*Programmable Array Logic*). Estos dispositivos disponían solo de compuertas lógicas (no tenían flip-flop), por lo que solo permitía la implementación de circuitos *combinacionales*. Para salvar este problema, Los *registered* PLDs fueron lanzados pocos después, los cuales incluían un flip-flop por cada salida del circuito. Con esta versión de los PAL, se podría implementar funciones *secuenciales* simples.

En el comienzo de 1980s, se agregaba más circuitos lógicos adicionales a la salida de los PLD. Este circuito de salida se lo identificaba como celda, llamado también *Microcelda*, que contenía (además de flip-flop) compuertas lógicas y multiplexores. Por otra parte, la celda era reprogramable, permitiendo varios modos de operación. Además, se podía proveer una señal de retorno (*feedback*) desde la salida del circuito a la lógico principal de la PAL, lo que le daba mayor flexibilidad a estos dispositivos reprogramables. Esta nueva estructura era llamada *generic PAL* o GAL. Una arquitectura de dispositivo similar fue conocido como PALCE (*PAL CMOS Electrically erasable/programmable*).

Todos estos chips (PAL, *registered* PLD, y GAL/PALCE) son ahora categorizados

como SPLDs (*Simple* PLDs). Los dispositivos GAL/PALCE son los únicos fabricados aún en una encapsulado independiente.

Luego, varios dispositivos GAL fueron fabricados en un solo chip, usando un esquema de direccionamiento más sofisticado, mayor tecnología en su fabricación, y varias características adicionales (como soporte JTAG e interfaces para varios estándares lógicos). Esta nueva propuesta se la conoció como CPLD (*Complex* PLD). Los CPLDs son actualmente muy populares debido a su alta densidad, funcionalidad, y bajo costo.

Finalmente, a mediados de 1980s, las FPGA (*Field Programmable Gate Array*) fueron introducidos al mercado de los IC. Las FPGAs diferían de los CPLDs en su arquitectura. tecnología, recursos internos, y costo. Estos dispositivos tenían como principal objetivo su implementación en diseños de gran requerimientos en recursos de hardware como así también un alto rendimiento.

Un pequeño resumen de los diferentes dispositivos PLDs se puede observar en la Tabla 2.1.

PLDs	Sample PLD (SPLD)	PAL Registered PAL GAL
	Complex PLD (CPLD)	
	FPGA	

Tabla 2.1 Evolución de los PLDs.

Por último, todos los PLDs (*simple* o *complex*) son no volátiles. Estos puede ser OTP (*One-Time Programmable*), en la que pequeños fusibles electrónicos son usados para la reprogramación, de igual forma que las EEPROM o memorias Flash. Las FPGAs, por otra lado, son en su mayoría volátiles. Para estas últimas se deben usar dispositivos externos para cargar las conexiones.

2.2 SPLDs

Como se mencionó anteriormente, los dispositivos PAL, PLA y GAL se clasifican como los *Simple* PLD (SPLDs). Una descripción de las arquitecturas de cada uno de estos dispositivos se presenta a continuación.

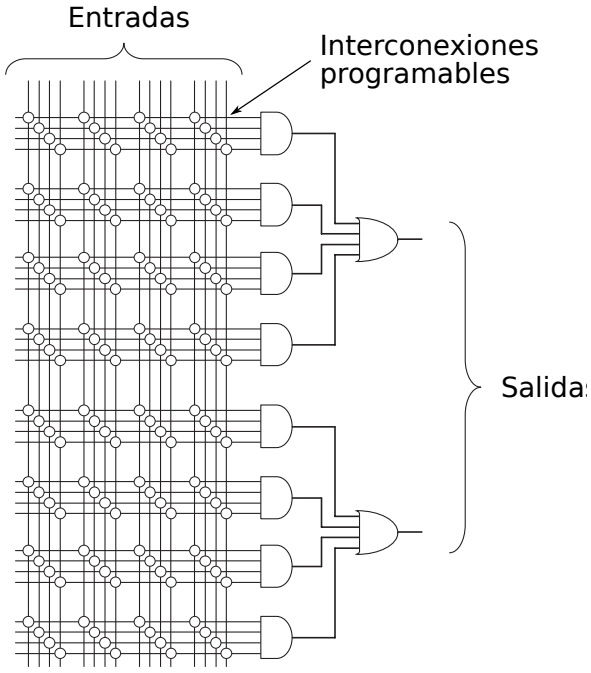


Figura 2.1 Arquitectura básica de una PAL.

2.2.1 PALs

Los *Programmable Array Logic* (PAL) son introducidos por Monolithic Memories Inc. a mediados de 1970. Su arquitectura básica se ilustra en la Figura 2.1, donde se representa con un pequeño círculo las conexiones programables. Como puede verse, el circuito esta compuesto de un arreglo de compuertas AND *programables*, seguido por un arreglo *fijo* de compuertas OR.

La implementación de la Figura 2.1 se basa en que cualquier función combinacional puede ser representada como una Suma de Productos (SOP); es decir, si a_1, a_2, $\cdots$, a_N son las entradas lógicas, entonces cualquier salida combinacional x puede ser compuesta como

$$x = m_1 + m_2 + \cdots + m_M \ , \tag{2.1}$$

donde $m_i = f_1\left(a_1, a_2, \cdots, a_N\right)$ son los términos mínimos de la función x. Por ejemplo

$$x = a_1\bar{a}_2 + a_2a_3\bar{a}_4 + \bar{a}_1\bar{a}_2a_3a_4\bar{a}_5 \tag{2.2}$$

Por lo tanto, el producto (términos mínimos) puede ser obtenido por medio de las compuertas AND, cuya salidas están conectadas a una compuerta OR para calcular su suma.

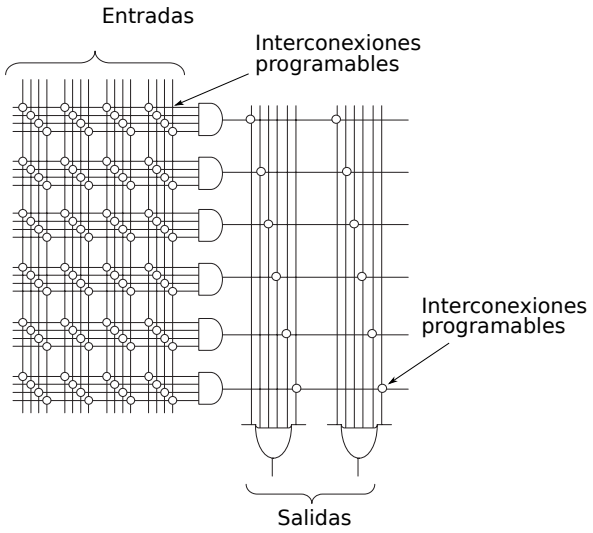


Figura 2.2 Arquitectura básica de una PLA.

La principal limitación de esta arquitectura es el hecho de que solo permite la implementación de funciones combinacionales solamente. Para solucionar este problema, las *registered* PALs fueron lanzadas a fines de la década de 1970s. Estas incluían un flip-flop en cada salida (luego de la compuerta OR en la Figura 2.1), de esta manera permitió la implementación de funciones secuenciales (aunque muy simples).

La primeras tecnologías empleadas en la fabricación de los dispositivos PALs fue bipolar, con una tensión de alimentación de 5V y un consumo de corriente al rededor de 200mA. La máxima frecuencia rondaba los 100Mhz, y las celdas programables eran de PROM (*fuse links*) o EPROM (con un tiemp de borrado de 20min. UV).

2.2.2 PLAs

Los PLA (*Programmable Logic Array*) fueron introducidos a mediados de 1970s (por Signetics Inc.). La arquitectura básica de un PLA se ilustra simbólicamente en la Figura 2.2. Si comparamos esta arquitectura con la Figura 2.1, se observa que la única diferencia fundamental entre estos es que mientras una PAL tiene compuertas AND programables y otras compuertas OR fijas, en el caso de las PLA *ambas* (las compuertas AND y OR) son programables. De esta manera se logra una ventaja en la flexibilidad del diseño. Sin embargo, se presentan elevados tiempos de retardos en los nodos de conexión internos que reducen la velocidad de funcionamiento del circuito.

La tecnología que se empleó en la fabricación de las PLAs fue la misma que en el caso de las PALs. Aunque las PLAs se encuentran obsoletas actualmente, estos han

reaparecido como parte de las arquitecturas de las primeras familias de los CPLDs de baja potencia, como por ejemplo la familia de los *CoolRunner* (de Xilinx Inc.).

2.2.3 GALs

La arquitectura de las GAL (*Generic* PAL) fueron introducidas por Lattice Inc. en los comienzos de 1980s. Este contenía varias mejoras sobre los primeros dispositivos PALs:

1. Se construyeron sealidas más sofisticadas de las celdas (*Macrocell*), las que incluían, además de flip-flop, varias compuertas y multiplexores.

2. Las Macrocell eran programables, permitiendo varios modos de operación.

3. Una señal de "retorno" desde la salida a la Marocell al arreglo reprogramable se incluyó, confiriendo al circuito mayor versatilidad.

4. Se utilizaron EEPROM en lugar de la PROM o EPROM.

Como se mencionó, la GAL es el único SPLD que todavía es fabricado en un encapsulado estándar. Además, éste también sirvió como parte en la construcción de la mayoría de los CPLDs.

La Figura 2.3 muestra un ejemplo de un dispositivo GAL, el GAL16V8. Este circuito cuenta con 16 entradas y 8 salidas, en un *package* de 20 pines. En cada salida hay una Macrocell (luego de la compuerta OR), que contiene, además del flip-flop, compuertas lógicas y multiplexores. Las interconexiones programables son representadas por pequeños círculos. Una señal de realimentación desde la Macrocell al arreglo programable puede también ser observado. Notar que esta arquitectura se asemeja directamente a la de la PAL (Figura 2.1), excepto por la presencia de una macrocell en cada salida y la señal de realimentación.

Actualmente los dispositivos GALs usan tecnología CMOS, alimentados a 3.3V, tecnología EEPROM o Flash, y alcanzan frecuencias máximas que rondan los 250Mhz. Varias compañías fabrican estos dispositivos (Lattice, Atmel, Texas Instruments, etc.).

2.3 CPLDs

La estructura fundamental en la arquitectura de los CPLDs se ilustra en la Figura 2.4. Como se puede ver, este consiste en varios PLDs (en general del tipo GAL) con

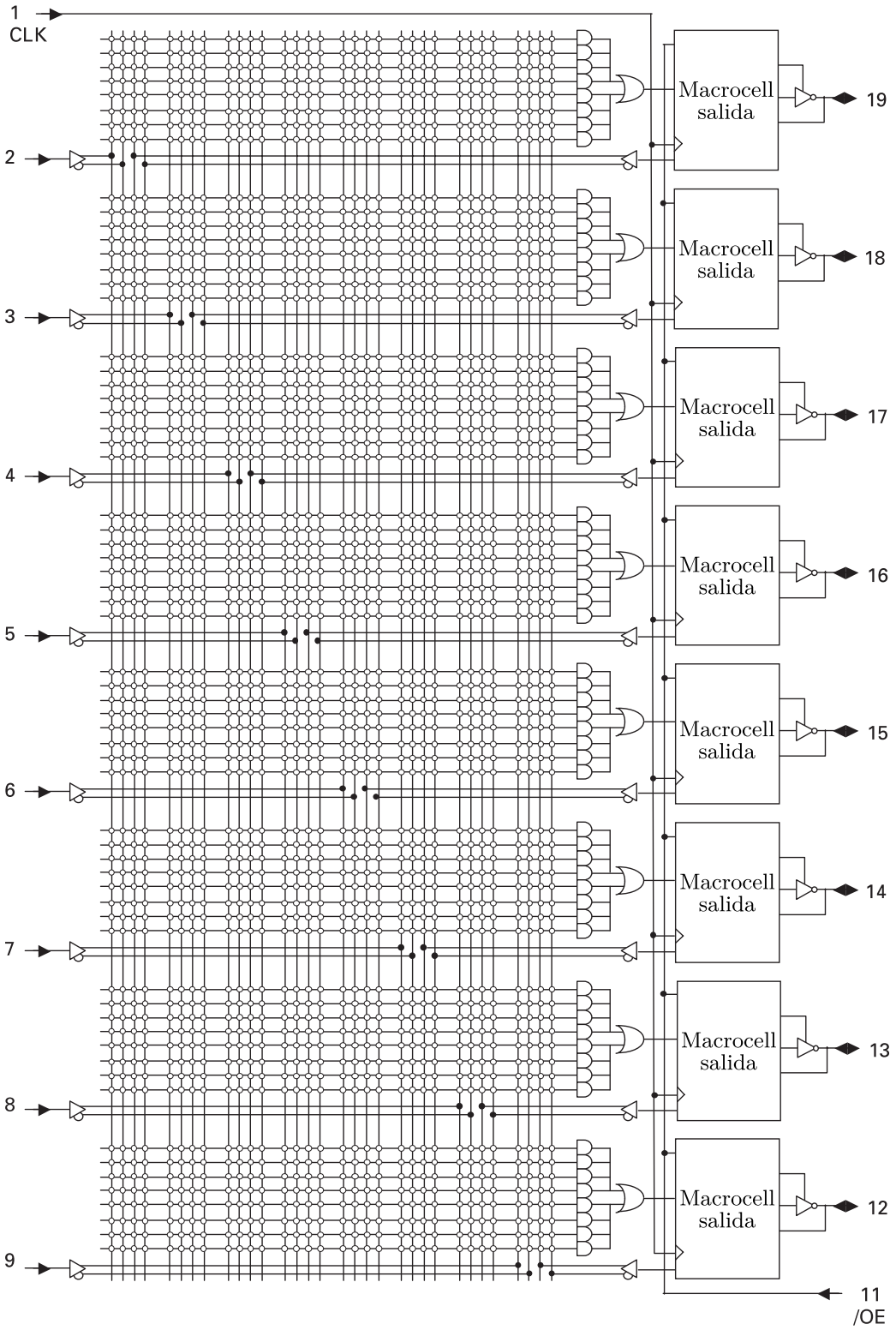


Figura 2.3 Dispositivo GAL 16V8.

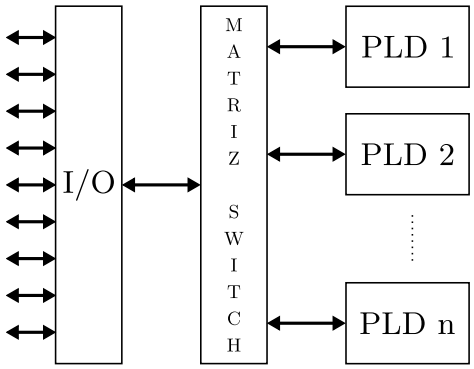


Figura 2.4 Arquitectura básica de un CPLD.

Familia	**XC9500 (XVm, XL)**	**CoolRunner XPLA3**	**CoolRunner II**
Macrocell	36 – 288	32 – 512	32 – 512
System gates	800 – 6,400	750 – 12,000	750 – 12,000
Pines I/O	34 –192	36 – 260	33 – 270
Frec. máxima interna	222 Mhz	213 Mhz	350 Mhz
Building Block	GAL 54V18 (XV, XL) GAL 36V18 (–)	Bloques PLA	Bloques PLA
Voltaje	2.5 V (XV), 3.3 V (XL), 5 V	3.3 V	1.8 V
Interconexiones	Flash	EEPROM	
Tecnología	0.35 μ CMOS	0.35 μ CMOS	0.18 μ CMOS
Corriente estática	11 – 500 mA	$<$ 0.1 mA	22 μA – 1 mA

Tabla 2.2 Características de los CPLDs de Xilinx.

una matriz de *switches* programables usadas para conectarlos todos juntos a al bloque de entrada y salida. Además, los CPLDs contiene normalmente otras características, como soporte JTAG e interfaz a otros estándares lógicos (1.8V, 2.5V, 5V, etc.).

Son varias las compañías que fabrican CPLDs, entre las más reconocidas tenemos Xilinx, Altera, Lattice, Atmel, Cypress, etc. En las Tablas 2.2 y 2.3 se disponen de las características de dos CPLDs, Xilinx y Altera. Como puede verse, más ade 500 Macrocells y más de 10000 compuertas pueden encontrarse en estos dispositivos.

Familia	**MAX7000 (B, AE, S)**	**MAX3000 (A)**	**MAX II (G)**
Macrocell / LUTs	32 – 512 macrocells	32 – 512 macrocells	192 – 1,700 macrocells 240 – 2,210 LUTs
System gates	600 – 10,000	600 – 10,000	
Pines I/O	32 –512	34 – 208	80 – 272
Frec. máxima interna	303 Mhz	227 Mhz	304 Mhz
Voltaje	2.5 V (B), 3.3 V (AE), 5 V (S)	3.3 V	1.8 V (G), 2.5 V, 3.3 V
Interconexiones	EEPROM	EEPROM	Flash + SRAM
Tecnología	0.22 μ CMOS EEPROM 4 capas de metal (7000 B)	0.3 μ 4 capas de metal	0.18 μ 6 capas de metal
Corriente estática	9 – 450 mA	9 – 150 mA	2 –50 mA

Tabla 2.3 Características de los CPLDs de Altera.

2.4 FPGAs

Las FPGAs fueron introducidas al mercado por la empresa Xilinx Inc. a mediados de 1980s. Estos dispositivos se diferencian de los CPLDs en su arquitectura, tecnología de almacenamiento, funcionalidades integradas, y costo, y además están orientadas a la implementación de altos rendimientos y grandes tamaños en lo que se refiere a recursos de hardware.

La arquitectura básica de una FPGA se ilustra en la Figura 2.5. Esta consiste de una matriz de *CLBs* (*Configurable Logic Blocks*), interconectados por un arreglo de matrices de conmutadores (*Switch Matrix*). Para caracterizar con más detalle estos dispositivo se debe recurrir a la información de los fabricantes, donde además se puede disponer de un interfaz JTAG a diversos niveles lógicos, otra funcionalidad como memorias SRAM, multiplicadores de clock (PLL o DLL), interfaz PCI, etc. Algunos chips también incluyen bloques dedicados como multiplicadores, DPSs, y microprocesadores.

Las FPGAs puede ser muy sofisticadas. La fabricación de chips con una tecnología CMOS de 90 nm., con nueve capas ed cobre y mas de 1000 pines de I/O, se encuentran actualmente disponible en el mercado. Algunos ejemplos de los empaquetados (*package*) de las FPGAs son ilustrados en la Figura 2.6, en los cuales se puede apreciar

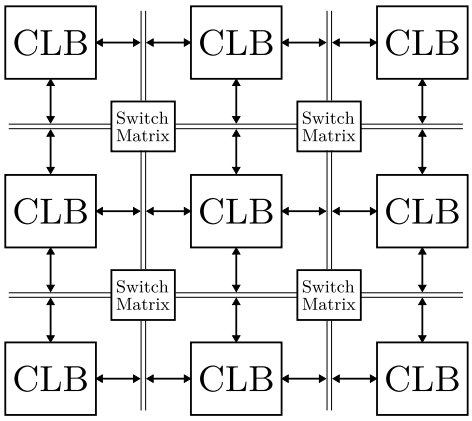
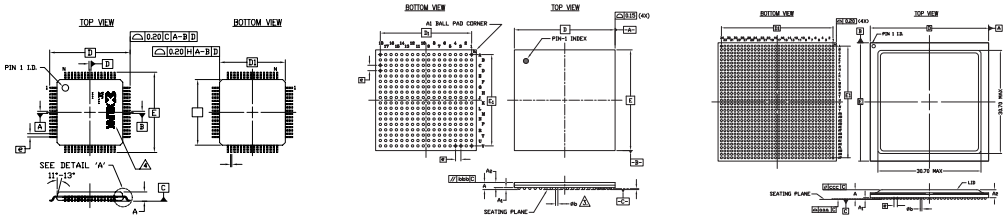


Figura 2.5 Arquitectura básica de una FPGA.



(a) *Package* VQ100 (b) *Package* CSG324 (c) *Package* FFG1156

Figura 2.6 Diferentes *package* de las FPGAs comerciales.

uno de los *package* más pequeños (Fig. 2.6a) con 100 pines, un *package* de tamaño mediano (Fig. 2.6b) de 324 pines, y uno de los grandes *package* con 1156 pines (Fig. 2.6c.

Varias compañías fabrican FPGAs, como Xilinx., Actel, Altera, QuickLogic, Atmel, etc. Ejemplo de dos fabricantes (Xilinx y Actel) se disponen en las Tablas 2.4 y 2.5. Como puede verse, estos dispositivos pueden contener miles de flip-flops y varios millones de compuertas lógicas.

Nótese que todas las FPGAs de Xilinx usan SRAM para almacenar las interconexiones, por lo que son reprogramables, pero volátiles (es así que requieren de una ROM externa). en cambio, las FPGAs de Actel son no-volátiles (estos usan fusibles electrónicos), pero no son reprogramables (excepto una familia, la cual usa memoria *Flash*). Ya que cada enfoque tiene sus propias ventajas y desventajas, la aplicación real dictará cual arquitectura de chip es la apropiada.

Familia	**Virtex II Pro**	**Virtex II**	**Virtex E**	**Virtex**	**Spartan 3**	**Spartan IIE**	**Spartan II**
CLBs	352 – 11.024	64 – 11.648	384 – 16.224	384 – 6.144	192 – 8.320	384 – 3.456	96 – 1.176
Celdas Lógicas	3.168 – 125.136	576 – 104.882	1.728 – 73.008	1.728 – 27.648	1.728 – 74.880	1.728 – 15.552	432 – 5.292
System gates		40k – 8M	72k – 4M	58k – 1.1M	50k – 5M	23k – 600k	15k – 200k
Pines de I/O	204 – 1200	88 – 1108	176 – 804	180 – 512	124 – 784	182 – 514	86 – 284
Flip-flops	2.816 – 88.192	512 – 93.184	1.392 – 64.896	1.392 – 24.576	1.536 – 66.560	1.536 – 13.824	384 – 4.704
Frec. máxima interna	547 MHz	420 MHz	240 MHz	200 MHz	326 MHz	200 MHz	200 MHz
Voltaje	1.5 V	1.5 V	1.8 V	2.5 V	1.2 V	1.8 V	2.5 V
Inter-conexiones	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM	SRAM
Tecnología	0.13 μm 9 capas de cobre CMOS	.15 μm 8 capas de metal CMOS	0.18 μm 6 capas de metal CMOS	0.22 μm 5 capas de metal CMOS	0.09 μm 8 capas de metal CMOS		
SRAM bits (Bloques de RAM)	216k – 8M	72k – 3M	64k – 832k	32k – 128k	72k – 1.8M	32k – 288k	16k – 56k

Tabla 2.4 Características de FPGAs fabricadas por Xilinx.

Familia	**Accelerator**	**ProASIC**	**MX**	**SX**	**eX**
Módulos lógicos	2.016 – 32.256	5.376 – 56.320	295 – 2.438	768 – 6.036	192 – 768
System gates	125k – 2M	75k – 1M	3k – 54k	12k – 108k	3k – 12k
Pines de I/O	168 – 684	204 – 712	57 – 202	130 – 360	84 – 132
Flip-flops	1.344 – 21.504	5.376 – 26.880	147 – 1.822	512 – 4.024	128 – 512
Frec. máxima interna	500 MHz	250 MHz	250 MHz	350 MHz	350 MHz
Voltaje	1.5 V	2.5 V, 3.3 V	3.3 V, 5 V	2.5 V, 3.3 V, 5 V	2.5 V, 3.3 V, 5 V
Interconexiones	*Antifuse*	*Flash*	*Antifuse*	*Antifuse*	*Antifuse*
Tecnología	0.15 μm 7 capas de metal CMOS	.22 μm 4 capas de metal CMOS	0.45 μm 3 capas de metal CMOS	0.22 μm CMOS	0.22 μm CMOS
SRAM bits	29 k – 339 k	14 k – 198 k	2.56 k	n.a.	n.a.

Tabla 2.5 Características de FPGAs fabricadas por Actel.

2.5 Lenguajes Descriptivos de *Hardware*

La forma tradicional de diseñar circuitos digitales es dibujar diagramas lógicos que contengan compuertas (SSI) y funciones lógicos (MSI). Sin embargo, a fines de 1980s y comienzo de 1990s este proceso de diseño presentaba limitaciones como así algunos problemas. *¿Como se puede dibujar diagramas esquemáticos que contienen cientos de miles o millones de compuertas?* Con la disponibilidad de los dispositivos lógicos programables para reemplazar sistemas donde se utilizaban integrados como los TTL, un nuevo enfoque para el diseño digital fue necesario. Las herramientas asistidas por computadoras son esenciales para diseñar circuitos digitales en la actualidad. Es claro que en las últimas décadas los ingenieros digitales de hoy diseñan sistemas digitales mediante la utilización de *software*! Esto es un importante cambio de paradigma del tradicional método empleado para el diseño de sistemas digitales[2].

Actualmente los diseñadores digitales usan *Lenguajes Descriptivos de Hardware* (HDLs) para diseñar sistemas digitales. Los lenguajes más utilizados son *VHDL* y *Verilog*. Ambos lenguajes descriptivos permiten al usuario diseñar sistemas digitales mediante la escritura de código que describen el comportamiento de un circuito digital. Este código puede ser utilizado tanto para *simular* la operación del circuito

y *sintetizar* también implementarse dicho circuito en un CPLD, una FPGA o en un circuito integrado de aplicaciones específica (ASCI).

2.5.1 VHDL

El lenguaje VHDL surgió como parte de un programa norteamericano denominado *Very High Speed Integrated Circuits* (VHSIC), a comienzos de 1980. En el desarrollo de la ejecución de este programa surgió la necesidad de contar con un lenguaje que permita describir la estructura y funciones para los circuitos integrados (ICs). Es así que el VHSIC *Hardware Description Language* (VHDL) fue desarrollado. Luego la IEEE adoptaría como un lennguaje estándar en los Estados Unidos.

VHDL fue diseñado para cubrir necesidades el proceso de diseño. Primero, este lenguaje permite la descripción de la estructura de un diseño, de esta forma se puede descomponer en sub-diseños, y a la vez como estos sub-diseños se interconectan entre sí. Segundo, VHDL permite la especificación de la función de los diseños usando las formas del lenguaje de programación similares a otros lenguajes familiares. Tercero, permite a un diseño ser simulado antes de ser fabricado, por lo que los diseñadores puede rápidamente compara alternativas y probar correciones sin el retardo y espera de los prototipos en *hardware*.

2.5.2 Verilog

Verilog esta basado en el lenguaje de programación C en la estructura de la sintaxis pero la manera en la que se comporta es diferentes pues es un lenguaje descriptivo. Este formato permitió una rápida aceptación por parte de los diseñadores de *hardware*.

Con el incremento en el éxito de VHDL, Cadence decidió hacer el lenguaje abierto y disponible para estandarización. Cadence transfirió Verilog al dominio público a través de Open Verilog International, actualmente conocida como Accellera. Verilog fue después enviado a la IEEE que lo convirtió en el estándar IEEE 1364-1995, habitualmente referido como Verilog 95.

2.6 Diseño de sistemas digitales

En el proceso de enseñanza de los sistemas digitales se requiere de recursos físicos que complementen el contenido teórico. En la carrera de ingeniería electrónica de nuestra casa de estudio, las técnicas digitales se clasifican en cuatro niveles:

Técnicas Digitales I el contenido comprende conceptos desde el Álgebra de Boole, funciones lógicas, sistema de numeración, codificadores/decodificadores, circuitos secuenciales, manejo de lenguajes descriptivos de *hardware*.

Técnicas Digitales II Métodos de discretización, convertidores AD/DA, microprocesadores, microcontroladores.

Técnicas Digitales III Instrumentación virtual, adquisición y acondicionamiento de señales (DAQ), redes de computadoras, DSP, sistemas lineales (convolución, correlación/autocorrelación y Fourier), interpolación/decimación (ventanas: rectangular, Hanning, Hamming, Blackman, Kaiser), filtros digitales.

Técnicas Digitales IV Arquitecturas de lógicas programables, sistemas de diseño para PLDs, procesado y mecanismos de simulación del lenguajes VHDL, síntesis, modelado con VHDL.

La tecnología lógica programable (PLD) es desarrollada en dos de las cuatro cátedras del área de *técnicas digitales*. A mediados de la década del 2000, se comenzó a introducir fuertemente la posibilidad de implementar los diseños digitales sobre dispositivos PLD. Lo que ha requerido la capacitación de los docentes sobre esta tecnología. El Centro Universitario de Desarrollo en Automoción y Robótica fue quién innovó sobre la formación de recursos humanos para la transferencia de conocimientos y desarrollos para los laboratorios. Tal es así la inserción y actualización del área que la cátedra electiva *Técnicas Digitales IV* surgió no hace más de cuatro años. Estos recursos tecnológicos fueron paulatinamente incluidos en las cátedras.

2.7 Influencia de la Programabilidad

En muchos textos la ley de Moore es usada para destacar la evolución de la tecnología de silicio en la industria de los dispositivos semiconductores. Poro hay otro interesante punto de vista particularmente para los dispositivos PLDs, la *onda de Makimoto* que fue publicada por primera vez en Enero de 1991 por la revista *Electronics Weekly* [3]. Este concepto se basa en la observación de Tsugio Makimoto quién notó que la tecnología se desplazaba entre la *estandarización* y la *personalización* (véase la Figura 2.7). En el comienzo de la década de 1960s, un número de componentes estándares fueron desarrollados, llamados series lógicas 7400 (por Texas Instruments). Estos dispositivos servían para crear diversas aplicaciones digitales. Entrada la década de

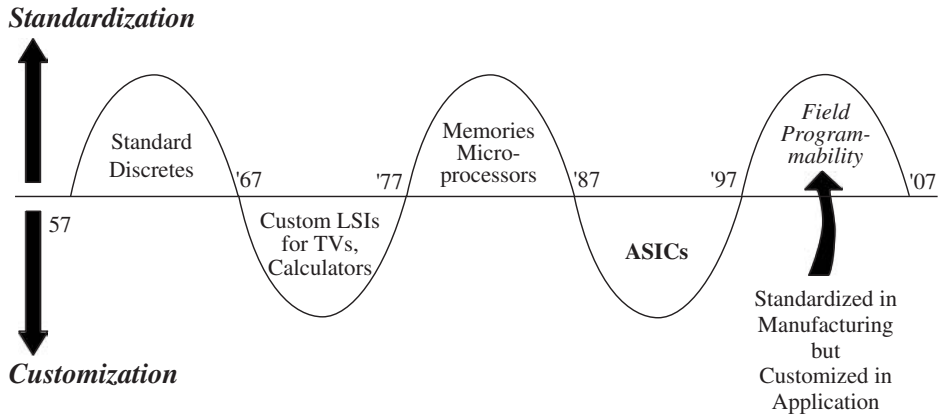


Figura 2.7 Onda de Makimoto.

1970s, la época de los dispositivos personalizados (LSI, siglas en inglés de *Low-Scape Integration*) comenzó a desarrollarse donde los chips eran creados para aplicaciones específicas como ser una calculadora. El chip fue incrementando su nivel de integración y así fue que nació el termino integración a media escala (MSI, siglas en inglés de *Medium-Scale Integration*). La evolución de los microprocesadores en la década de 1970s llevó a la estandarización de chips que fueran usados para un amplio rango de aplicaciones. Es entonces que en 1980s nació el ASIC (*Application-Specific Integrated Circuit*) donde el diseñador podría superar la limitación de la secuencialidad de los microprocesadores, quienes poseían varias limitaciones en aplicaciones en DSP (*Digital Signal Processing*) donde se requería un mayor nivel de cálculos. La aparición de la FPGA como un dispositivo con la capacidad de proporcionar recursos lógicos necesarios para conectar varios componentes entre sí llevo a que se conviertan en dispositivos populares.

Se podría considerar la existencia de dos épocas de la *programabilidad* donde la *primera* época ocurre con la aparición del microprocesador en la década de 1970s, aquí los programadores desarrollaban soluciones programables basados sobre dispositivos (*hardware*) fijos. El gran reto de la esta época fue el entorno de *software* ya que los desarrolladores trabajaban con lenguajes *assembly* e incluso cuando los compiladores y ensambladores surgieron para el lenguaje C, pues se obtenían mejores rendimientos con la codificación manual. Se comenzaron a obtener librerías que proporcionaban funciones básicas, permitiendo al diseñador concentrarse en la programación de la aplicación. Estas funciones actualmente son fácilmente accedidas desde los compiladores y ensambladores comerciales/libres. Actualmente hay una gran demanda de lenguajes de programación de alto-nivel como C y Java. Tal es así la abstracción del lenguaje que

incluso entornos de desarrollos de alto nivel como UML están siendo implementados.

La *segunda* época de la programabilidad se encuentra marcada por las FPGAs. En la Figura 2.7, Makimoto indica que el campo de la programabilidad se estandariza para su fabricación y la personalización del diseño se encuentra en la capa de aplicación de un desarrollo con las tecnologías mencionadas. Esto puede ser considerado como lo que ofrece la programabilidad de *hardware* en el dominio del *software* donde el *hardware* permanece fijo. Esto es un reto fundamental como la mayoría de las herramientas de programación de computadora que trabajan sobre el principio de una plataforma de *hardware* fijo, lo que permite realizar optimizaciones ya que hay una orientación clara sobre la manera de mejorar el rendimiento de una representación algorítmica. Con las FPGAs, el usuario tiene plena libertad para definir la arquitectura que mejor se adapte a la aplicación. Sin embargo, esto presenta un problema en el que cada solución debe ser *hecha a mano* y todos los diseñadores de *hardware* conocen los problemas en el diseño y verificación.

Algunas de las tendencias en las dos épocas tienen similitudes. En los primeros días, el modo esquemático (*schematic capture*) fue usado para diseñar los primeros circuitos que era sinónimo con el nivel *assembly* en programación. Los lenguajes de descripción de *hardware* como el VHDL y Verilog emergieron ya que podrían ser utilizados para producir un nivel de abstracción más alto con el objetivo de contar con una herramienta basada en C como son SystemC y CataultC de Mentor Graphics como un entorno único de programación. Inicialmente como con los lenguajes de programación de *software*, había una desconfianza en la calidad de los resultados que producía el código con este nuevo enfoque. Sin embargo, con el fin de mejorar los costos de desarrollo, las herramientas de síntesis que eran equivalentes a la evolución de los compiladores de *software* eficientes para los lenguaje de alto-nivel, y también la evolución de las funciones de librería, estableció un alto grado de confianza que posteriormente llevó al uso de los lenguajes descriptivos de *hardware* (HDLs) sean comnues para la implementación en FPGA. En efecto, el surgimiento de los IP-cores refleja la evolución de librerías como son funciones programables de entradas/salidas para el flujo del *software* donde funciones comunes fueron reutilizadas donde los desarrolladores confiaban en la calidad de los resultados que producían estas librerías, especialmente en lo que las presiones para producir más código en el mismo lapso de tiempo crecieron con la evolución tecnológica. Los primeros IP-cores surgieron a partir de funciones de librerías básicas en el procesamiento de señales complejas y funciones de comunicación la mayoría de estos suministrados por los proveedores de FPGA y diversos repositorios

web de IP-cores.

2.8 Actualización tecnológica de los recursos educativos

Los anteriores conceptos alientan a la búsqueda, por parte de las instituciones académicas, de nuevas herramientas y materiales educativos que permitan a los estudiantes manipular nuevas tecnologías. Esto demanda por parte de las autoridades académicas que permitan la incorporación/modificación de las cátedras a fines. En la publicación *"Actualización de la currícula - Incorporación de la lógica programable en ingenieria"* se plantea este reto [4]. En esta publicación se propone modificar ligeramente el contenido de ciertas materias de las carreras de Ingeniería Electrónica e Ingeniería en Sistemas de Información, de la Universidad Tecnológica Nacional, para adecuarlas a la renovación tecnológica de la electrónica de consumo actual, que si bien aún no ha inundado Argentina, al punto de hacer absolutamente obsoletas las técnicas digitales discreta y de integración moderada, y las metodologías de diseño de sistemas, no va a pasar demasiado antes de que sea necesario un cambio radical en la industria, en el mercado de consumo masivo, y si seguimos a este ritmo, en última instancia en la educación técnica y tecnológica de nivel terciario-universitario.

2.8.1 Análisis de contenidos en Ingeniería Electrónica e Ingeniería en Sistemas de Información

En Ingeniería Electrónica, se introduce a los alumnos a las tecnologías digitales de procesamiento de información en materias dictadas desde el primer año de cursado, a saber: Informática I y II, Técnicas Digitales I, II y III.

El programa de Informática I de 1er año, comienza por un pequeño despliegue de conocimientos someros de arquitectura de computadoras modernas, y continúa con la enseñanza de la resolución algorítmica más básica en programación, eligiendo primero los diagramas de flujo, luego el pseudocódigo, y por último el muy popular lenguaje C, de medio y bajo nivel, por lo que se pierde una gran cantidad de tiempo, dada la repetición de temas con distintas herramientas. Durante el 2do. año en Informática II, los alcances determinan una ampliación de habilidades en C, tales como las estructuras de datos entre otros temas, y por último un corto (de vista) repaso de conceptos de orientación a objetos, que no llega a concretarse en mucha práctica con C++.

En Técnicas Digitales I, de 3er año, se ven los principios de lógica digital hasta diseño secuencial, y se emplea en la parte práctica, tecnología de baja escala de integración, los muy conocidos y populares, TTL línea 7400 y CMOS línea 4000.

En Técnicas Digitales II, de 4o año, se estudia no con mucho detalle, la clásica arquitectura Intel X86.

En Técnicas Digitales III, de 5o año, se tocan de oído los temas referidos al desarrollo de sistemas de procesamiento digital de señales, mediante programación en C/C++ y Matlab sobre DSPs integrados o PCs con plataformas X86s.

En Ingeniería en Sistemas de Información, debe tenerse en cuenta que la orientación no es precisamente técnica, sino más bien gerencial-administrativa; sin embargo, en su base presenta suficientes materias de contacto con Ingeniería Electrónica, y sobre ellas discutiremos.

En Ingeniería en Sistemas de Información, durante el último cuatrimestre del 1er año, se dicta con mucho éxito la materia Algoritmos y Estructuras de Datos (con C/Java), que en contenido y extensión, abarca prácticamente el total de las dos Informáticas de Ingeniería Electrónica. También en este cuatrimestre, se dicta la materia Arquitectura de Computadoras, en la que se ven temas varios de su correcto nombre, sin pasar más abajo del nivel de lógica digital y llegando hasta la arquitectura de microprocesadores interna y externa. En el 2do. año, en la materia Paradigmas de la Programación, se llegan a ver entre otros, los paradigmas concurrentes (con Ada), orientado a objetos (Java/C++), funcional (Scheme/Lisp) y lógico (Prolog).

2.8.2 Propuesta de modificación de contenido

Lo primero que se propone en este caso, es unificar contenidos entre las materias de programación de los primeros años de ambas ingenierías, mas no tal vez su dictado conjunto, para poder hacer énfasis individuales sobre las especializaciones de estos temas en cada carrera por aparte.

Así, nos quedaría una primera materia Informática I dentro de Ingeniería Electrónica, equivalente a Algoritmos y Estructuras de Datos de Ingeniería en Sistemas de Información, pero de duración anual, lo cual puede ser una ventaja para la maduración de los temas, aunque de carga horaria equivalente similar, la mitad por semana dado su doble extensión temporal anual.

En el 2do. año en Ingeniería Electrónica, la materia Informática II, también anual, tendría tiempo suficiente para incluir extensamente el paradigma orientado a objetos con C++, muy importante para la formación del ingeniero electrónico, y tan impor-

tante como el anterior, incluir el paradigma concurrente, enseñado mediante ejemplos en VHDL con simuladores, para de esta forma, preparar al alumno para las técnicas digitales sobre FPGAs, sin tener que comenzar desde cero con un lenguaje de descripción de hardware, en el 3er año.

En cuanto a Ingeniería en Sistemas de Información, en la materia Arquitectura de Computadoras, se podría introducir descripciones de microprocesadores y otros circuitos en VHDL con simuladores, facilitando la ampliación del paradigma concurrente en VHDL/Ada del siguiente año. Ya en Paradigmas de la Programación del 2do. año, podría emplearse el lenguaje VHDL junto al Ada (dado que este último es un antecesor del primero), para unificar temas entre materias de los dos niveles y compartir contenidos por un lado, y por otro, para mostrar a los alumnos plataformas de hardware (y no solo las X86s) que se programan directamente con ese paradigma, a la cual algunos tal vez podrían dedicar esfuerzos en el futuro, siendo también una forma de entablar algunos puentes de unión entre estas dos carreras parientes, aunque no cercanos.

Como segundo paso, convendría la incorporación temprana en Técnicas Digitales I, de temas de lógica reconfigurable, y dado que VHDL ya se habría visto con mediana intensidad durante el año anterior, se puede concentrar la atención en el diseño de hardware, y en la descripción del mismo mediante estas técnicas, algo ajenas al diseño discreto convencional, del que se puede ver un poco, tan solo para no dejar de lado, que aun hoy una buena parte de la industria tiene implantados con esta tecnología mucho de su infraestructura, a la cual sin duda deberá darse mantenimiento; sin embargo el siguiente paso es la modernización de esa misma infraestructura con estas nuevas herramientas, para lo cual este cambio sería instrumental.

Luego en Técnicas Digitales II, se puede ver no solo la arquitectura X86, sino otras variantes tan importantes en la industria como ella, directamente en VHDL con implementación sobre placas de trabajo construidas alrededor de lógica programable, con lo cual no se queda en teoría el diseño ni de microprocesadores (irrealizable con electrónica discreta salvo en simulación) ni de arquitectura externa al micro.

Finalmente en Técnicas Digitales III, todo los temas referidos al procesamiento digital de señales, se pueden aprender con VHDL y arquitecturas construidas en FPGAs, dado que son en muchos casos la elección prioritaria, complementándolas con tecnología DSP integrada o embebida en FPGAs.

En la Figura 2.8, podemos ver un esquema sintético de la propuesta.

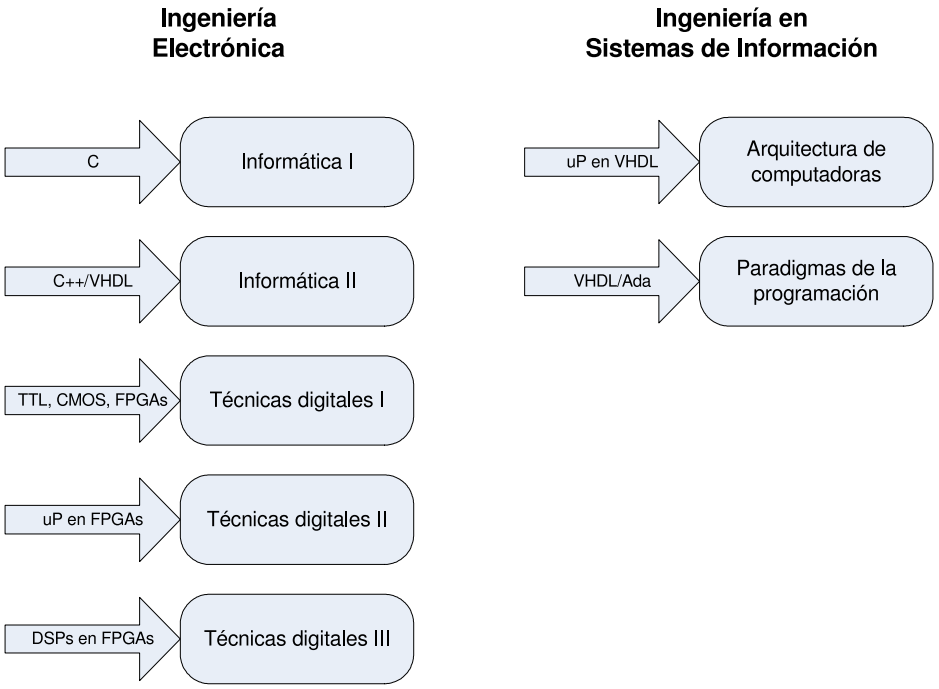


Figura 2.8 Propuesta de modificación de contenido a las carreras de Ingeniería Electrónica e Ingeniería en Sistemas de Información.

2.8.3 Un atisbo de cambio en nuestra carrera

En el CUDAR (Centro Universitario de Desarrollo en Automación y Robótica), decidimos colaborar con la incorporación de estas modificaciones, en principio, limitados a las materias de Técnicas Digitales de Ingeniería Electrónica.

Se lo propusimos a los profesores titulares de cada materia, sin la necesidad de cambiar los programas, pero incorporando las herramientas y el lenguaje VHDL en las tres. Estuvieron de acuerdo y propusieron un cambio mayor en los contenidos, a lo que nosotros dimos nuestro apoyo.

Para ello, el secretario de Ciencia y Técnica de la Facultad Regional Córdoba, instrumentaría un presupuesto, cuando sea posible, para el equipamiento de laboratorios con tecnología de lógica programable, es decir con placas experimentales y estaciones de trabajo con software adecuado.

Además, para poder realizar esta renovación, puede recurrirse a los programas universitarios de empresas como Xilinx [5] y Altera [6], que donan hardware y disponen de entornos de desarrollo y simuladores gratuito.

Otra alternativa que evaluamos fue el diseño en casa, de pequeñas placas de trabajo construidas con CPLDs, aunque nos topamos con la dificultad de la falta de provee-

dores adecuados. Finalmente las diseñamos y probamos, pero nos encontramos que al querer conseguir CPLDs y FPGAs de bajo precio nominal, los distribuidores para Argentina de Xilinx y Altera no dan gran importancia a los programas universitarios de esas empresas, generando además sobreprecios, aún cuando los precios sean bajos originalmente, con detalles como compra de una cantidad mínima por componente, gastos de envío entre Buenos Aires y el interior con valores de transporte internacional, etc.

2.8.4 Resultados

A partir del año 2006, colaboramos activamente en la incorporación de temas de lógica programable en el dictado de la materia Técnicas Digitales I, a través del titular de la cátedra y un profesor de trabajos prácticos, ambos integrantes del CUDAR, a los que asistimos con material de lectura, prácticos y exposiciones preparadas en filminas sobre la tecnología interna de los dispositivos programables.

También hemos terminado con las etapas de diseño y pruebas de un Kit de desarrollo educativo con CPLD [7], que dejamos disponible libremente para su construcción por parte de los alumnos, y a partir de este año se utilizará en la materia Técnicas Digitales I como herramienta de trabajos prácticos. Asimismo, el departamento Ingeniería Electrónica ha dispuesto incorporar este kit en el Laboratorio de Técnicas Digitales, esfuerzo que está siendo coordinado por nosotros desde el CUDAR.

A propuesta nuestra también, a partir de 2007, dictaremos la materia electiva del 6to. año de Ingeniería Electrónica, dedicada a la lógica programable, y bautizada como Técnicas Digitales IV.

Capítulo 3

Antecedentes

Una de las principales actividades de los centros y grupos de investigación es la generación de recursos y contenidos académicos. Si bien las líneas de trabajo son definidas por una secretaría a nivel regional, en muchos casos la transferencia de investigación y desarrollo (I+D) beneficia a las mismas instituciones académicas permitiendo la actualización tecnológica con los resultados arribados por dichos centros.

El *Centro Universitario de Desarrollo en Automoción y Robótica* (CUDAR) desarrolla sus actividades en la Universidad Tecnológica Nacional – Facultad Regional Córdoba. Las principales actividades y áreas del Centro se puede clasificar como,

- Investigación y Desarrollo
  - Área Electrónica
  - Área Mecánica
  - Área Informática
- Extensión y Tecnología
  - Convenios y transferencias
  - Cursos de capacitación
  - Consultoría y servicios

En la búsqueda de nuevas tecnologías es que miembros del CUDAR comienzan el estudio de la tecnología PLD en la década del 2000. La principal actividad se centra en la participación de congreso y seminarios relacionados con la lógica programable.

Con el proyecto *Robot Industrial de Arquitectura Reconfigurable Implementado con FPGA* desarrollado en el CUDAR se institucionalizó la investigación y se comenzó a realizar desarrollos relacionados con dispositivos reconfigurables como CPLD y FPGA. Estos esfuerzos en la formación y adquisición de nuevas tecnologías llevó a la actualización de la cátedra *Técnicas Digitales I*, obviamente que fue necesaria la introducción de estos nuevos conceptos debido a la posibilidades que se tenían para adquirir estos

dispositivos que anteriormente no se encontraban comercializados en forma tan masiva en nuestra región. Hace menos de cinco años el CUDAR impulsó la creación de la cátedra electiva *Técnicas Digitales IV* con el objeto de dar continuidad a los estudiantes de ingeniería electrónica interesados en especializarse sobre la implementación de sistemas complejos basados en sistemas digitales reconfigurables.

Para acompañar el proceso de formación tanto en el CUDAR como en las cátedras de grado se requirió contar con recursos de *hardware* donde implementar los sistemas digitales diseñados. Si bien se logró adquirir plataformas educativas, estratégicamente el CUDAR, desde el comienzo, dispuso el desarrollo de plataformas propias que lograran cubrir las necesidades del momento en materia de periféricos y otras cuestiones.

3.1 Placa MiniLab

En la cátedra de Técnicas Digitales I, la primera aproximación con los sistemas digitales físicos es el armado del denominado *Proyecto MiniLab*. El circuito del MiniLab es una herramienta para los trabajos prácticos de diferentes cátedras de electrónica que cursan los alumnos del tercer nivel de la carrera ingeniería electrónica. Y es que esta plataforma permite montar y testear circuitos con relativa facilidad. Dispone recursos digitales básicos con el fin de realizar prueba, tiene señales lógicas de entrada ('0' y '1') y también cuenta con indicadores LED que puede ser utilizados como salidas digitales. Además de entrada/salidas se dispone de un generadores de pulsos de reloj. En la Figura 3.1 se puede ver una fotografía del proyecto MiniLab.

Sobre esta plataforma multi-propósito se montan diferentes circuitos integrados (ICs) con compuertas (NOT, OR, AND, XOR, etc.) que permiten al estudiante implementar las diferentes funciones digitales a medida que avanza el contenido de la materia. Es aquí el primer antecedente sobre la disposición de recursos físicos vinculados con el contenido académico. Al transcurrir los años consecuentemente los avances tecnológicos demandaron, por parte de plataformas como el MiniLab, ofrecer nuevos recursos. Ya el MiniLab resultaba limitado para los estudiantes iniciales. Un factor clave resulta ser la decisión de introducir en la cátedra de Técnicas Digitales I los conceptos de lógica programable y acercar a los estudiantes a esta nueva tecnología, complementando así los recursos que dispone el MiniLab.

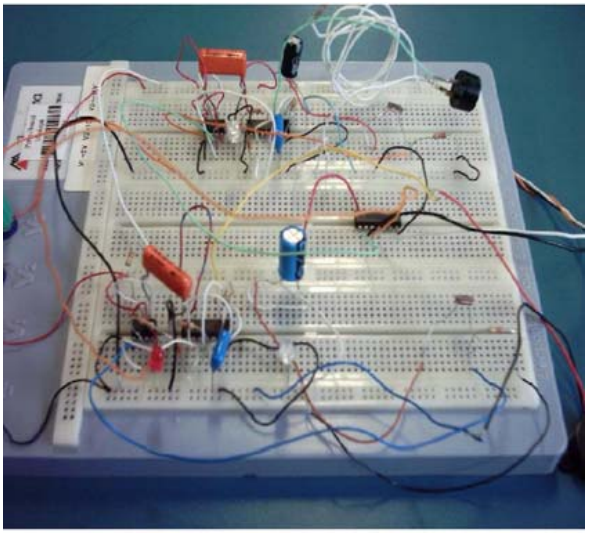


Figura 3.1 Implementación de un circuito electrónico montado sobre le *MiniLab*.

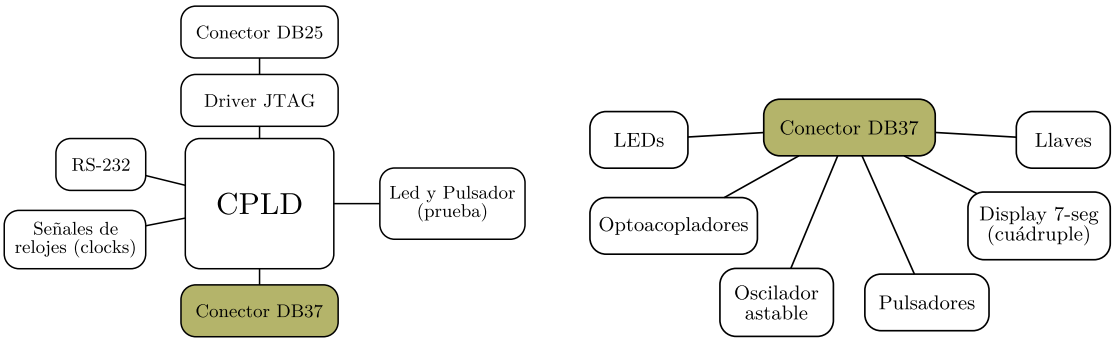
3.2 Kit CPLD

El *Kit CPLD* desarrollado por el CUDAR en el año 2005, publicado posteriormente en el año 2006, se presenta como un *Equipo de Laboratorio basado en CPLD*. Esta plataforma permite simular y desarrollar circuitos digitales basados en lógica programable. Optimizado para reducir el costo de su fabricación, el Kit CPLD permitía introducir a los estudiantes en la tecnología PLDs a través de los lenguajes de descripción de hardware (principalmente VHDL).

El diseño consta de dos bloques funcionales perfectamente acotados y cada uno de ellos está implementado sobre una placa independiente una de la otra. En la Figura 3.2 se representa en un diagrama los dos bloque que forman el Kit CPLD. Estos bloques se denominarán de aquí en adelante como *bloque programador* (Fig. 3.2a) y *bloque expansión I/O* (Fig. 3.2b). El primer bloque resulta indispensable en este Equipo y tiene las siguientes particularidades. Alojar un CPLD con la electrónica necesaria para poder ser programarlo (JTAG), comunicación en forma serial para usos diversos y conectores que interconectan los pines físicos de entrada y salida del chip CPLD con el exterior. El otro bloque se puede considerar de uso opcional debido a que es una extensión del primer bloque y es utilizado principalmente como interfase entre el usuario y la lógica con la cual será programado el CPLD.

Las principales aplicaciones que se planteaba con este desarrollo fueron:

Aprendizaje Es su misión fundamental. Que los alumnos o gente que esta aprendiendo pueda disponer de una plataforma sencilla, y fácil de usar. Se puede



(a) Placa principal con el CPLD (b) Placa expansión, conector DB-37

Figura 3.2 Diagrama en bloque de la primera versión del *Kit CPLD*.

disponer de todos los esquemas, los estudiantes pueden comprender todos los detalles y construirse sus propias entrenadoras o reproducir esta misma, bien en modo prototipo o bien mandando fabricar el mismo PCB.

Conexión con microcontroladores Por tratarse de un sistema autónomo es muy útil para desarrollar periféricos para diferentes microcontroladores: 6811, 6808, PIC, etc.

Robótica Muy útil en la construcción de robots autónomos o periféricos para ellos

Investigación Posibilidad de implementar distintos tipos de lógica , que luego funcionaran de manera autónoma y no se borra la programación cuando se le retira la alimentación.

***Hardware* abierto** La posibilidad de que esta placa se convierta en un sistema de desarrollo libre empleado por desarrolladores de hardware libre, que diseñen cores o hardware para estos CPLD con una licencia libre. Esto hará, al igual que en el case del software libre, que todos nos beneficiemos de las aportaciones que otras personas hacen a la comunidad.

Para poder cumplir con el objetivo de pensar en un prototipo para docencia enfatizando en lo económico, se realizó el diseño del circuito en PCB en simple faz. Esto hace que el circuito pueda ser construido por un alumno utilizando procesos hogareños para la fabricación de placas electrónicas. Los componentes utilizados se pueden conseguir fácilmente en los comercios locales. Los circuitos esquemáticos como también el PCB son de libre difusión. En la Figura 3.3 se presenta una fotografía de las dos placas que componen esta primera versión del Kit CPLD.

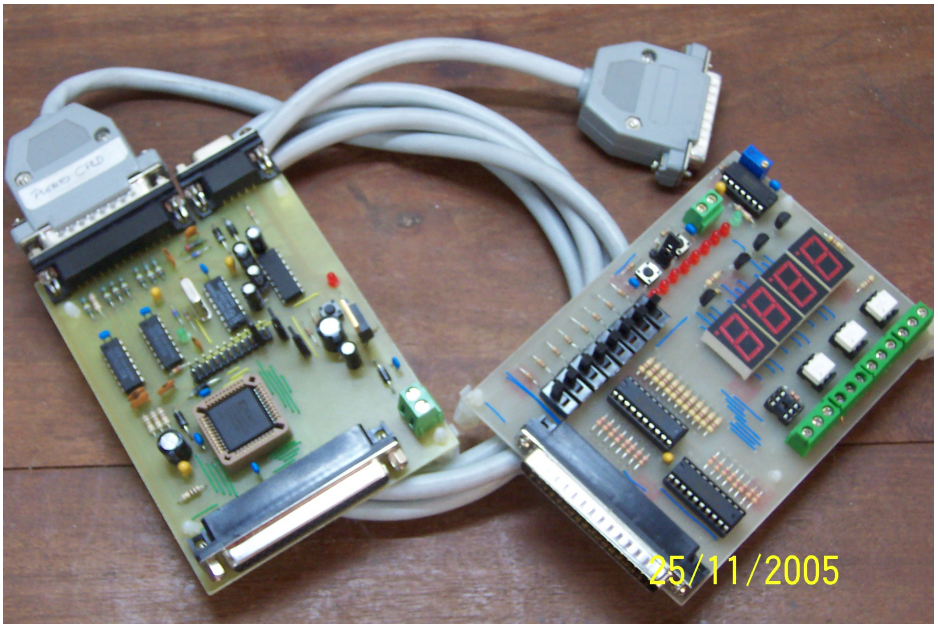


Figura 3.3 Fotografía de la plataforma *Kit CPLD* desarrollada en el CUDAR.

Capítulo 4

Desarrollos de Referencia

En la actualidad todos los desarrollos en recursos físicos basados en dispositivos lógicos programables son de origen extranjero. Se puede encontrar desarrollos nacionales pero estos no son comercializados por ningún emprendimiento local. Esta situación no es necesariamente por falta de recursos humanos calificados, sino por la demanda de la industria nacional. De todas formas el proyecto *Plataforma de Hardware Reconfigurable* pretende abrir camino hacia el desarrollo de plataformas educativas que permitan la difusión del diseño digital utilizando estos dispositivos PLDs.

Los desarrolladores de este proyecto, la mayoría de ellos miembros investigadores del CUDAR[1], cuentan con experiencia en el manejo de plataformas evaluadoras basadas en CPLDs/FPGAs. Además, como se describió en el Capítulo 3, el Centro tiene antecedentes en el desarrollo de plataformas educativas y la transferencia de recursos físicos a Laboratorios e Instituciones Académicas.

4.1 Desarrollos comerciales

En el mercado internacional existen varias empresas desarrolladoras de plataformas evaluadoras con dispositivos PLDs. Las principales empresas fabricantes de sistemas embebidos basados en dispositivos PLDs son Xilinx, Altera y Digilent. Estos fabricantes desarrollan plataformas para determinados perfiles de usuarios. En la Sección 2.7 se realiza un análisis sobre la importancia de la flexibilidad que presentan los dispositivos programables como las FPGAs, y esta posibilidad de reconfigurar su arquitectura permite implementar esta tecnología en diferentes aplicaciones. Las líneas

[1] Centro Universitario de Desarrollo en Automoción y Robótica – Universidad Tecnológica Nacional - Facultad Regional Córdoba.

de desarrollo más destacadas son,

- Sistemas de comunicaciones digitales
- Procesamiento de Señales Digitales (DSP)
- Automoción

A continuación se describe algunas de los desarrollos fabricados por las empresas anteriormente nombradas. De esta manera se pretende dilucidar los perfiles de plataformas y lo que se ofrece en el mercado.

4.1.1 Xilinx Spartan-6 FPGA LX9 MicroBoard

La plataforma *Spartan-6 FPGA LX9 MicroBoard* ofrece un completo entorno de *hardware* para que los diseñadores aceleren su tiempo de desarrollo y comercialización [8]. Este kit presenta una estable plataforma para desarrollar y testear diseños de bajo costo/consumo de potencia sobre la familia de FPGA Xilinx Spartan-6. Esta placa contiene las siguientes características[2].

- FPGA
  - Xilinx Spartan-6 XC6SLX9-2CSG324C FPGA
- Clocks
  - Triple output, user programmable, Texas Instruments CDCE913 clock
  - Optional user installable Maxim DS1088LU-66+, low-cost, fixed-frequency oscillator
- Memory
  - 32 Mb x 16 (512 Mb) Micron LPDDR Mobile SDRAM component.
  - 128 Mb Micron Multi-I/O SPI Flash
- Communication
  - One USB 2.0, Full Speed USB-to- JTAG bridge via Atmel AT90USB162, Digilent JTAG firmware, and Tyco USB-A connector

[2]La características se las describe tal cual las hojas de datos publicadas por los fabricantes, con su idioma original.

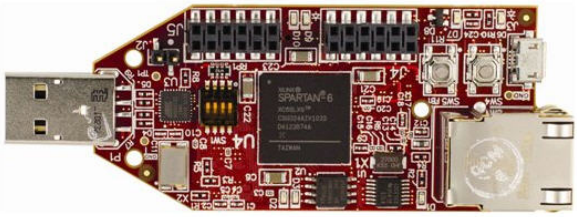


Figura 4.1 Placa Xilinx Spartan-6 FPGA LX9 MicroBoard.

  - One USB 2.0, Full Speed USB-to-UART bridge via Silicon Labs CP2102 and Tyco Micro-B connector.
  - One 10/100 Ethernet port via National Semiconductor DP83848J PHY and Tyco RJ45 connector with Integrated Magnetics.

- User I/O and Expansion Connectors
  - Two Digilent 12-pin, 0.245mm pitch, Peripheral Module (PMOD) headers support 3rd party expansion modules.

- User Interfaces
  - Four user LEDs
  - Four configurable FPGA user DIP switches
  - Two system push-button switches: one tied to user I/O and used for logical reset in the factory test image, one hardwired for FPGA program initialization.

- Power
  - Texas Instruments TPS65708 PMU multi-channel regulator, with 5V input supplied by either USB connection.

- Configuration
  - 128Mb SPI Configuration Flash
  - On-board USB Programming/Configuration based on the Digilent USB Full Speed JTAG design utilizing the Atmel AT90USB162
  - Xilinx Compatible JTAG Cable

4.1.2 Altera DE0-Nano

La placa *DE0-Nano* presenta una plataforma de desarrollo FPGA en un tamaño compacto adecuado para un gran rango de diseños portables, tales como robótica y móviles. EL DE0-Nano es ideal para usar con *soft processors* embebidos. Esto se debe a las potentes características que presenta con la FPGA Altera Cyclone IV (con 22,230 celdas lógicas), 32MB de SDRAM, 2 Kb de EEPROM, y un dispositivo de memoria de configuración serial de 64Mb. En fin, es un desarrollo completo con todos los recursos físicos necesarios para la evaluación y prueba de sistemas embebidos [9]. A continuación se listan lo que ofrece esta plataforma.

- Featured device
  - Altera Cyclone® IV EP4CE22F17C6N FPGA
  - 153 maximum FPGA I/O pins
- Configuration status and set-up elements
  - On-board USB-Blaster circuit for programming
  - Spansion EPCS64
- Expansion header
  - Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins
- Memory devices
  - 32MB SDRAM
  - 2Kb I2C EEPROM
- General user input/output
  - 8 green LEDs
  - 2 debounced pushbuttons
  - 4-position DIP switch
- G-Sensor
  - ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)

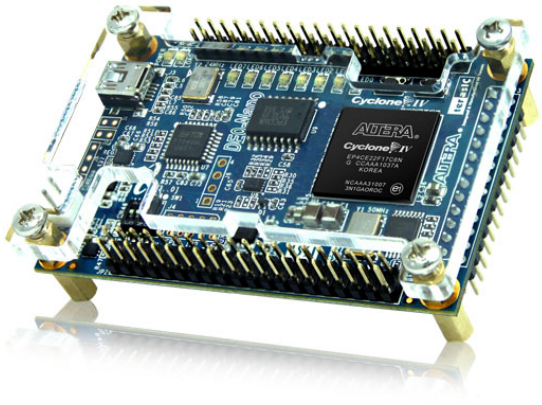


Figura 4.2 Placa DE0-Nano.

- A/D Converter
  - NS ADC128S022, 8-Channel, 12-bit A/D Converter
  - 50 Ksps to 200 Ksps
- Clock system
  - On-board 50MHz clock oscillator
- Power Supply
  - USB Type mini-AB port (5V)
  - DC 5V pin for each GPIO header (2 DC 5V pins)
  - 2-pin external power header (3.6-5.7V)

4.1.3 Digilent Spartan-3 Board

La placa *Spartan-3 Starter Board* proporciona una potente y autónoma plataforma de desarrollo para diseños basados en la FPGA Spartan-3 de Xilinx. Ésta cuenta con 200K compuertas, dispositivos de entrada y salidas, y una memoria SRAM rápida de 1MB; haciendo esta una perfecta plataforma para experimentar con diferentes diseños desde simples circuitos lógicos a implementación de procesadores embebidos [10]. La placa también contiene un dispositivo PROM programable por JTAG, por lo que los diseños puede fácilmente ser no volátiles. Las características de esta plataforma se describen a continuación.

- 200,000-gate Xilinx Spartan-3 XC3S200 FPGA in a 256-ball thin Ball Grid Array package (XC3S200FT256)
  - 4,320 logic cell equivalents
  - Twelve 18K-bit block RAMs (216K bits)
  - Twelve 18x18 hardware multipliers
  - Four Digital Clock Managers (DCMs)
  - Up to 173 user-defined I/O signals
- 2Mbit Xilinx XCF02S Platform Flash, in-system programmable configuration PROM
- 1Mbit non-volatile data or application code storage available after FPGA configuration
- 1M-byte of Fast Asynchronous SRAM
  - Two 256Kx16 ISSI IS61LV25616AL-10T 10 ns SRAMs
  - Configurable memory architecture
    - Single 256Kx32 SRAM array, ideal for MicroBlaze code images
    - Two independent 256Kx16 SRAM arrays
  - Individual chip select per device
  - Individual byte enables
- 3-bit, 8-color VGA display port
- 9-pin RS-232 Serial Port
  - DB9 9-pin female connector (DCE connector)
  - RS-232 transceiver/level translator
  - Uses straight-through serial cable to connect to computer or workstation serial port
  - Second RS-232 transmit and receive channel available on board test points
- PS/2-style mouse/keyboard port
- Four-character, seven-segment LED display

- Eight slide switches
- Eight individual LED outputs
- Four momentary-contact push button switches
- 50 MHz crystal oscillator clock source
- Socket for an auxiliary crystal oscillator clock source
- FPGA configuration mode selected via jumper settings
- Push button switch to force FPGA reconfiguration (FPGA configuration happens automatically at power-on)
- LED indicates when FPGA is successfully configured
- Three 40-pin expansion connection ports to extend and enhance the Spartan-3 Starter Kit Board
  - See www.xilinx.com/s3boards for compatible expansion cards
  - Compatible with Digilent, Inc. peripheral boards
  - FPGA serial configuration interface signals available on the A2 and B1 connectors
    - PROG_B, DONE, INIT_B, CCLK, DONE
- JTAG port for low-cost download cable
- Digilent JTAG download/debugging cable connects to PC parallel port
- JTAG download/debug port compatible with the Xilinx Parallel Cable IV and MultiPRO Desktop Tool
- AC power adapter input for included international unregulated +5V power supply
- Power-on indicator LED
- On-board 3.3V, 2.5V, and 1.2V regulators

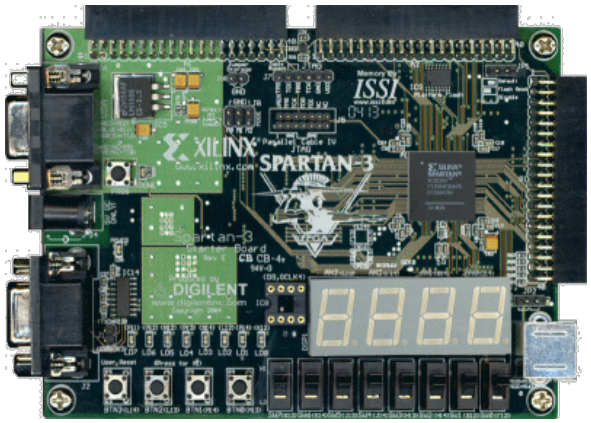


Figura 4.3 Placa S3BOARD.

Todos los desarrollos anteriormente vistos, como se mencionó al comienzo de este Capítulo, son manufacturados en el exterior del país. Los requerimientos tecnológicos para poder lograr estos productos son complejos pero no necesariamente alejado a la realidad de nuestra región. En el país se tiene algunas experiencias de plataformas basadas en FPGA.

4.2 Desarrollos nacionales

En nuestra región las tecnologías PLD se encuentran integradas en varias líneas de investigación y desarrollos hace algunos años. Instituciones gubernamentales de defensa como es el *Instituto de Investigación Científica y Técnicas para la Defensa* que en una de sus líneas de desarrollo implementa algunos sistemas electrónicos de adquisición de datos controlados con FPGA [11]. Así también otros organismos vinculados al área aeroespaciales y comunicaciones como la *Comisión Nacional de Actividades Espaciales* están implementando dispositivos como FPGAs y CPLDs en sus sistemas electrónicos [12]. Además existe una constante actualización por parte de las instituciones académicas en los programas analíticos de las carreras relacionadas a los sistemas embebidos. El Instituto Nacional de Tecnología Industrial impulsa un proyecto denominado *FPGALibre*. Este proyecto busca desarrollar y brindar herramientas de *software* libre y diseños de *hardware* abiertos para trabajar con tecnologías FPGA.

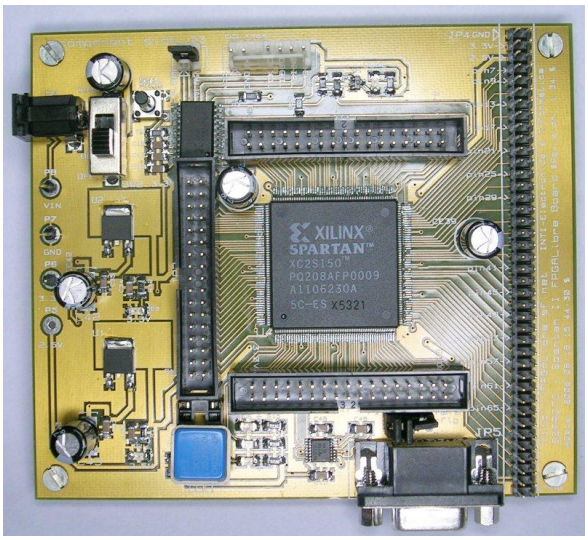


Figura 4.4 Placa S2PROTO.

4.2.1 S2PROTO

Desarrollado por el *Instituto Nacional de Tecnología Industrial* (INTI), el S2PROTO es el resultado de un laboratorio de *Software Libre* de dicha organización estatal. La *Unidad Técnica en Informática y Control* (UTIC) sostiene un proyecto libre llamado FPGALibre que alberga a desarrolladores que tengan intensiones de compartir sus proyectos basados en PLDs [13].

Algunas de las principales características del circuito son,

- Diseño brindado bajo licencia GPL para permitir su libre utilización, implementación, modificación y comercialización.
- Diseño ensamblado y probado.
- Desarrollado y probado con herramientas de software libre: Kicad y GNU jtag.
- Impreso doble faz de 12x10 cm de fácil fabricación en el país y posibilidad de montaje manual de los componentes.
- Soporte para dispositivos Xilinx Spartan II PQ208 (hasta XC2S150 equivalente a 150.000 compuertas).
- Puerto de niveles RS-232 para implementar transmisión y recepción serie.

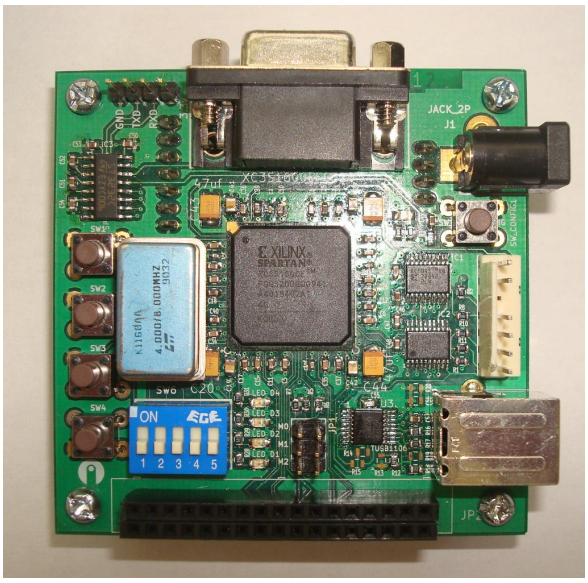


Figura 4.5 Placa S3PROTO-MINI.

- Puerto JTAG compatible con el cable Parallel III de Xilinx, también conocido como DLC5. Circuito realizado en Kicad, y también disponible en el proyecto FPGA Libre.
- 152 pines de I/O disponibles para el usuario, distribuidos en cinco conectores. Estos permiten conectar plaquetas adicionales con pulsadores, leds, displays, etc.
- Conector USB tipo B y driver para soportar niveles estándares USB.
- Alimentación simple de 5 V CC.

4.2.2 S3PROTO-MINI

Continuando con las ideas abordadas en la S2PROTO se encaró un proyecto con Spartan-3 y encapsulado BGA. El primer resultado es la S3PROTO-MINI.

La tarjeta S3PROTO-MINI es una plataforma básica y simple de hardware libre para desarrollo con FPGA. Posee un dispositivo Xilinx Spartan 3E (XC3S1600E) con encapsulado BGA de 320 pads. Utiliza el módulo de alimentación triple S3Proto, también disponible como hardware libre [14].

Las principales características de esta plataforma son,

- Dispositivo FPGA Xilinx Spartan 3E (XC3S1600E) de 33.192 celdas lógicas.
- 2 Memorias de configuración XCF04S (4+4 Mbit).

- USB Transceiver TUSB1106 de 12 Mb/s (Full Speed) con conector tipo B.
- 2 Puertos seriales RS232 de hasta 300Kbps (ST3232). Uno con conector DB-9.
- 4 Pulsadores.
- 5 Dip switch.
- 4 LEDs.
- 1 Puerto JTAG.
- 26 Pines de I/O.
- Oscilador con zócalo.
- Alimentación simple de 5V.
- Dimensiones de 7x7 cm.

Estas primeras experiencias locales resulta alentadoras para realizar nuestros propios recursos académicos para la enseñanza de tecnologías programables como son los PLDs. Si bien los proyectos realizados por el INTI aún no se encuentran comercializados, se encuentran antecedentes de la transferencia de estos desarrollos a instituciones académicas.

Tantos los desarrollos comerciales como los académicos presentan perfiles de *hardware* similares y no es casualidad pues son los recursos que se necesitan para la enseñanza y ejercitación en un ámbito académicos y de protipado. Por lo tanto son estos diseños que nos sirven como referencias para plantear y definir que recursos físicos se debe disponer para la *Plataforma de Hardware Reconfigurable*.

Capítulo 5

El proyecto PHR

5.1 Estructura general del proyecto

Al comienzo de la década de los 90s surgieron varios trabajos donde se planteaba la necesidad de una plataforma educativa orientada a la implementación de diseños lógicos digitales basados en PLDs. Los principales demandantes eran diseñadores de arquitecturas de microprocesadores, desarrollos que años anteriores resultaban dificultosos por el costo de la implementación en *hardware*. El avance en el proceso de integración de los circuitos integrados ha llevado a que se desarrollen plataformas más complejas que ofrecen una gran cantidad de recursos de hardware. Al día de hoy se han generado varios proyectos desarrollados por instituciones académicas, otras con especificaciones abiertas y también con fines comerciales [15]. Todos estos trabajos tienen algunas características en común[1]:

- El dispositivo lógico programable central es una FPGA
- Poseen Memoria de configuración de la FPGA
- El acceso al dispositivo es a través de JTAG
- Disponen de algún software para interactuar con la plataforma desde una computadora
- Tienen dos perfiles de diseño:

[1]La caracterización anterior no es un intento de generalizar a todas las plataformas educativas basadas en PLDs, pero sí resulta útil para definir el perfil de la plataforma que se describe en este trabajo.

Tabla 5.1 Recursos de hardware en función de los niveles de aprendizaje

Nivel	Llaves/pulsadores Diodos LED	ADC&DAC/SPI Display LCD/VGA	USB/ETH HDMI
Inicial	✓		
Medio	✓	✓	
Avanzado	✓	✓	✓

- Para la implementación de sistemas lógicos generales
- Orientado a un área específica

En función del perfil del usuario de la plataforma se definen los dispositivos que se utilizarán. La Tabla 5.1 ilustra una clasificación de los recursos que ofrecen diferentes plataformas basada en dispositivos PLDs. A niveles iniciales en el estudio de la lógica digital se requieren periféricos básicos como ser llaves conmutadoras de estados lógicos, pulsadores, dispositivos indicadores como diodos LED, etc. A un nivel medio se manejan controladores para display gráficos LCD/LED, comunicaciones entre varios dispositivos mediante SPI, I2C, etc. Y por último, en la formación de especialistas de sistemas embebidos, se requieren recursos como interfaces físicos para ethernet, controladores HDMI, USB, y otros más.

La estructura del proyecto *Plataforma de Hardware Reconfigurable* debe ser un proyecto a medida de las necesidades en la enseñanza de los sistemas digitales lógicos en las cátedras iniciales. Se debe ofrecen recursos básicos para que los estudiantes interactúen con la tecnología de los dispositivos PLDs, pero también dispone de puertos para conectar otros recursos físicos permitiendo que estudiantes avanzados puedan hacer uso de ellas sin limitaciones. Al ser publicado bajo licencia libre/abierta permitirá que el diseño, o parte de él, sirva como referencia a otras instituciones académicas que se encuentren en búsqueda de una plataforma para implementar en sus diferentes cátedras.

5.2 Consideraciones sobre la estructura de las placas

Las dimensiones y disposiciones de las diferentes placas que forman este proyecto también ha requerido un previo análisis por parte de los desarrolladores. Si bien parece

un tema trivial, las consecuencias de las definiciones a tomar implican alteraciones tanto funcionales como económicas.

Como se describió en la Sección 5.1, en esta parte del desarrollo se conoce que dispositivos electrónicos se van a incluir en nuestro proyecto como así también las dimensiones y distribuciones que se puede tomar. Los principales dispositivos que requieren mayor importancia y que definirán la estructura física del diseño son:

- Dispositivo Lógico Programable (FPGA)
- Memoria de programación
- Interfaz USB-JTAG
- Sistema de alimentación
- Periféricos

Las primeras observaciones que se hicieron sobre estos puntos fue la posibilidad de dar el mayor uso y flexibilidad a todos los recursos a implementar. Por ejemplo, para el caso del interfaz USB-JTAG se podría implementar en forma aislada a la placa principal del proyecto. Eso permitiría que reutilizar esta interfaz con otros proyectos. Pero si esto se realizara se incrementaría el costo del proyecto pues se necesitaría fabricar una placa aislada que, si bien puede que ocupe el mismo área que si se integraría el diseño a la placa principal, los costos de fabricación son mayores. Además, por el perfil del proyecto, se intentó implementar diseños publicados en forma libre por la comunidad de *Hardware Libre*. Un ejemplo de esto es la decisión de utilizar el sistema de alimentación desarrollado por el Instituto Nacional de Tecnología Industrial, lo que se describirá con mayor detalle en secciones futura. En definitiva, todos los análisis llevaron a la diseño e implementación de tres placas que integran todos los requerimientos planteados al iniciar el desarrollo.

5.3 Selección de dispositivos principales

Los dispositivos principales del proyecto, enunciados en la sección anterior, deben ser definidos al comenzar con el desarrollo. Una vez que se decida porque dispositivos utilizar, los demás componentes electrónicos serán funcionales a estos primeros.

5.3.1 Dispositivos FPGA

La FPGA que se utiliza pertenece a la familia Spartan-3 de Xilinx Inc. Esta familia a la vez se clasifican en [16]

- Familia Spartan-3A extendida (bajo costo):
  - Spartan-3A
    - Ideal para uso de interfaz entre dispositivos.
  - Spartan-3A DSP
    - Mayor densidad de recursos en comparación que la familia Spartan-3A
    - Dispone de un dispositivo DSP (DSP48A)
  - Spartan-3AN
    - Dispositivos no volátiles
    - Ideal para aplicaciones con restricciones de espacio
- Familia Spartan-3E
- Familia Spartan-3

Altera, Atmel y otros fabricantes de FPGAs también presentan familias similares a las Spartan-3. Aquí se optó por Xilinx Inc. debido a la experiencia en software/hardware con que cuenta el Centro de Investigación[2] donde se desarrolla el proyecto. La familia extendida Spartan-3A es la que se utiliza en el diseño de la PHR, que se distingue en la comparativa entre costo y recursos de hardware. Las Spartan-3A, permiten una gran variedad de modos de configuración en contraste con la familia Spartan-3. Por otro lado, no es necesaria una gran capacidad de procesamiento que justifique la inclusión de un DSP, debido al perfil del usuario de la plataforma que se desarrolla. Algunas de las características más relevantes de esta familia de FPGA son,

- $Vin_{Máx}$: 4,6V. Compatible con fuentes de 3.3V +/- 10%.
- Señales estándar: LVCMOS, LVTTL, HSTL y SSTL.
- Driver de salida hasta 24mA.
- Tasa de transferencia 622Mb/s.

[2]CUDAR – Centro Universitario de Desarrollo en Automoción y Robótica.

- 18x18 multiplicadores dedicados con *pipeline* opcional.
- Puerto programación/debug JTAG IEEE 1149.1/1532.
- Digital Clock Manager (DCMs)
  - Rango de frecuencia 5Mhz hasta 300Mhz.
- Ocho global clock.
- Interfaz de configuración para PROMs estándar.
  - PROM flash SPI, bajo costo.
  - PROM flash NOR paralelo x8 o x8/x16.
- Reconfiguración automática Multi-boot entre dos archivos.
- *Package* de bajo costo QFP y BGA.

La arquitectura de la Spartan-3 consiste de cinco elementos fundamentales funcionales programables:

Configurable Logic Block (CLBs) contienen flexibles *Look-Up Tables* (LUTs) que implentan elementos lógicos usados como flip-flop o *latch*.

Input/Output Blocks (IOBs) controla el flujo de datos entre los pines de I/O y la lógica interna del dispositivo. Los IOBs soportan flujo de datos bidireccionales además de operaciones 3-*state*.

Block RAM provee almacenamiento de datos en la forma de bloques *dual-port* de 18Kbit.

Multiplier Blocks toma dos números binarios de 18bit como entrada y calcula el producto. La línea Spartan-3A DSP incluye bloques especiales DSP.

Digital Clock Manager (DCM) Block proporciona auto-calibración, retardos, multiplicadores, divisores, y señales de clock de cambio de fase (*phase-shifting*).

La generación de FPGAs Spartan-3 son programadas por la carga de datos de configuración en dispositivos sólidos, reprogramables, *static CMOS configuration latches* (CCLs) que en conjunto controla todo los elementos funcionales y designan las fuentes.

El dato de configuración de las FPGA es almacenado en dispositivos externos como una PROM o algún dispositivo no-volátil.

Todas las señales que entran y salen de la FPGA deben pasar a través de los recursos I/O, conocidos como IOBs. Ya que las FPGAs son usadas en muchas aplicaciones complejas, estos dispositivos deben soportan un incremento variable de I/O. La revolucionaría *SelectIO* (patentado por xilinx), que contiene la Spartan-3 reúne esta necesidad proporcionando una enorme *configurabilidad*, alto *performance* de recursos adecuados para aplicaciones como son memorias de alta velocidad y interfaces de placas complejas programables.

La generación de FPGA Spartan-3 simplifica diseños de alto-*performance* ofreciendo un seleccionable diseño I/O estándar para entrada y salida. Más de 20 diferentes estándares son soportados en cada familia, con diferentes especificaciones de corriente, voltaje, I/O baffering, y terminaciones técnicas. Como un resultado, la generación de FPGA Spartan-3 puede ser usada para transformadas integrales discreta y drive direccional en muchas placas avanzadas, buses, y memorias. Directamente proporciona el interfaz estándar necesario no solo para eliminar el costo externo de traslación, sino también mejora significativamente la velocidad de *chip-to-chip* y reduce el consumo de potencia.

La FPGA que se utiliza pertenece a la familia Spartan-3 de Xilinx Inc. Esta familia a la vez se clasifican en

- Familia Spartan-3A extendida (bajo costo):
  - Spartan-3A
    - Ideal para uso de interfaz entre dispositivos.
  - Spartan-3A DSP
    - Mayor densidad de recursos en comparación que la familia Spartan-3A
    - Dispone de un dispositivo DSP (DSP48A)
  - Spartan-3AN
    - Dispositivos no volátiles
    - Ideal para aplicaciones con restricciones de espacio
- Familia Spartan-3E
- Familia Spartan-3

Tabla 5.2 Característica de la familia Spartan-3A

Devices	**System Gates**	**Block RAM bits**	**Dedicated Multipliers**	**Maximum User I/O**
XC3S50A	50K	54K	3	144
XC3S200A	**200K**	**288K**	**16**	**248**
XC3S400A	400K	360K	20	311
XC3S700A	700K	360K	20	372
XC3S1400A	1400K	576K	32	502

Altera, Atmel y otros fabricantes de FPGAs también presentan familias similares a las Spartan-3. Aquí se optó por Xilinx Inc. debido a la experiencia en software/hardware con que cuenta el Centro de Investigación[3] donde se desarrolla el proyecto. La familia extendida Spartan-3A es la que se utiliza en el diseño de la PHR, que se distingue en la comparativa entre costo y recursos de hardware. Las Spartan-3A [17], permiten una gran variedad de modos de configuración en contraste con la familia Spartan-3. Por otro lado, no es necesaria una gran capacidad de procesamiento que justifique la inclusión de un DSP, debido al perfil del usuario de la plataforma que se desarrolla. Las principales características de las FPGAs Spartan-3A se describen en la Tabla 5.2.

El dispositivo seleccionado, como se puede ver en la Tabla 5.2, es el XC3S200A. Éste cuenta con una gran densidad de recursos de hardware (200K compuertas lógicas) a la vez que se puede encontrar en un encapsulado de pequeñas dimensiones (VQ100) que facilita el diseño del PCB (Printed Board Circuit). En este encapsulado se puede contar con 68 puertos de entrada/salida (I/O) para ser utilizados externamente a diferentes tecnologías programables (LVTTL, LVCMOS33/25/18, entre otros). El perfil del diseño de la PHR no requiere de una gran cantidad de puertos de I/O debido a las aplicaciones para las que se lo diseña.

5.3.2 Memoria de configuración

La tecnología utilizada en las FPGAs Spartan-3A requieren de una memoria externa que configure al dispositivo ya que es volátil. Esta familia permite la utilización de varios tipos de memorias como modos de configuración para embeber el diseño digital

[3]CUDAR – Centro Universitario de Desarrollo en Automoción y Robótica.

Tabla 5.3 Tipo de memoria para la familia Spartan-3A

Devices	**Configuration Bits**	**ISP PROM Solution**
XC3S50A	437,312	XCF01S
XC3S200A	**1,196,128**	**XCF02S**
XC3S400A	1,886,560	XCF02S
XC3S700A	2,732,640	XCF04S
XC3S1400A	4,755,296	XCF08P

en la FPGA. Xilinx comercializa memorias Flash PROM [18] para todas sus familias de FPGA. Hay una relación directa entre la capacidad lógica de una FPGA con el tamaño de la memoria de configuración, en la Tabla 5.3 se puede apreciar esta relación para el caso de la familia Spartan-3A.

Tanto la FPGA como la memoria de configuración Flash PROM se encuentran conectadas en cadena a través de una interfaz JTAG Boundary-Scan (IEEE 1149.1) que Xilinx Inc. implementa en sus dispositivos FPGAs, CPLDs y memorias Flash PROM para transferir los diseños sintetizados.

5.4 Descripción de las placas

La *Plataforma de Hardware Reconfigurable* (PHR) consiste fundamentalmente en tres módulos de soporte físico. El módulo principal es la *placa PHR* donde se encuentran el chip FPGA, relojes, interfaces de entradas y salidas, periféricos (tales como LEDs, botones, llaves DIP, Displays de siete segmentos), etc [19].

Además tiene conectores especiales para otros dos módulos sin los cuales la placa principal carece de funcionalidad. Uno de ellos se emplea para la regulación de las tensiones que alimentan al resto de los dispositivos. Se trata de la denominada *placa S3Power*. El otro módulo es una interfaz de comunicaciones necesaria para configurar la FPGA o escribir la memoria PROM de configuración y se dispone en la *placa OOCD Link*.

La conexión de las placas auxiliares a la principal se ilustra en la Fig. 5.1. La placa S3Power se acopla con los conectores que se indican con los números 8 y 16 en la Fig. 5.3, mientras que la OOCD Link se une a la principal (mediante un adaptador) con los pines demarcados con 7. Esta última, a diferencia de la S3Power, no precisa

estar siempre conectada, pero si cada ves que se desee configurar la FPGA o grabar su memoria PROM.

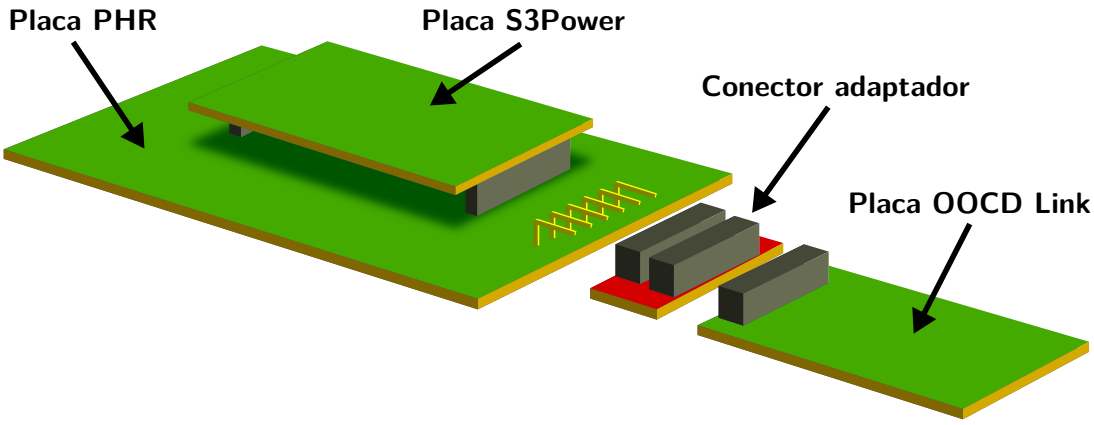


Figura 5.1 Conexionado de la placa PHR con las placas auxiliares.

5.5 Diagrama de bloques del hardware

Un esquema algo mas detallado del funcionamiento de la plataforma se ilustra en la Fig. 5.2. En líneas punteadas se demarcan las distintas plaquetas, y en línea continua se muestran los bloques que componen a cada una. Cuando hay funciones desempeñadas por un chip en particular los rectángulos se destacan con un fondo gris.

La función de la placa S3Power la realiza principalmente el *chip TPS75003* el cuál tiene un regulador lineal y controladores para dos fuentes conmutadas, lo cual permite suministrar energía regulada con tres valores de tensión y distintas características de arranque[4]. Los voltajes utilizados por la FPGA son de 1.2V, 2.5V y 3.3V.

En la placa PHR, la FPGA se conecta a los distintos periféricos que se distinguen en la Fig. 5.2 con un fondo amarillo. El conjunto de periféricos está compuesto por *LEDs*, *Llaves DIP*, *Botones*, el *display de siete segmentos cuádruple* y el *puerto serie*[5].

Un recurso que puede facilitar el diseño de los proyectos es la señal de reloj. La placa PHR pone a disposición del usuario diversas señales de clock que van desde una frecuencia mínima de 977 Hz hasta una frecuencia máxima de 50MHz[6].

[4]Una explicación mas detallada de *S3Power* puede consultarse en el capítulo 5.9.
[5]Mas información sobre *periféricos* en capítulo 5.7, página 59.
[6]Mas información sobre *relojes* en capítulo 5.7, página 58.

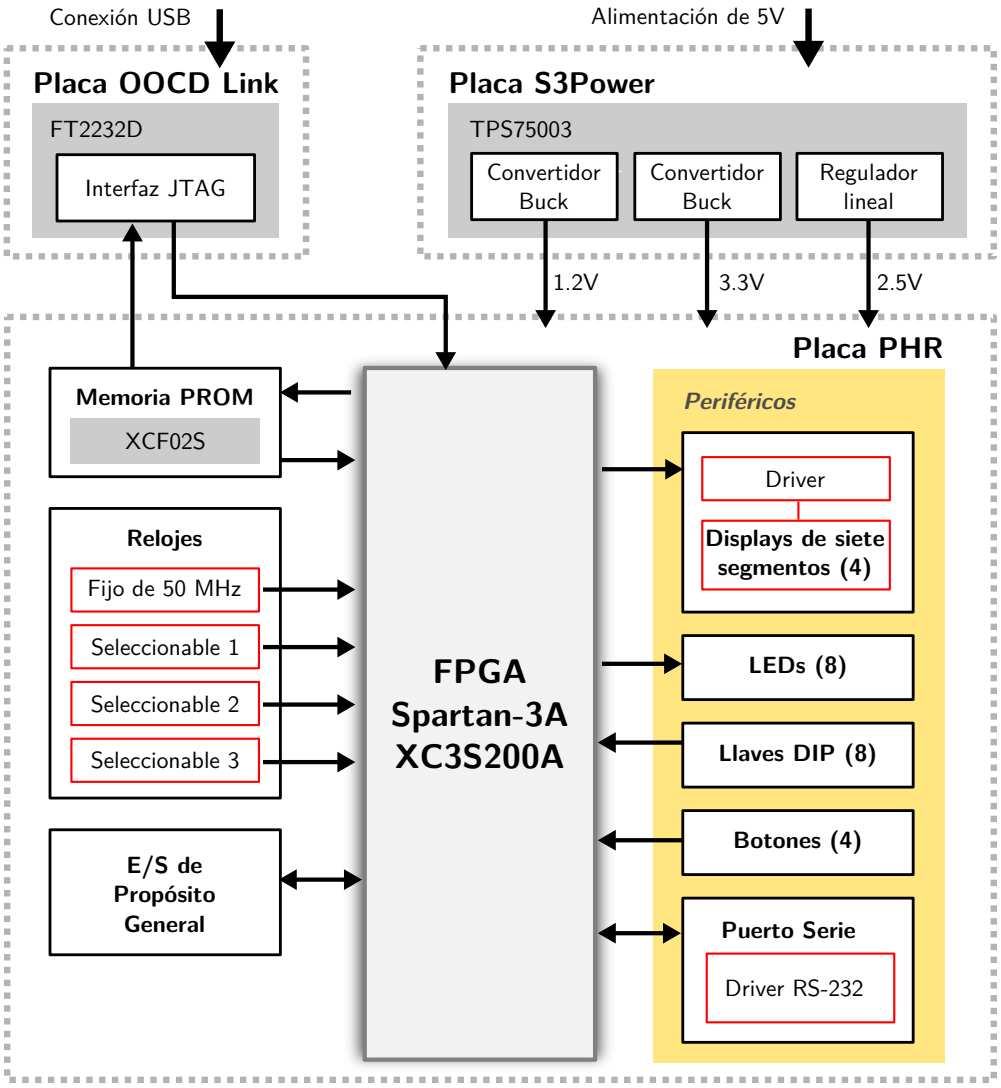


Figura 5.2 Diagrama de bloques de la plataforma.

Para que la placa sea apta además para la realización de prototipos, se incluyeron dos conectores, con los cuales se el usuario tine acceso directo a los pines de la FPGA[7].

La placa OOCD Link incluye el *chip FT2232D* que establece una interfaz JTAG controlable mediante una conexión USB. Un anillo JTAG se establece con la FPGA y la memoria PROM, esta ultima implementada con el *chip XCF02S*[8].

[7]Mas información sobre *entradas y salidas de propósito general* en capítulo 5.7, página 64.

[8]Mas información sobre la *placa OOCD Link* en capítulo 5.8. Sobre el proceso de configuración refiérase a la sección *Configuración de la FPGA* del capítulo 5.7, página 55.

5.6 Componentes de la placa principal

En la Fig. 5.3 se tiene la vista superior de la placa PHR con sus principales componentes demarcados. Según la numeración, estos componentes son:

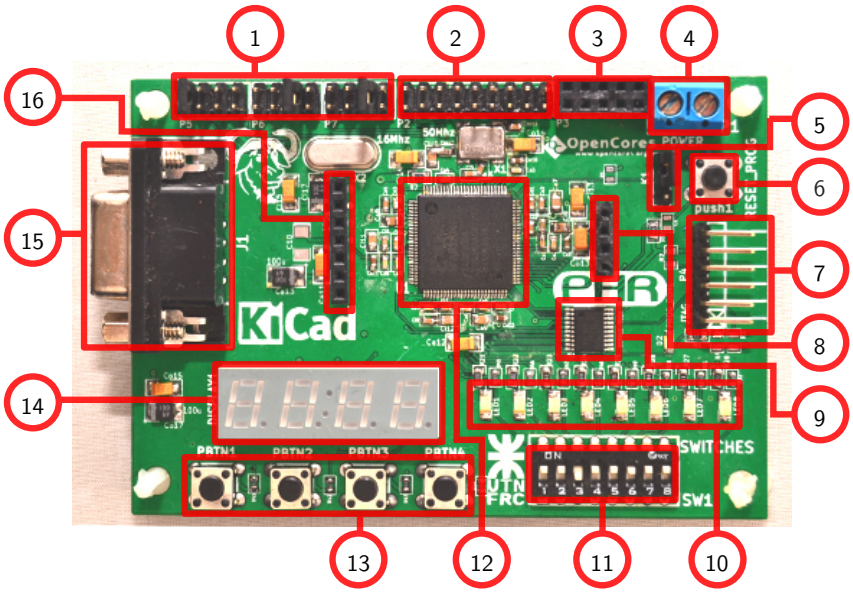


Figura 5.3 Componentes de la placa PHR.

1. Selector de *clocks*.
2. Conector de módulos externos macho.
3. Conector de módulos externos hembra.
4. Entrada de voltaje de alimentación (5V).
5. Selector de modo de configuración.
6. Boton de RESET.
7. Conector de la plaqueta *OOCDLink*.
8. Conector de alimentación de la placa *S3Power*.
9. Memoria PROM.
10. LEDs.

11. Llaves DIP.
12. Chip FPGA XC3S200A.
13. Botones pulsadores.
14. Display de 7 segmentos cuádruple.
15. Conector para puerto serie.
16. Conector de alimentación para la placa PHR desde S3Power.

5.7 PHR

5.7.1 El chip FPGA

Características principales

- Número de compuertas: 200K
- Celdas lógicas equivalentes: 4032
- CLBs: 448 (distribuidos en 32 filas y 16 columnas)
- Bits de RAM distribuida: 28K
- Bits de Bloques de RAM: 288K
- Multiplicadores dedicados: 16
- DCMs: 4
- Máximo número de E/S = 248
- E/S pares diferenciales máximo: 112

Descripción de la arquitectura

La familia Spartan-3A tiene cinco bloques fundamentales a tener en cuenta respecto de la arquitectura:

Bloques Lógicos Configurables (CLBs) contienen *Look-Up Tables* (LUTs) que implementan funciones lógicas y además sirven como elementos de almacenamiento.

Bloques de Entrada/Salida (IOBs) controlan el flujo de datos entre los pines de E/S y la lógica interna del dispositivo. Los IOBs soportan datos en forma bidireccional además de la operación 3-state.

Bloque de RAM sirve como forma de almacenamiento.

Bloques multiplicadores aceptan como entrada dos números de 18 bits cada uno y calcula el producto entre ambos.

Bloque DCM (*Digital Clock Manager*) tiene la capacidad para distribuir, retardar, multiplicar, dividir y desplazar en fase las señales de clock.

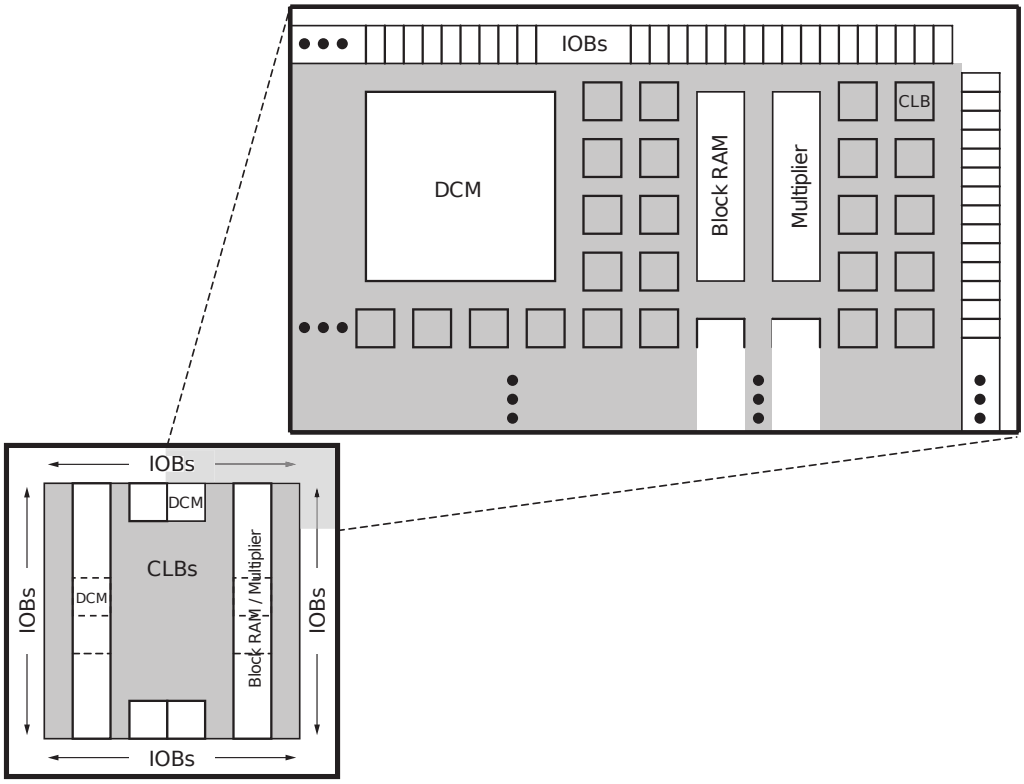


Figura 5.4 Bloques fundamentales de la FPGA.

Para el caso del chip XC3S200A, la Fig. 5.4 muestra la organización de estos bloques fundamentales dentro del dispositivo. Los IOBs se ubican alrededor de la matriz de CLBs. Los bloques de memoria RAM se encuentran en dos columnas, en cada una se disponen varios bloques de RAM de 18 Kbit asociados con un multiplicador dedicado. Dos DCM se ubican en el centro hacia arriba y otros dos en el centro hacia abajo.,

Capacidades de E/S

La familia Spartan-3A soporta varios estándares para sus entradas y salidas que deberán ser configurados apropiadamente para el uso específico.

- Terminales simples
  - TTL de 3.3V (LVTTL).
  - CMOS de bajo voltaje (LVCMOS) en tensiones de 3.3V, 2.5V, 1.8V, 1.5V o 1.2V.
  - PCI de 3.3V a frecuencias de 33MHz o 66MHz.
  - HSTL I, II y III a 1.5V y 1.8V (comúnmente usado en memorias).
  - SSTL I y II a 1.8V, 2.5V y 3.3V (comúnmente usado en memorias).
- Terminales diferenciales
  - Entradas/Salidas LVDS, mini-LVDS, RSDS y PPDS a 2.5V o 3.3V.
  - Bus LVDS a 2.5V.
  - TMDS a 3.3V.
  - HSTL y SSTL diferenciales.
  - Entradas LVPECL a 2.5V y 3.3V.

Requerimientos de alimentación

El chip XC3S200A tiene varias entradas de alimentación que se describen de manera sucinta en la Tabla 5.4. La FPGA cuenta con un circuito especializado de *Power-On Reset* (POR) que controla tres tensiones de alimentación (VCCINT, VCCAUX y VCCO2) y mantiene al chip en estado de reset hasta que se alcanzan los niveles seguros de trabajo para proseguir con la carga del sistema.

A diferencia de otras FPGAs, la XC3S200A no tiene requerimientos respecto de la secuencia en se deben activar las fuentes de alimentación, pero si respecto de la pendiente de arranque. Los tiempos de pendiente recomendados se muestran en la Tabla 5.5.

Para mas información referida al sistema de alimentación ver el capitulo 5.9 en la pág. 67.

Entrada	Descripción	Tensión nominal
VCCINT	Es la tensión de alimentación del núcleo interno. Alimenta las funciones lógicas internas como los CLBs (*Bloques Lógicos Configurables*) y los bloques de RAM.	1.2V
VCCAUX	Fuente de tensión auxiliar. Alimenta elementos tales como los DCMs (*Digital Clock Managers*), drivers diferenciales, pines de configuración dedicados y la interfaz JTAG.	2.5V o 3.3V
VCCO0	Alimenta los buffers de salida del Banco de E/S número 0.	Seleccionable entre 3.3V, 3.0V, 2.5V, 1.8V, 1.5V y 1.2V.
VCCO1	Alimenta los buffers de salida del Banco de E/S número 1.	Seleccionable entre 3.3V, 3.0V, 2.5V, 1.8V, 1.5V y 1.2V.
VCCO2	Alimenta los buffers de salida del Banco de E/S número 2.	Seleccionable entre 3.3V, 3.0V, 2.5V, 1.8V, 1.5V y 1.2V.
VCCO3	Alimenta los buffers de salida del Banco de E/S número 3.	Seleccionable entre 3.3V, 3.0V, 2.5V, 1.8V, 1.5V y 1.2V.

Tabla 5.4 Voltajes de alimentación para la familia Spartan-3A.

5.7.2 Configuración de la FPGA

La FPGA al inicializarse no contiene dato alguno y para que trabaje como lo desea el usuario debe pasar por el proceso de *configuración*. Los datos se cargan desde el exterior en *latches* de configuración CMOS (CCLs según las iniciales en inglés) y usando alguno de los siguientes modos aplicables a la familia Spartan-3A [20]:

- *Master Serial* desde una memoria PROM Flash de Xilinx.
- *Serial Peripheral Interface* (SPI) desde una memoria Flash SPI.
- *Byte Peripheral Interface* (BPI) desde una memoria NOR Flash.
- *Slave Serial*, típicamente cargada desde un procesador.
- *Slave Parallel*, típicamente cargada desde un procesador.

Símbolo	**Descripción**	**Min**	**Max**
VCCINTR	Rampa desde GND a VCCINT	0.2 ms	100 ms
VCCAUXR	Rampa desde GND a VCCAUX	0.2 ms	100 ms
VCCO2R	Rampa desde GND a VCCO del Banco 2	0.2 ms	100 ms

Tabla 5.5 Tiempos de subida para las rampas al encender las fuentes de alimentación.

- *Boundary Scan* (JTAG), típicamente cargada desde un procesador.

La elección de cada uno de los modos se hace mediante tres pines de la FPGA a los que se hace referencia con M[2:0]. La Tabla 5.6 muestra cuales son los valores lógicos de los modos aplicables para ésta familia de FPGA, y se resaltan con color aquellos a los que se recurre en la placa PHR.

Pines M[2:0]	**Modo**
<0:0:0>	Modo *Master Serial*
<0:0:1>	Modo *Master SPI*
<0:1:0>	*BPI Up*
<0:1:1>	Reservado
<1:0:0>	Reservado
<1:0:1>	Modo JTAG
<1:1:0>	Modo *Slave Parallel*
<1:1:1>	Modo *Slave Serial*

Tabla 5.6 Seteo de los modos de configuración para la familia Spartan-3A. Se resaltan con amarillo los usados en la placa PHR.

En la placa PHR los modos utilizados son el *JTAG* (a través de la placa OOCDLink) y el *Master Serial* (desde la memoria PROM XCF02S). Cuando se elije el primer método se configura el chip con una computadora que debe correr una aplicación al efecto. Asimismo, para aplicar el segundo método, hay que usar la placa OOCDLink para programar la PROM al menos una vez. De ahí en mas, con cada ciclo de arranque de la FPGA, ésta tendrá la posibilidad de trabajar en forma independiente de la computadora cargando los datos pre-grabados en la memoria.

La Fig. 5.5 muestra el diagrama de como se ve la elección del modo de configuración a un nivel físico. Sin puente alguno se leen 3.3V en los pines M0 y M2, mientras que se lee 0V en M1 (la FPGA recibirá los datos desde JTAG). Al colocar un jumper entre los pines 1 y 2 se tira la tensión de M0 y M2 al potencial de masa (la FPGA intentará cargar desde la PROM).

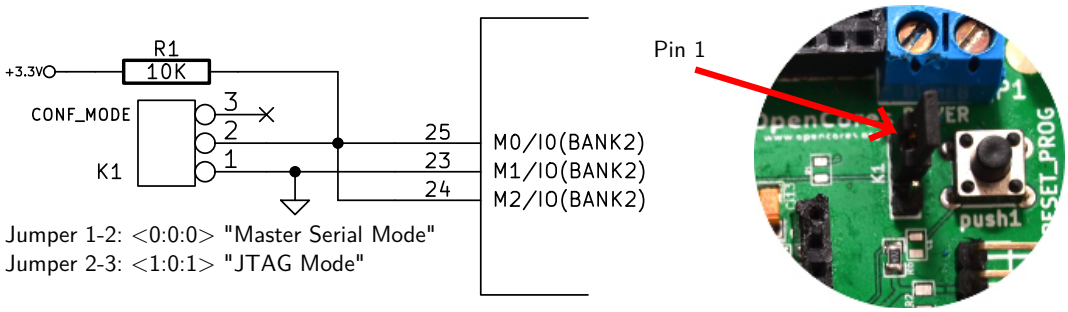


Figura 5.5 Modos de configuración.

A nivel de usuario la selección se trata simplemente de cambiar la posición del jumper indicado con 5 en la Fig. 5.3. Una ilustración de las dos posibles vías que toman los datos para alcanzar la FPGA se muestra en la Fig. 5.6.

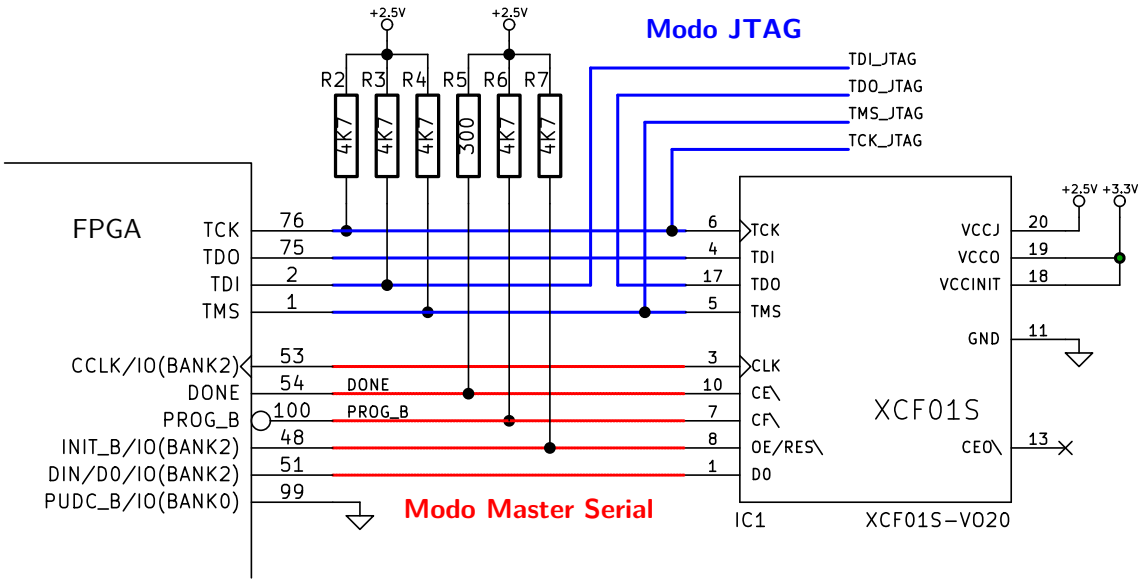


Figura 5.6 Modos de configuración.

5.7.3 Fuentes de *clock*

La placa PHR provee a la FPGA de cuatro fuentes de reloj. Uno de los relojes (el más rápido) tiene una frecuencia de oscilación fija de 50 MHz. El resto tienen frecuencias seleccionables por usuario. En la Fig. 5.3 se indican con el numero 1 los jumpers con los cuales se eligen las frecuencias para estos relojes.

Los pines de la FPGA a los cuales se asignan cada clock se muestran en la Tabla 5.7. Estos son pines de *Global Clock* que están especialmente diseñados para tratar señales de alta frecuencia. Proveen una capacidad asociada muy baja y un retardo uniforme para cada bloque dentro del chip.

Reloj	Fijo (50 MHz)	Seleccionable 1	Seleccionable 2	Seleccionable 3
Pin	43	44	41	40

Tabla 5.7 Pines para los relojes.

El reloj de 50 MHz

Esta frecuencia se genera con el dispositivo ACOL-50MHZ-EK, que tiene un oscilador a cristal y encuentra aplicaciones en chips digitales y microprocesadores. Se alimenta con un bajo nivel de tensión (3.3V) y su salida es compatible con HCMOS y TTL.

Relojes seleccionables

El conjunto de relojes se basa en un cristal y el chip contador MC74HC4060A. Una frecuencia principal de 16 MHz es dividida en dos sucesivamente por el contador para obtener todas las frecuencias seleccionables.

Los pines para selección de los relojes se muestran en la Fig. 5.7. En los primeros dos relojes se puede elegir una de entre cuatro frecuencias mientras que para el tercer reloj se puede elegir una de entre tres frecuencias.

El conexionado de los jumpers para los clocks se muestra junto con las posibles frecuencias seleccionables en la Fig. 5.8.

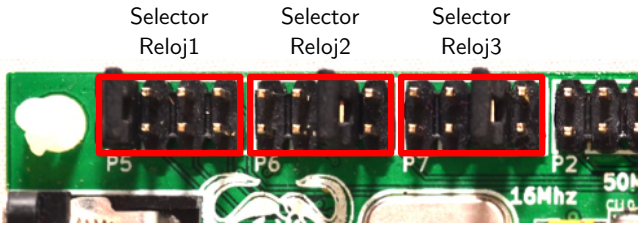


Figura 5.7 Selectores de los relojes en la placa.

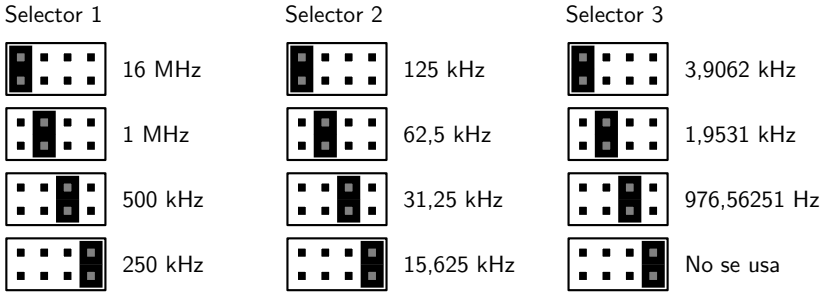


Figura 5.8 Disposición de los jumpers para la configuración de los relojes.

5.7.4 Periféricos

LEDs

En la placa se encuentran ocho LEDs de montaje superficial indicados con el numero 10 en la Fig. 5.3. Son etiquetados desde LED1 a LED8 y su relación con los pines de la FPGA se muestra en la Tabla 5.8.

Periférico	LED1	LED2	LED3	LED4	LED5	LED6	LED7	LED8
Pin	84	86	89	93	98	3	5	7

Tabla 5.8 Correspondencia entre los pines de la FPGA y los LEDs (periféricos).

Los cátodos de cada LED se conectan a potencial cero y los ánodos se conectan a los pines respectivos de la FPGA mediante un resistencia de 330 Ω. Para encender un determinado LED basta con poner en alto la señal de control.

Pulsadores (*Tact switches*)

Están disponibles cuatro botones pulsadores como los esquematizados en la Fig. 5.9 y son identificados con el numero 13 en la Fig. 5.3. Los mismos son etiquetados como

PBTN1, PBTN2, PBTN3 y PBTN4. Los pines de la FPGA relacionados con estos periféricos se identifican en la Tabla 5.9. El esquemático detallado del circuito puede encontrarse en el Apéndice B.

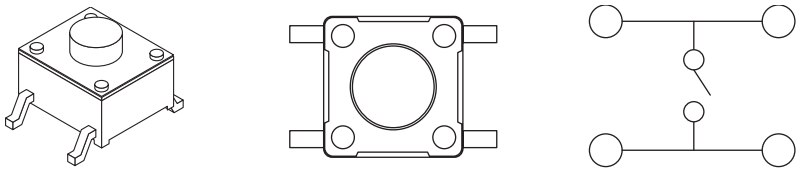


Figura 5.9 *Tact switches.*

Periférico	PBTN1	PBTN2	PBTN3	PBTN4
Pin	77	78	82	83

Tabla 5.9 Correspondencia entre los pines de la FPGA y los botones.

Cuando se presiona alguno de los botones se genera un valor lógico alto en el pin asociado de la FPGA. No hay circuito antirrebote y esto debe ser tenido en cuenta al momento de escribir el código que luego vaya a cargarse en el dispositivo.

Llaves DIP

Alternativamente a los pulsadores se puede optar como periféricos de entrada a unas llaves DIP como se muestran en la Fig. 5.10. La ubicación de las llaves en la placa PHR se muestra con el índice numero 11 en la Fig. 5.3. El circuito de estas llaves puede consultarse en el Apéndice B y los pines de la FPGA que los controlan se revelan en la Tabla 5.10.

Periférico	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
Pin	85	88	90	94	97	4	6	9

Tabla 5.10 Correspondencia entre los pines de la FPGA y las llaves.

Cuando una llave se coloca en la posición de *encendido*, el pin de la FPGA correspondiente se pone a un valor lógico *alto*. En contraposición, si la llave se coloca en la posición *apagado*, la FPGA leerá un valor lógico *bajo*. Al igual que con el caso de los

botones pulsadores, no se provee un circuito antirrebote, y debe ser tenido en cuenta en el diseño del sistema.

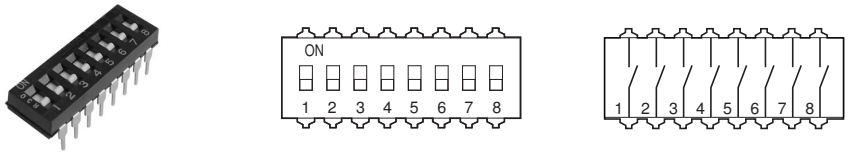


Figura 5.10 *DIP switches*.

Displays de 7 segmentos

La placa PHR cuenta con un display de siete segmentos cuádruple de *ánodo común* indicado con el índice 14 en la Fig. 5.3. El circuito de conexión entre la FPGA y el display se muestra en la Fig. 5.11 y se resalta la denominación alfabética para los segmentos de cada display.

Esta misma figura además muestra como ejemplo, el estado de los pines de la FPGA para indicar el numero 3 en la posición 2. Al tener esta configuración, cada LED encenderá con un *nivel bajo* en el pin correspondiente al segmento pero además necesitará que el ánodo del carácter particular esté energizado. Este ultimo también es activo por bajo (*active low*).

Para dar el efecto deseado de representar cuatro caracteres distintos a la vez, se recurre a la técnica de multiplexación en el dominio del tiempo. La técnica consiste en mostrar uno a uno y cíclicamente cada carácter a una frecuencia lo suficientemente alta para que el ojo humano persiva una imagen completa. Un diagrama temporal de las señales se muestra en la Fig. 5.12.

Si bien el método requiere algo mas de complejidad que la conexión directa a cada segmento de cada display, reduce el numero de pines necesarios de $8 \times 4 = 32$ a $8 + 4 = 12$ lo cuál representa un significativo ahorro en recursos de hardware.

La Tabla 5.11 muestra los pines de conexión de la FPGA a las distintas entradas del periférico. La Fig 5.13 muestra las representaciones de los caracteres comunes en los displays de siete segmentos. Además de los dígitos, se pueden utilizar los caracteres desde la A a la F para representar números en notación hexadecimal.

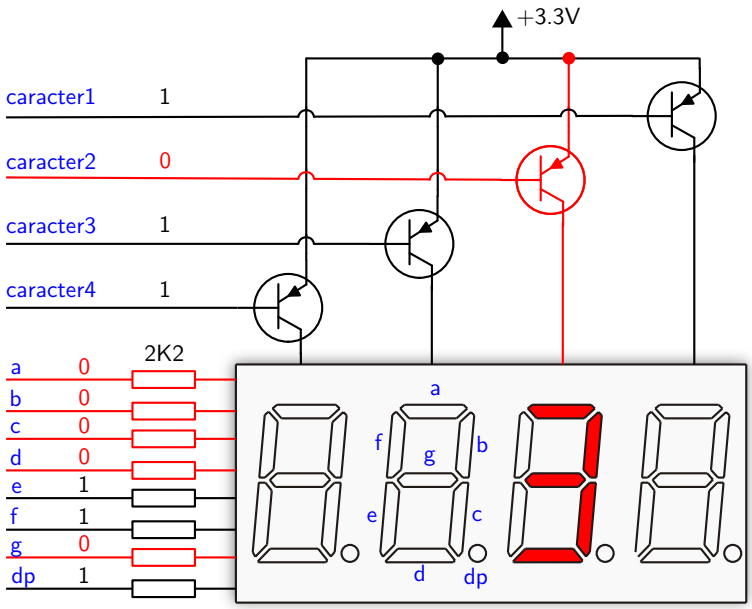


Figura 5.11 Conexionado del display de siete segmentos cuádruple.

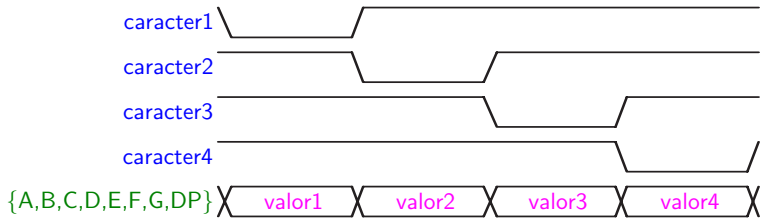


Figura 5.12 Diagrama temporal de la multiplexación.

Periférico	Caracter1	Caracter2	Caracter3	Caracter4
Pin	59	57	61	60

Segmento	A	B	C	D	E	F	G	DP
Pin	65	64	72	70	68	62	73	71

Tabla 5.11 Conexionado del diplay de 7 segmentos cuádruple a la FPGA.

Puerto serie

La placa PHR dispone de un puerto serial RS-232. El conector DB9 hembra/macho se señala con el numero 15 en la Fig. 5.3. La placa representa un *Data Communications*

Figura 5.13 Representación de caracteres comunes en los displays de siete segmentos.

Equipment (DCE) y se puede conectar a una computadora personal con un cable serial derecho.

La Fig. 5.14 muestra el circuito de la interfaz entre la FPGA y el conector DB9. Un chip (ST3232) permite una alimentación de entre 3V y 5.5V y utiliza un conjunto de capacitores para alcanzar los niveles de tensión RS-232, garantizando tasas de transferencia de hasta 250 kbps.

El control de flujo por hardware está deshabilitado. Las lineas de DCD, DTR y DSR se conectan entre sí. Analogamente para las señales RTS y CTS.

Los pines de la FPGA que controlan las señales seriales se muestran en la Tabla 5.12.

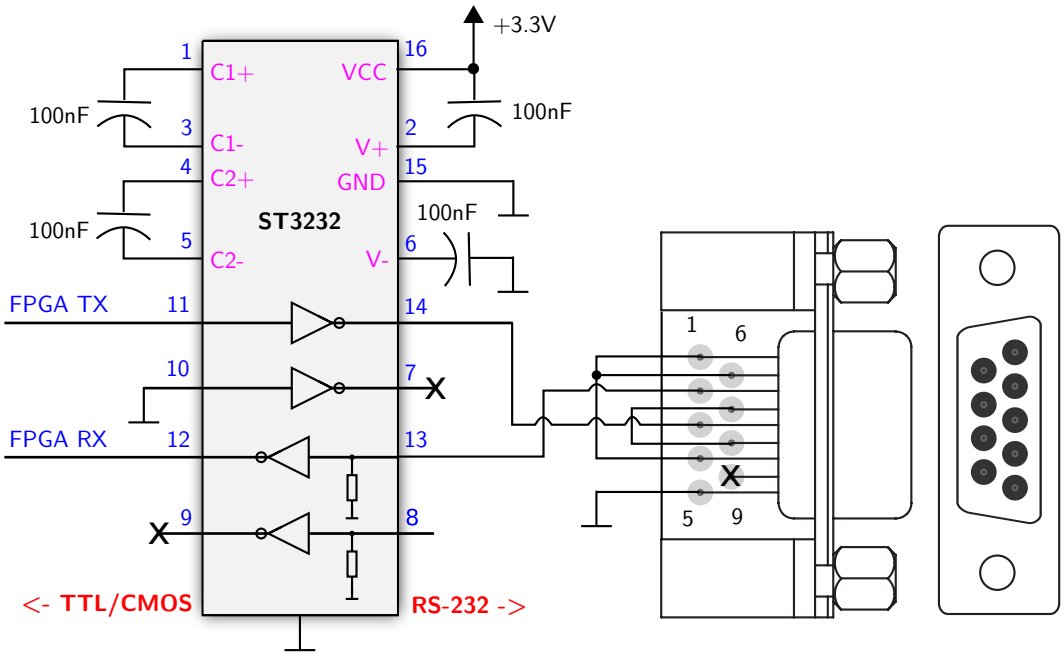


Figura 5.14 Circuito de la interfaz RS-232.

Señal serial	RX	TX
Pin en la FPGA	52	56

Tabla 5.12 Correspondencia entre los pines de la FPGA y el puerto serie RS-232.

5.7.5 Entradas y salidas de propósito general

Para que el usuario realice prototipos, use placas de expansión de terceros o diseñe sus propias placas de expansión, se proveen dos conectores, uno macho y otro hembra, que pueden reconocerse respectivamente con los números 2 y 3 en la Fig. 5.3. Una imagen ampliada se muestra en la Fig. 5.15 indicando además la numeración de los pines. La mayoría de estos pines están conectados directamente al chip FPGA tal como lo muestra la Tabla 5.13. También se proveen pines de alimentación de 3.3V y GND.

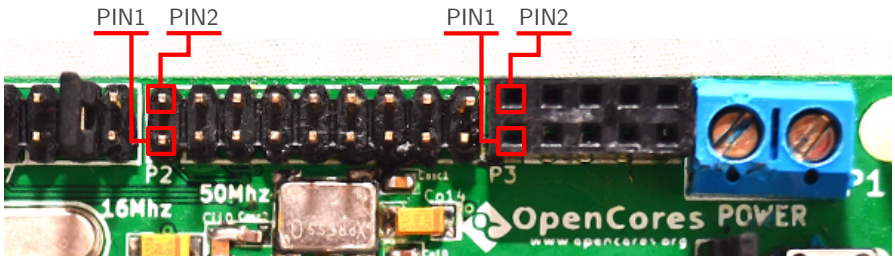


Figura 5.15 Conectores para entradas y salidas de propósito general.

5.8 OOCDLink

Esta placa es la interfaz que permite la comunicación entre una computadora y la placa PHR. Su característica modular, o de circuito separado de la placa PHR principal, hace que su utilización no quede restringida a la FPGA y posibilita la interacción con los multiples dispositivos que soportan JTAG.

La idea original de esta placa no nace en este proyecto sino que es parte de los pensamientos de Joern Kaipf quien publica su diseño en su sitio web[9].

La placa OOCDLink se muestra en la Fig. 5.16 donde también identifican los elementos principales que la constituyen. Un esquemático mas detallado se muestra en el Apéndice B.

[9]Diríjase a `http://www.oocdlink.com/`.

Conector macho					
Dir	**Conectado a**	**Pin**	**Pin**	**Conectado a**	**Dir**
E	FPGA Pin 39	1	2	FPGA Pin 50	E/S
E/S	FPGA Pin 37	3	4	FPGA Pin 49	E/S
E/S	FPGA Pin 36	5	6	FPGA Pin 46	E/S
E/S	FPGA Pin 35	7	8	FPGA Pin 34	E/S
E/S	FPGA Pin 33	9	10	FPGA Pin 32	E/S
E/S	FPGA Pin 31	11	12	FPGA Pin 30	E/S
E/S	FPGA Pin 29	13	14	+3.3V	
E/S	FPGA Pin 28	15	16	No conectado	
E/S	FPGA Pin 27	17	18	GND	

Conector hembra					
Dir	**Conectado a**	**Pin**	**Pin**	**Conectado a**	**Dir**
E	FPGA Pin 21	1	2	FPGA Pin 20	E/S
E/S	FPGA Pin 19	3	4	FPGA Pin 16	E/S
E/S	FPGA Pin 15	5	6	FPGA Pin 13	E/S
E/S	FPGA Pin 12	7	8	+3.3V	
E/S	FPGA Pin 10	9	10	GND	

Tabla 5.13 Conexión de los pines para las entradas/salidas de propósito general.

El dispositivo que cumple la función primordial dentro de la placa OOCDLink es el chip FT2232D [21]. Este es un conversor de USB a UART u otras interfaces seriales que dispone de dos canales de comunicación (*conversor dual*).

Si bien para el fin de configurar la FPGA es suficiente solo uno de los canales (configurado en modo JTAG), en la placa además se encuentra habilitado el segundo canal (configurado como UART) para que se use con propósitos generales. Los conectores para los canales JTAG y UART se indican en la Fig. 5.16 con los números 4 y 1 respectivamente.

El conector JTAG es el que debe usarse para el acoplamiento con la placa PHR mediante un adaptador que cambia la disposición mecánica de los pines.

La placa OOCDLink tiene varios LEDs indicadores. El LED numerado con 9 en

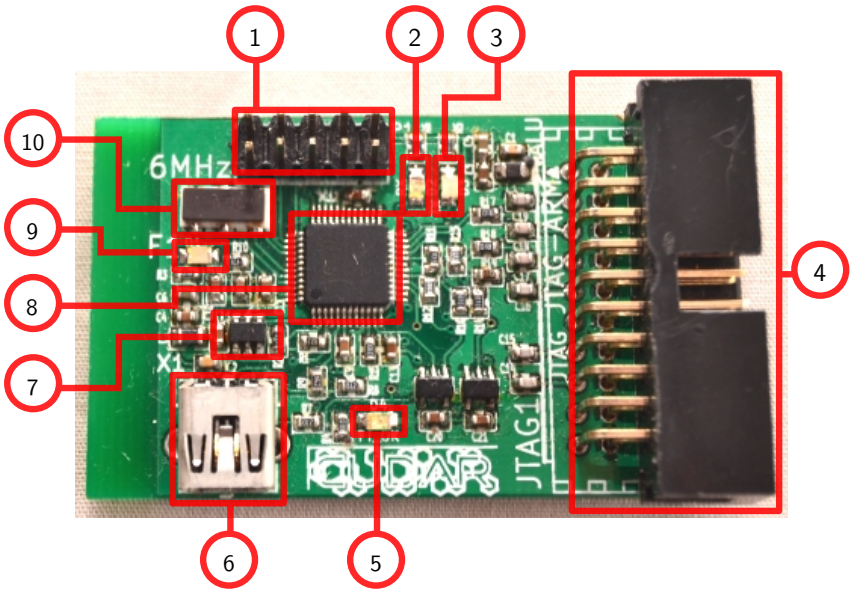


Figura 5.16 Componentes de la placa OOCDLink. 1) Conector UART, 2) LED RX (UART), 3) LED TX (UART), 4) Conector JTAG, 5) LED FT_OK, 6) Conector USB 2.0, 7) EEPROM, 8) FT2232D, 9) LED Alimentación, 10) Oscilador de 6 MHz.

la Fig. 5.16 ilumina cuando la placa está encendida. Aquellos demarcados con los números 2 y 3 (RX y TX) encienden cuando el chip tiene flujos de datos en la UART. El LED indicado con 5 (FT_OK) señala si hay un dispositivo JTAG activo y conectado a la placa OOCDLink.

5.8.1 El chip FT2232D

Algunas características del FT2232D que se pueden destacar son:

- Cumple con las especificaciones de USB 2.0 Full Speed (12 Mbits/sec).
- Simplifica la comunicación de USB con los protocolos de comunicaciones seriales JTAG, I2C y SPI.
- Tiene una tasa de transferencia de entre 300 y 3 MBaud.
- Desde el sistema operativo, la interfaz puede verse como un *puerto serie virtual* (necesita el driver que provee el fabricante sin costo adicional).
- También están disponibles librerías para facilitar el uso de JTAG, I2C y SPI (compatible con sistemas Windows y Linux).

La Fig. 5.17 muestra de manera esquemática el funcionamiento del chip aplicado a la placa OOCDLink.

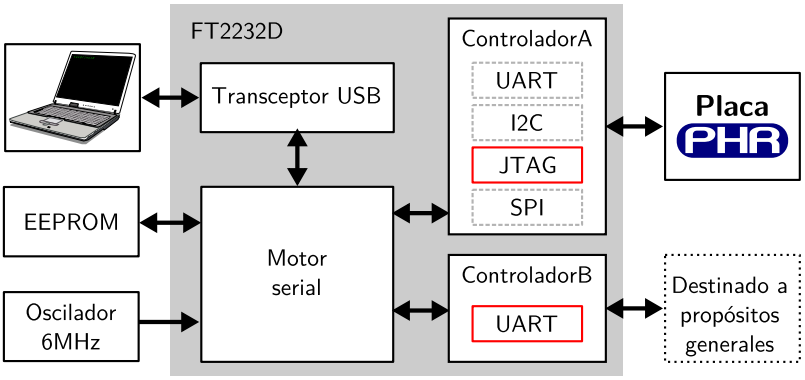


Figura 5.17 Empleo del chip FT2232D.

El chip necesita que por fuera se provean una memoria EEPROM (para almacenar los datos de configuración) y una señal de reloj de 6 MHz.

El flujo de datos (descripto desde la computadora del usuario) comienza por el bus de comunicaciones USB (función en la que interviene el *transceptor USB*), luego sigue a través del *Motor serial* (que podría considerarse el núcleo del sistema) y continua por alguno de los dos controladores capaces de manejar protocolos seriales.

El canal A es el que efectivamente permite implementar una interfaz JTAG cuando trabaja en el modo denominado *MPSSE* (*Multi-Protocol Synchronous Serial Engine*). El canal B tiene posibilidades de funcionamiento mas acotadas por lo que se usa como UART.

Para usar apropiadamente las interfaces se pueden usar los controladores de sistema operativo que el fabricante pone a disposición de los usuarios de Windows y Linux.

5.9 S3Power

Tal como se describió en el capitulo 5.7, el chip FPGA tiene requerimientos de tensión que deben satisfacerse para que funcione correctamente. Para cumplir con las especificaciones se utiliza la placa S3Power[10], que fue desarrollada por el *Instituto Nacional de Tecnología Industrial* (INTI) y que está disponible bajo licencia GNU [22].

[10]Puede consultar el paper *Módulo de alimentación para placas con dispositivos FPGA*, por Christian Huy y Diego Brengi, del *Instituto Nacional de Tecnología Industrial*.

Una imagen de la placa se muestra en la Fig. 5.18 en donde también se señalan sus principales elementos.

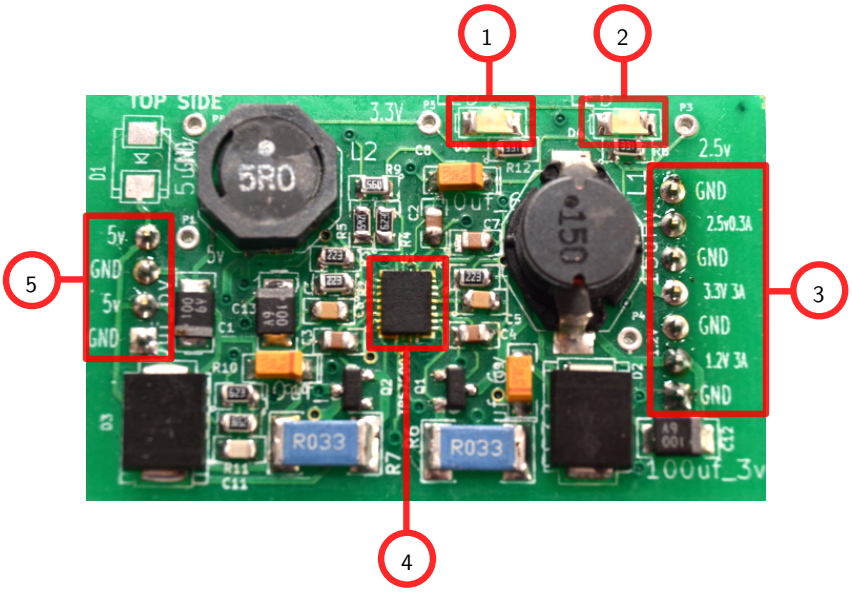


Figura 5.18 Componentes de la placa S3Power. Las distintas partes resaltadas representan: 1) LED de tensión VCCO, 2) LED de tensión VCCAUX, 3) Conector de salida de la placa, 4) Chip TPS75003, 5) Conector de tensión de entrada.

La placa se alimenta con una tensión de 5V y en la salida es capaz de proporcionar tres valores de tensión regulados:

- 1.2V y 2.5A para la lógica interna.
- 3.3V y 2.5A para los bancos de pines.
- 2.5V y 200mA para el módulo de comunicación JTAG.

El componente principal de la placa es el chip TPS75003 que no solo se encarga de regular las tensiones, sino que además asegura un arranque lo suficientemente suave para las FPGA actuales y sus predecesoras con requerimientos mas exigentes. Un diagrama temporal de la repuesta en el arranque se muestra en la Fig. 5.19.

5.9.1 El chip TPS75003

Este chip es un regulador de tensiones de *Texas Instruments* especialmente diseñado para servir de fuente para las familias de FPGA de Xilinx Spartan-3, Spartan-3E y Spartan-3L. Entre las características mas importantes se pueden mencionar:

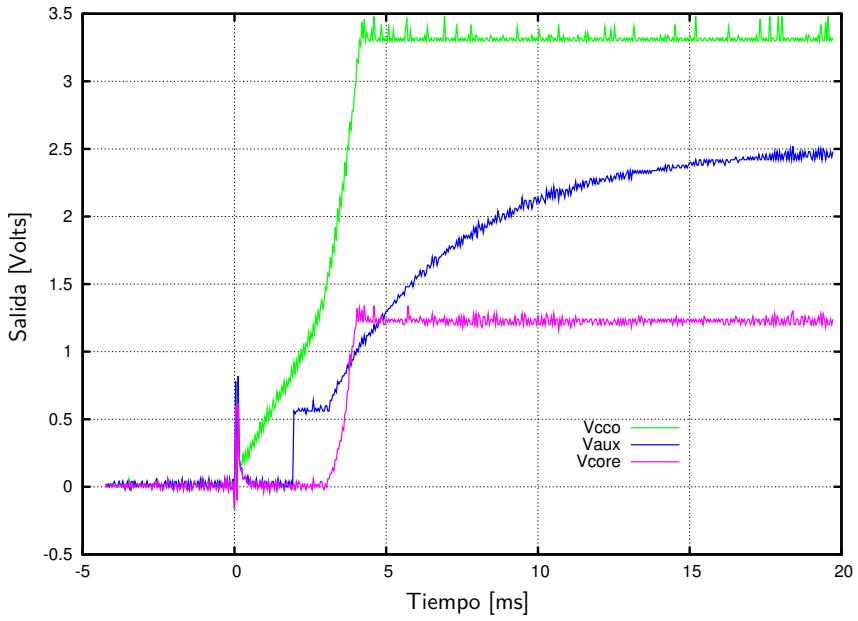


Figura 5.19 Tensiones de salida en el arranque.

- Posee tres reguladores de tensión: Dos tipo Buck de 3A y eficiencia del 95% y otro regulador lineal de 300 mA.
- Voltaje de entrada de entre 2.2V y 6.5 V.
- Arranque suave e independiente para cada regulador.
- Tensiones ajustables de 1.2 V a 6.5 V para los convertidores Buck y de 1.0 V a 6.5 V para el convertidor lineal.

5.10 Proceso para el diseño de Placas

El desarrollo de una placa sigue un proceso que se podría describir como la Figura 5.20. Cada una de estas etapas se encuentra documentada y ha sido utilizada para el presente informe. La interacción entre las diferentes etapas se representa por flechas. De aquí se puede obtener otra información sobre la metodología de trabajo, que es el sentido en que se comparte información. Por ejemplo, la etapa denominada *Especificaciones generales* proporciona información a las etapas de *Diseño del esquemático* y la etapa *Diseño de la placa electrónica*, y a su vez estas últimas ofrecen información a la primera para asegurar que determinados aspectos definidos inicialmente se estén cumpliendo. Se hace una breve descripción de las etapas.

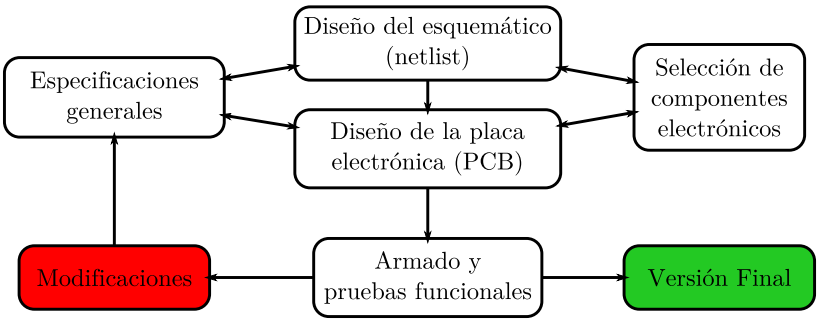


Figura 5.20 Proceso de diseño de las placas

Especificaciones generales En el diseño de un circuito electrónico se deben determinar las pautas y características que éste debe cumplir. Por ejemplo, tecnología a utilizar, dimensiones, costo, etc..

Diseño del esquemático Con la utilización de herramientas de *software*, en este caso kiCAD[11], se realiza el diseño del esquema de conexiones del circuito eléctrico. Aquí se representan los componentes electrónicos con símbolos y bloques en vez de sus verdaderas formas físicas, lo que facilita su interpretación.

Diseño de la placa electrónica Luego de obtener el circuito esquemático, se debe convertir dicho esquema en su equivalente real. Aquí sí se tiene un modelo de los dispositivos electrónicos reales (dimensiones y formas), con la ventaja que las conexiones eléctricas se encuentran ya definidas. Lo importante de esta etapa es definir la dimensión de la placa, como así también la disposición de cada componente electrónico.

Selección de componentes electrónicos En esta etapa se lleva un registro de los dispositivos electrónicos a utilizar. Para el caso de la generación del circuito esquemático, se determina que simbología utilizar en la representación de los componentes. Y en el caso de la generación de la placa, cada símbolo debe tener su representación física real. Gran parte de esta etapa se basa en documentación para generar la compra de los materiales necesarios.

Armado y pruebas funcionales El montaje de los componentes electrónicos sobre las placas requieren toda la documentación previa generadas en las etapas anteriores. Una vez que se logran ensamblar todos los dispositivos, se realizan

[11]kiCAD es un entorno de *software* usado para el diseño de circuitos electrónicos. El paquete kiCAD posee licencia GNU GPL (licencia libre).

pruebas sobre éstas, donde se registran e intentan resolver los inconvenientes que se presenten. Es esta etapa clave donde se define si el desarrollo ha sido exitoso o deben realizarse modificaciones, lo que implica volver al comienzo, desde la etapa *Especificaciones generales*.

Modificaciones El desarrollador evalúa la "gravedad" de los inconvenientes presentados en la etapa de *Armado y pruebas funcionales*. También se aclara que resulta fundamental la documentación en esta etapa ya que servirá para describir como se ha llegado a una versión funcional.

Versión Final En el caso ideal, luego de que se hayan realizado las pruebas correspondientes sobre la placa, y todas éstas hayan sido exitosas, la documentación final se referencia a dicha versión.

5.11 Conceptos para el Armado

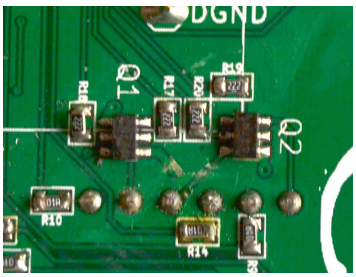
Aquí se busca una metodología a implementar en el proceso de armado de las placas. Si bien a primera vista resulta una actividad sencilla, aquí se utilizan tecnologías SMD para los encapsulados y al tener varios componentes es necesario documentar el proceso a seguir. Además se tiene en cuenta que la documentación generada será útil para otros proyectos similares. El proceso de armado podría componerse por,

- Identificación de los componentes.
- Instalación y reconocimiento de las herramientas necesarias para el proceso de soldadura SMD.
- Testeo visual (utilización de cámaras con zoom) y eléctrico sobre los pines de alimentación de los dispositivos SMD, especialmente los dispositivos semiconductores.

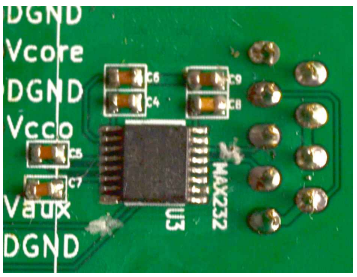
5.11.1 Identificación de los componentes

Para el armado de las diferentes placas, en principio, se podría suponer que se quieren montar y soldar todos los componentes. Por lo tanto se necesitará disponer de la lista de componentes por cada una de las placas. Se debe tener en cuenta que los componentes SMD son de dimensiones muy pequeñas y para algunos dispositivos pasivos no se encuentra visible su valor.

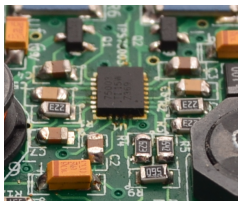
Por otro lado, los componentes que tienen una gran cantidad de pines y dimensiones chicas (como por ejemplo los encapsulados SMD de los microprocesadores, QFN) son los primeros en ser soldados. Al requerir mayor precisión en la ubicación sobre sus pines es recomendable que no se tenga la incomodidad de otros componentes en aproximaciones del componente.



(a) Soldadura simple.



(b) Soldadura media.



(c) Soldadura compleja.

Figura 5.21 Diferentes niveles de complejidad en el soldado de componentes con varios pines SMD.

5.11.2 Herramientas para soldadura SMD

La soldadura de componentes SMD fácilmente puede ser realizada con los soladores comunes. Los equipos especializados para la soldadura de componentes SMD son costosos y en algunos casos innecesarios o fácilmente suplantado con soldadores básicos. Lo que realmente marca una diferencia entre realizar una buena o mala soldadura es tener en cuenta,

... un buen Flux El flux es una resina adherente, que mejora substancialmente la adherencia del estaño. Esta sustancia se presenta en dos formas, una líquida y otra en pasta.

... una buena máscara del PCB La máscara de una placa PCB permite identificar el área sobre el pads que está en contacto con pin/pines de componente a soldar. Esta máscara es de un material aislante y térmicamente resistente al proceso de soldadura. De no existir esta capa sobre el cobre, el estaño se desplazaría por todas las pistas.

Obviamente que se necesitarán otras herramientas comunes como son,

- Lupa o cámara de video con zoom óptico y digital
- Estaño de 0.5 mm de diámetro
- Pinzas para sostener componentes pequeños (SMD)
- Alcohol isopropílico y telas de algodón
- Mallas de cobre para retirar estaño

Cada una de las placas que se armaron siguieron la estructura planteada. En las secciones siguientes se muestran las diferentes placas ya listas y se realizarán observaciones que se encontraron en su armado.

5.12 Placas armadas

Las placas que se listan a continuación fueron armadas en el orden cronológico dispuesto. El orden fue propuesto por la complejidad que presentan cada una de ellas y la rapidez con la que se podría probar independientemente una de otra.

- OOCDLinks
- S3Power
- PHRBoard

Las primeras dos placas ya fueron testeadas anteriormente pero en estas versiones se presentan cambios que no son significativos. La última placa es la continuación del desarrollo llevado anteriormente con la placa FPGA (PHR version BETA)[23].

En la descripción de cada una de las placas armadas se marcarán las *modificaciones* necesarias para mejorar las próximas versiones.

5.12.1 OOCDLink

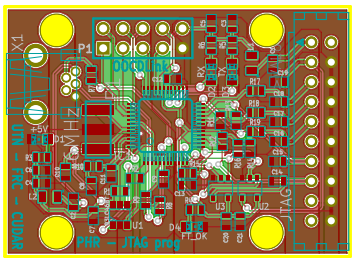
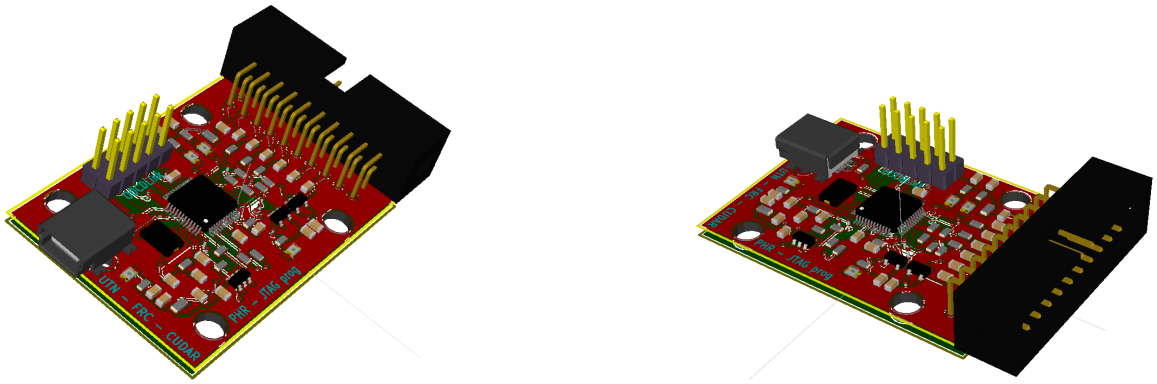
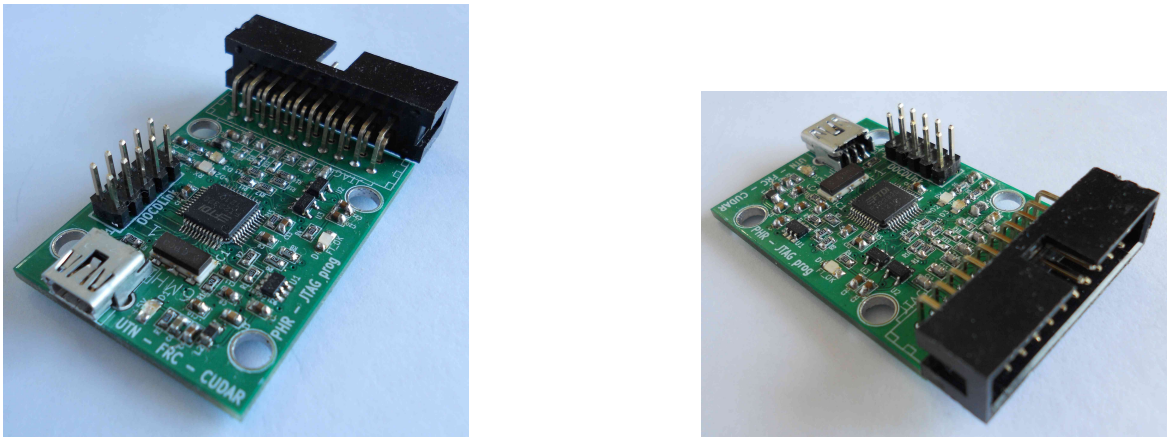


Figura 5.22 Distribución de los componentes en la placa *OOCDLink*.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.23 Modelo en 3D de la placa *OOCDLink*.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.24 Fotografías de la placa *OOCDLink*.

Modificaciones

La placa OOCDLink fue la primera en ser armada y probada. Sobre esta placa se realizó un reporte [23] donde se explica como usar con diferentes placas que implementan el protocolo JTAG para la programación y depuración del dispositivo central utilizado. En el proceso de testeo se encontraron varios cambios a implementar en las nuevas versiones. A continuación se describen los más relevantes.

Gabinete Se podría pensar en dimensionar la nueva versión de la placa para que quepa en algún gabinete estándar. Por lo pronto la versión actual tiene agujeros para sujetar en un principio a una base de acrílico.

Eliminar resistores Se podría pensar en eliminar los resistores que se encuentran conectados entre el FT2232D y el conector JTAG. Estos resistores son: $R17$, $R18$ y $R19$. Originalmente se utilizaban los resistores para que la tensión V_{REF}, a $3.3V$, y las señales TDI, TMS y TCK, a $2.5V$, que darán todas adaptadas. Es decir, los resistores funcionaban como divisores resistivos. Para la placa PHR se utilizan las señales JTAG (TDI, TDO, TMS, TCK) y V_{REF} a $2.5V$. Por lo que ya no se necesitan los resistores divisores.

Cambio de conector JTAG Se podría pensar en utilizar otro conector más pequeño relacionado a las dimensiones. El conector que utiliza la placa OOCDLink actualmente es un conector para microcontroladores ARM7, ARM9, ARM10 y XSCALE, denominado *ARM 20-PIN*. Se podría apuntar a que el conector sea compatible con dispositivos programables PLDs de Xilinx. Por ejemplo, el *Xilinx Parallel Cable IV 14-PIN*. En el caso de no querer perder la compatibilidad con las señales de *debugging* para los microcontroladores ARM, se podría utilizar el mismo conector ARM 20-PIN pero con un encapsulado más pequeño.

Usar el FT232H (simple canal) El FT2232D dispone de dos canales independientes. Uno se utiliza para acceder a un puerto JTAG y el otro como una UART. Si bien la prestación de tener acceso a un puerto serial desde USB resulta muy beneficioso, no lo es así en el costo del programador JTAG. El FT232H cuenta con un solo canal que implementa la tecnología MPSSE (*Multi-Protocol Synchronous Serial Engine*). De esta forma se tendría un diseño más reducido y de menor costo.

Conector micro-USB Se podría utilizar un conector micro-USB en vez del mini-USB.

Utilizar menos indicadores LED La actual placa tiene muchos indicadores LED.

Selección automática del modo de configuración de la FPGA El modo de configuración de la FPGA actualmente es manual. A través del *jumper* $K1$ el usuario elije el modo de configuración. Se podría utilizar el canal libre del FT2232D (actualmente UART) para configurar la señales del modo de la FPGA.

5.12.2 S3Power

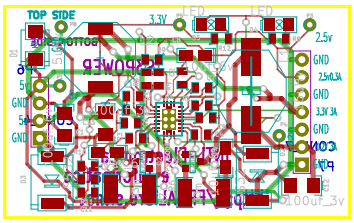
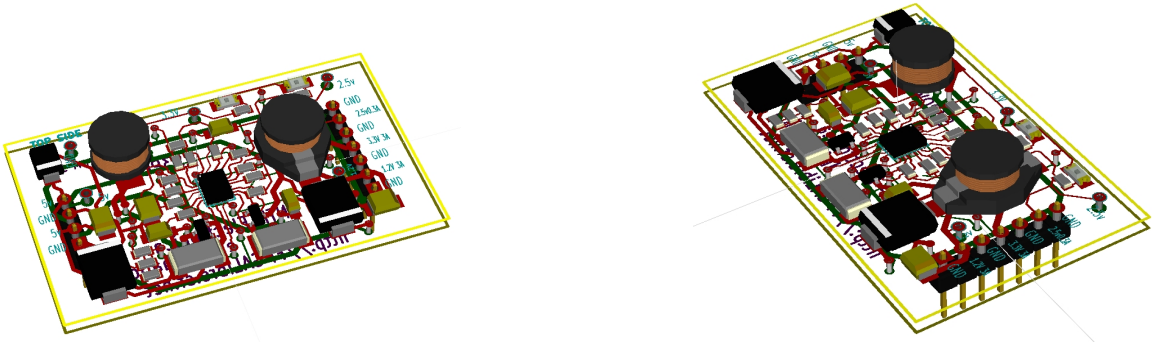
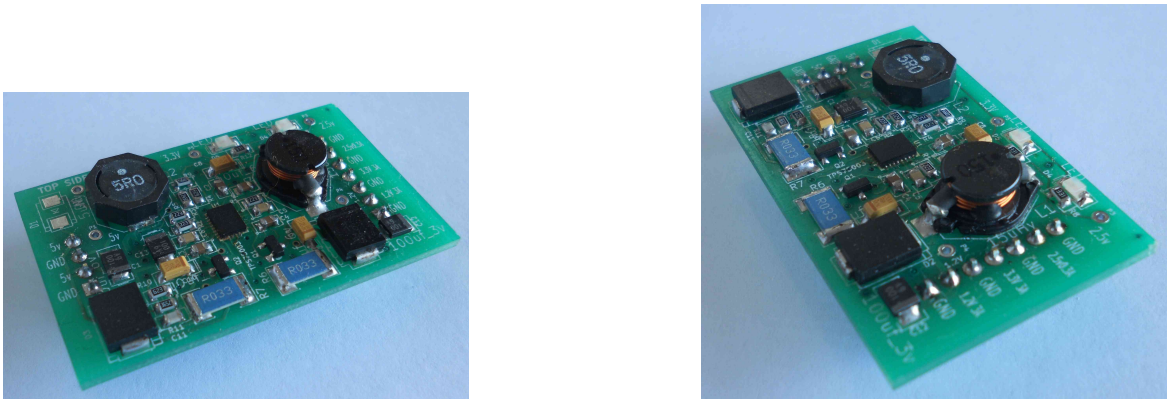


Figura 5.25 Distribución de los componentes en la placa.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.26 Modelo en 3D de la placa *S3Power*.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.27 Fotografías de la placa *S3Power*.

Modificaciones

Para esta versión no hay observaciones/modificaciones que se puedan hacer. Quizá se pueda pensar en el futuro embeber la parte de potencia en la misma placa. En esta versión del proyecto PHR se utilizó la placa S3Power desarrollada por el INTI con la intensión de re-utilizar desarrollos libres y dar un marco de cooperativismo sobre los desarrollos locales.

5.12.3 PHRBoard

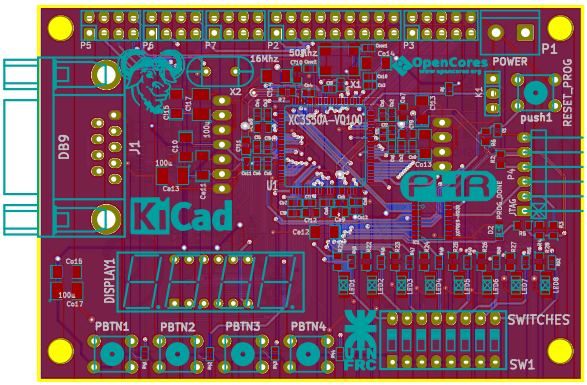
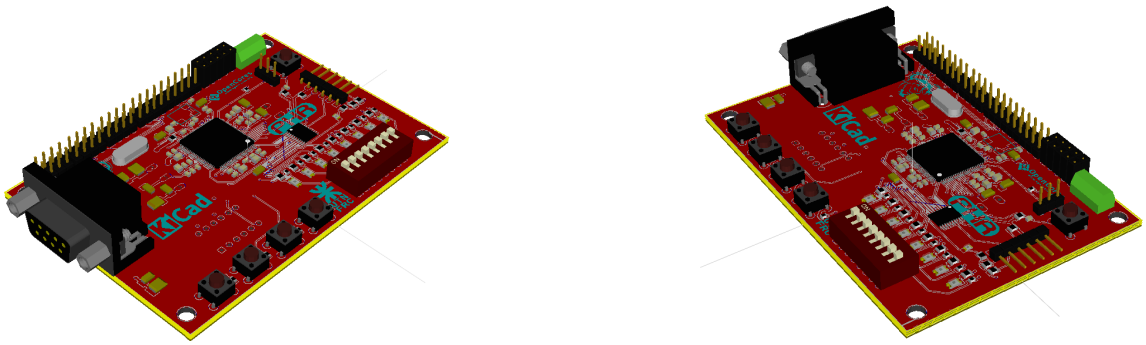
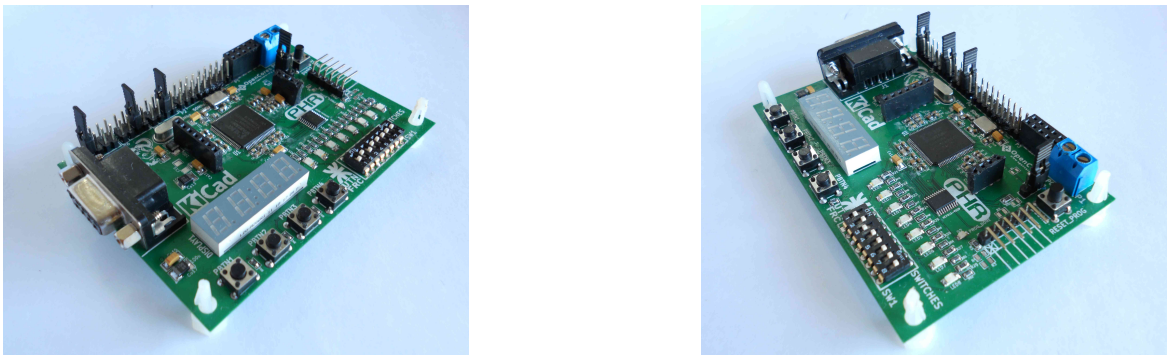


Figura 5.28 Distribución de los componentes en la placa.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.29 Modelo en 3D de la placa *S3Power*.



(a) Perspectiva 1. (b) Perspectiva 2.

Figura 5.30 Fotografías de la placa *S3Power*.

Modificaciones

Recableado de puertos En la FPGA XC3S200A se dispone de varios pines de entradas y salidas. Muchos de estos puertos no son bidireccionales. En la primera versión de la *PHRBoard* se tuvo errores en la asignación de algunos puertos de solo-entrada a periféricos de salida. Se ha solucionado el inconveniente "*recableando*" estos puertos y asignando a otros bidireccionales.

Periférico (etiqueta)	**PIN FPGA PCB**	**PIN FPGA corregido**
IOports_display_seg_e	68	28
IOports_led_8	7	27

Se podría asignar a todos los puertos *solo-entradas* de la FPGA a los periféricos de entrada.

Indicadores Se debería agregar texto que describa las funcionalidades de cada *jumpers*, conectores, o secciones de la placa PHR. A continuación se describen algunas que se encontraron:

- *Jumper* que configura la FPGA (JTAG - PROM)
- Nombre de cada pin del conector JTAG
- Señalar el sentido de *ON/OFF* de las llaves (revisar esquemático y el texto sobre el componente que dice ON)
- Números de los pines de los conectores para entradas y salidas de propósito general (especialmente el *PIN 1*).
- Indicación de las frecuencias seleccionadas con el selector de *clocks*.
- Señalar GND y +5V en la entrada de alimentación.

Conector JTAG Se podría seguir utilizando el conector de *6 PINES* pero cambiando el *footprint* de 90 grados, por uno común. También se podría pensar para la nueva versión utilizar un conector estándar de Xilinx por ejemplo el *XILINX PARALLEL CABLE III AND IV 9-PIN*.

Resistores *pullups* En la versión actual se tiene varios resistores *pullups* que se pusieron provisoriamente. Se debería chequear cuales quedarían definitivamente y cuales no.

Conector E/S Se debe reemplazar los dos conectores de propósitos generales por un solo conector. Preferentemente se utilizará un conector hembra. Se podría tomar como referencia los conectores estándares de Arduino.

Memoria de configuración Se podría reemplazar la memoria XCF02S de Xilinx por otra memoria de menor costo, por ejemplo SPI. Esto también está vinculado con la disponibilidad del *software*. El programa *xc3sprog*, en la versión utilizada, soporta memorias SPI y otras más.

Capítulo 6

Costos y Financiamiento

Con el avance en la industria de la manufacturación de dispositivos electrónicos, hoy en día es posible adquirir tecnología que hace unos 10 años atrás resultaba difícil poseer. Otro factor importante fue el avance en el desarrollo de *software* para el diseño de sistemas electrónicos, estos programas se denominan EDA (*Electronic Design Automation*).
Actualmente en Argentina no hay industria que fabrique componentes electrónicos con las especificaciones requeridas para el proyecto. Existen empresas proveedoras de dispositivos electrónicos pero el costo para el pedido de componentes específico es muy alto.
En la evaluación del pre-proyecto, por parte de los desarrolladores, se consideraron la factibilidad en la adquisición de los materiales necesarios para llevar adelante el proyecto. Se pidió presupuesto de los principales componentes electrónicos, en los proveedores locales, y se realizó un estimativo del costo total si se adquiría todos los materiales necesarios por estas empresas. Lo que llevó a considerar la importación de los componentes por cuenta propia. A continuación se describen los procesos y etapas que atravesó el proyecto en lo que hace a la adquisición de insumo.

La búsqueda de financiamiento del proyecto resulta fundamental debido a la tecnología utilizada y la situación actual de la industria de dispositivos electrónicos regional. Se evaluó ingresar en varios programas de financiamiento que apoyen el desarrollo de nuevas tecnologías. Tanto en organismos gubernamentales estatales, privados y extranjeros.

6.1 Financiamiento del proyecto

La Agencia para el Desarrollo Económico de la ciudad de Córdoba (ADEC) es la entidad ejecutora del Programa de Desarrollo Territorial del Área Metropolitana de Córdoba BID/FOMIN ATN/ME-11806-AR (en adelante el Programa). Dicho Programa es financiado por el Fondo Multilateral de Inversiones (FOMIN) que administra el Banco Interamericano de Desarrollo (BID), por el Gobierno de la Provincia de Córdoba, por la Municipalidad de Córdoba y por la propia ADEC.

Los fondos con los que se financia este convenio provienen del sub componente "Innovación y Capital Humano" que tiene por objetivo contribuir a la vinculación del sistema educativo con los sectores productivos del Área Metropolitana de Córdoba mediante el cofinanciamiento de proyectos que agreguen valor a las empresas y a otras entidades del territorio a través de la implementación de acciones que dichas firmas o asociaciones no realizaban anteriormente. Los recursos de este fondo se asignan por concurso [24].

6.1.1 Programa "Córdoba Innovadora"

"Córdoba Innovadora" es uno de los componentes del "Programa de Desarrollo Territorial del Área Metropolitana de Córdoba". Uno de los objetivo de dicho componente, es contribuir a incrementar la densidad de las relaciones entre los integrantes del sistema de Investigación, Desarrollo e Innovación (I+D+i) con empresas e instituciones. Se espera que ese incremento de relaciones aumente la competitividad local a través de un proceso integrado de desarrollo, destinado tanto a la mejora competitiva de las empresas como a la de los activos territoriales. La estrategia utilizada a tal efecto es la promoción de innovaciones que beneficien al sistema productivo y social, a través de mejoras en productos, procesos productivos, sistemas de organización o comercialización. Los ámbitos en los que se pueden implementar las innovaciones son empresas, organismos del estado y entidades de la sociedad civil [25].

Córdoba Innovadora tiene dos subcomponentes. El primero, "Innovación intersectorial", apoya proyectos asociativos para el desarrollo, adaptación e incorporación de nuevas tecnologías. El segundo subcomponente, "Innovación y capital humano", se centra en la innovación incremental a ser desarrollada por empresas u otras entidades de manera articulada con el sistema de I+D+i y dentro del mismo con el sistema educativo. Ambos subcomponentes disponen de un "fondo" (cuyos recursos están destinados es cofinanciar innovaciones). La selección de los proyectos a los que se

asignarán los recursos se rige - en el caso del fondo para el subcomponente de capital humano - por las bases y condiciones que se exponen seguidamente.

Conceptos utilizados

En el marco de este documento se considera *innovación* a toda actividad que no se realizaba anteriormente en una empresa o institución y que, al efectuarla, agrega valor a un producto o servicio. Esta definición abarca, en consecuencia, tanto a las pequeñas innovaciones incrementales que suelen realizar las pequeñas empresas como a los cambios paradigmáticos que son generados, en un porcentaje significativo de los casos, por organizaciones de mayor porte. Incluye modificaciones en los productos o servicios, procesos, sistemas organizativos, estrategias de comercialización, diseño y similares.

Las condiciones principales a satisfacer por las innovaciones son, en suma i) su incorporación a la firma u organización receptora y, ii) el agregado de valor a la actividad de la misma. Esas condiciones pueden ser satisfecha por una adaptación de prácticas habituales en otros contextos globales o regionales pero que no se habían introducido todavía en la comunidad local.

Se utiliza, en segundo lugar, la concepción de que una *beca* es una ayuda económica que se brinda a estudiantes que - interesados en mejorar el vínculo entre el sistema educativo y las empresas, e instituciones de la sociedad - realicen trabajos finales que constituyan una innovación de utilidad económica social. Dentro de esta categoría se encuentran las *becas de cofinanciamiento* que son aquellas en la que la entidad que las brinda solo aporta el 50 % del valor de un proyecto innovador. El otro 50 % debe ser aportado por la institución o empresa que receptará la innovación y que se beneficiará con el agregado de valor que dicha innovación le brinda. A este 50 % se lo denomina *aporte de contraparte*. El aporte de contraparte se podrá realizar en efectivo o en especie. Se considerará *aporte en especie* a la facilitación, para el logro de los objetivos del proyecto de horas máquina, la realización de trabajos por personal estable de la firma o institución, el suministro de materia prima en stock, la provisión de elementos de anclaje y otros materiales, embalajes, espacio físico, comunicaciones y similares. Dicho aporte se concretará en los plazos establecidos en el *cronograma* de ejecución del proyectoi. Es condición para que un aporte de contraparte sea considerado válido la existencia de un medio de verificación de que el mismo se ha producido.

Objetivos del fondo

Contribuir a la vinculación del sistema educativo con los sectores productivos del Área Metropolitana de Córdoba mediante el cofinanciamiento de proyectos que agreguen valor a las empresas y a otras entidades del territorio a través de la implementación de acciones que dichas firmas o asociaciones no realizaban anteriormente.

Destinatarios de las innovaciones

- Micro, pequeñas y medianas empresas agropecuarias, industriales, comerciales o de servicios, según resolución 21/2010 de la SEPYME y que tengan como mínimo 6 meses de antigüedad desde la fecha de su constitución.
- Grupos asociativos de empresas.
- Cooperativas y mutuales.
- Organizaciones de la sociedad civil.
- Áreas de los tres estamentos del gobierno

Montos del financiamiento

Se otorgarán becas de cofinanciamiento que subsidiarán el 50 % de las diferentes etapas de ejecución de los proyectos que resulten seleccionados. Para el caso de proyectos individuales el monto máximo de las becas de cofinanciamiento será de $ 8.000. – En los casos de proyectos colectivos que incluyan desde dos hasta cuatro estudiantes el monto máximo de la beca de cofinanciamiento será de $16.000 (cualquiera sea la cantidad de estudiantes involucrados).

En el caso de las escuelas que desarrollan acciones de formación profesional de relevancia para el territorio y que realicen proyectos colectivos de más de cuatro estudiantes la presentación la deberá realizar un tutor designado por el establecimiento. (Anexo II Carta tipo de designación de tutor) El monto máximo de la beca de cofinanciamiento de proyectos curriculares será de $16.000 (cualquiera sea la cantidad de estudiantes involucrados).

El monto del proyecto no cubierto por la beca –contraparte–, deberá ser afrontado por la empresa o entidad beneficiaria en efectivo y en especie. Lo aportado debe ser igual o mayor que el 50 % del costo del proyecto. El aporte en efectivo deberá ser, como mínimo, de un 15 % del monto de la contraparte. La acreditación de que

esa inversión se ha efectuado se realizará mediante la verificación de que i) se han alcanzado las metas definidas para cada una de las etapas del cronograma ii) se ha concretado el aporte de contraparte en efectivo y en especie correspondiente a cada una de las sucesivas etapas que se definan en el cronograma.

Evaluación del proyecto

Los ganadores del concurso de proyectos serán seleccionados sobre la base de un orden de mérito. Dicho orden será establecido, según la temática de las propuestas, por uno de los tres comités evaluadores que se constituirán (tecnología industrial, producción primaria, activos territoriales). Cada comité tendrá tres miembros. Apoyará las tareas de los tres tribunales un especialista de reconocida trayectoria.

Los integrantes del comité industrial serán propuestos por el Instituto Nacional de Tecnología Industrial (INTI), por el Ministerio de Producción Industria y Trabajo de la Provincia y por las Cámaras del sector industrial asociadas a la ADEC, los de sector agropecuario por el Instituto Nacional de Tecnología Agropecuaria (INTA), por el Ministerio de Agricultura y Ganadería de la Provincia y por las Cámaras del sector primario asociadas a la ADEC y los del comité de activos territoriales por la Municipalidad de Córdoba, por el Ministerio de Ciencia y Técnica de la Provincia y por el Ministerio de Desarrollo Social de la Provincia. Se financiarán siguiendo el orden de mérito, los proyectos que obtengan más de 60 puntos. El financiamiento comenzará por el proyecto que obtenga mayor puntaje y se agotará cuando se terminen los fondos disponibles para un dado llamado.

Los resultados de esta evaluación se someterán al Comité Directivo del Programa para su ratificación. Los resultados serán sometidos, finalmente a la no objeción del Banco.

Los evaluadores podrán solicitar información complementaria a la presentada en el proyecto y podrán realizar visitas a las empresas o instituciones beneficiarias con el objeto de verificar la pertinencia del proyecto.

Se elaborara un orden de merito de los proyectos hasta agotar los recursos de la correspondiente convocatoria del Fondo. Los solicitantes de los proyectos aprobados serán notificados para la formalización del apoyo mediante la suscripción de una Carta Compromiso. Los solicitantes cuyos proyectos hubiesen sido rechazados también serán notificados. Se pondrá a disposición el informe de evaluación.

6.2 Procedimientos en adquisición de materias primas

6.2.1 Distribuidor de componentes electrónicos

Existen muchas empresas internacionales que realizan la distribución de componentes electrónicos. Particularmente se optó por aquellas que son proveedoras de las empresas locales que son,

- Digi-Key (http://www.digikey.com.ar/)
- Farnell (http://www.farnell.com/)

Debido a la experiencia y facilidades en el soporte comercial, se elige adquirir los componentes a través de *Digi-Key*. Además cuenta con un sistema informático que permite realizar las cotizaciones en linea, lo que resultó importante para la estimación de los gastos en el comienzo del proyecto.

6.2.2 Empresas de trasporte

Al igual que el proceso de compra de los componentes electrónicos, la contratación de una empresa de transporte fue determinada por referencias tanto de miembros del grupo de investigación y desarrollo donde se lleva adelante el proyecto (CUDAR), como también empresas locales que adquieren componentes en el extranjero. Las principales empresas de transporte son,

- FedEx (http://www.fedex.com/ar/)
- UPS (http://www.ups.com/content/ar/es/index.jsx)

Para cualquier de las empresas mencionadas se necesita tener una cuenta de usuario registrado lo que, en principio, parecía ser un inconveniente puedo ser solucionado con la colaboración de un tercero, miembro del CUDAR, que ofreció su cuenta en FedEx para registrar el envío desde la empresa distribuidora de componentes electrónicos (Digi-Key) hasta Córdoba. El costo del envío depende del peso del paquete como así también de sus dimensiones. Además uno puede seleccionar el tiempo que tardará la entrega, obviamente que mientras menos tiempo tarda el envío es más costoso. El tiempo de entrega ha sido importante en el diagrama temporal de actividades, pues hasta que no se tuvo los componentes, se paralizó los trabajos relacionados con el *hardware*.

6.2.3 Fabricante de PCB

Los diseños de *hardware* fueron fabricados en el exterior de país, los factores por influyeron fueron el **costo** de las placas y el **tiempo** que le tomaba a las empresas locales en obtener el producto final. Seguramente si los diseños fueran un poco más sencillos se podría realizar en el país, pero debido a las especificaciones tecnológicas de los dispositivos semiconductores utilizados no fue posible, por lo menos con el presupuesto asignado. Hay una gran oferta en la fabricación de *PCB* pero se contaba con una buena experiencia comercial con *PCBWING*[1], pues ya se habían pedido la fabricación de placas en otros proyectos. Al igual que la empresa distribuidora de componentes electrónicos (*Digi-Key*), *PCBWING* dispone de un sistema informático que presupuesta las posibles órdenes que quiera uno fabricar y se puede realizar el envío por *FedEx*.

6.2.4 Observaciones

No todo los recursos de *hardware* fueron adquiridos en el extranjero en forma directa. Los dispositivos comunes se compraron en el país (directamente en Córdoba Capital). Esto permitió que se pudiera agilizar el trámite de facturación, y dicho costos fueron adjudicados a la contra-parte del proyecto (Departamento de Ingeniería Electrónica) [2].

Al adquirir componentes del extranjero, se tuvo que designar tiempos de "espera" hasta la llegada tanto de los componentes electrónicos como las placas (PCB). Es aquí donde resultó de suma utilidad el uso de ***Diagramas de Gantt***. El *diagrama de Gantt* es una popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. A pesar de esto, el diagrama de Gantt no indica las relaciones existentes entre actividades.

Los tiempos que requerían los envíos debían ser asignados de forma tal que se pudiera seguir trabajando en el desarrollo del proyectos sin necesitar ninguno de los paquetes pedidos. Un ejemplo de esto se puede ver en la Figura 6.1, donde se puede ver como se ubicaron las tareas de envíos.

[1] http://www.pcbwing.com/

[2] Universidad Tecnológica Nacional – Facultad Regional Córdoba

6.2.5 Inconvenientes

Los principales problemas que se presentaron en el proyecto, ajenos a lo meramente técnico, fueron de índole comerciales. Al realizar compra en el exterior se tuvo que contar con cuenta corriente o tarjetas de créditos internacionales. Además de los impuesto de importación, se tuvo que pagar intereses que graban al uso de divisa extranjera. Todos estos costos agregados no fueron considerados a la hora de estimar el gasto total del proyecto.

En base a la documentación que ofrece el fabricante de los dispositivos electrónicos principales, se debía cumplir con determinadas especificaciones en el diseño de las placas. Lo que llevó al estudio de nuevas tecnologías, que demandó horas en la búsqueda de complementos de *software* (por ej.:librerías para KiCAD) como así también recopilación de información por parte de los proveedores de los componentes electrónicos y el fabricante de PCB.

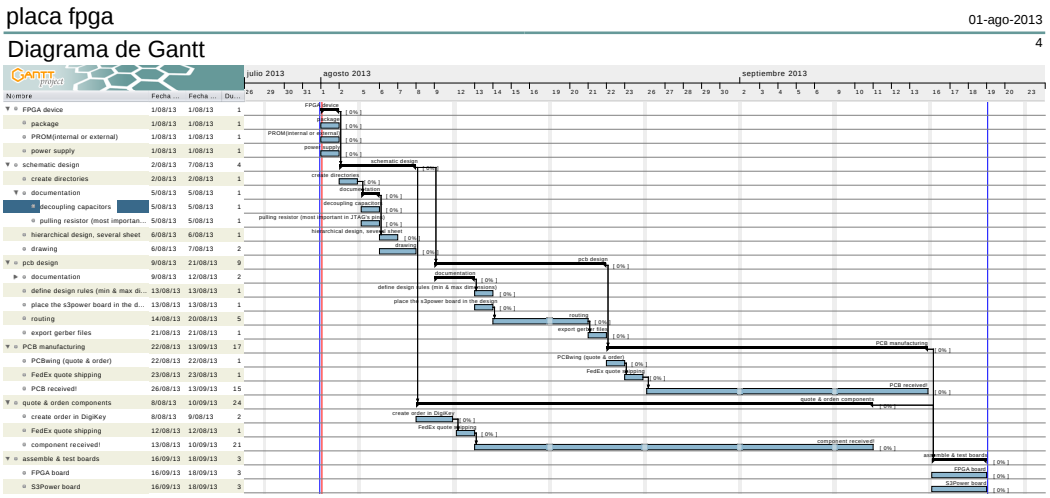


Figura 6.1 Diagrama de Gantt para el desarrollo de la placa *PHR*.

6.3 ¿Costos en *Software*?

Los desarrollos y herramientas de *software* que se integran en el proyecto son complementarios a los recursos de *hardware*. Se han requerido varios programas informáticos en todo el proceso. Si bien no es el objetivo del presente informe realizar una descripción de cada uno de los *software* utilizados, resulta necesario hacer una breve descripción de los mismos.
Todas los software utilizados son programas con ***Licencias Libres****. Las motivaciones de esta elección son varias, la más importante es que el proyecto será transferido a una institución académica, por lo que es imprescindible la libertad en el acceso de estas herramientas. El* ***Software Libre*** *le otorga libertades al usuario del programa y no plantea ninguna restricción en el aprendizaje como sí lo hacen los software con licencias privativas.*

6.3.1 Herramientas utilizadas

Las herramientas de *software* utilizadas para el desarrollo son,

Debian Debian o Proyecto Debian (en inglés *Debian Project*) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en *software* libre. El sistema se encuentra precompilado, empaquetado y en un formato deb para múltiples arquitecturas de computador y para varios núcleos [26].

KiCAD KiCad es un entorno de *software* usado para el diseño de circuitos eléctricos, muy flexible y adaptable, en el que se pueden crear y editar un gran número de componentes y usarlos en Eeschema. KiCad permite el diseño de circuitos impresos modernos de forma sencilla e intuitiva. Además, en Pcbnew, los circuitos se pueden diseñar con múltiples capas y ser visualizados en 3D [27].

Emacs Emacs es un editor de texto con una gran cantidad de funciones, muy popular entre programadores y usuarios técnicos. Gnu Emacs es obviamente parte del proyecto GNU y la versión más popular de Emacs con una gran actividad en su desarrollo. El manual de GNU Emacs lo describe como "un editor extensible, personalizable, auto-documentado y de tiempo real" [28].

subversion Subversion (SVN) es un sistema de control de versiones diseñado específicamente para reemplazar al popular CVS. Es *software* libre bajo una licencia

de tipo Apache/BSD y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de comando [29].

OpenCores OpenCores es la comunidad más grande del mundo de *hardware open source* de desarrollo digital a través de herramientas EDA (*Electronic Design Automation*) [30].

LaTeX LaTeXes un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas [31].

Inkscape Inkscape es un editor de gráficos en formato vectoriales SVG, gratuito, libre y multiplataforma. Las características de SVG soportadas incluyen formas básicas, trayectorias, texto, canal alfa, transformaciones, gradientes, edición de nodos, exportación de SVG a PNG, agrupación de elementos, etc [32].

LibreOffice LibreOffice es una suite ofimática libre y de código abierto desarrollada por *The Document Foundation*. Se creó como bifurcación de OpenOffice.org en 2010 [33].

Bash Bash (*Bourne again shell*) es un programa informático cuya función consiste en interpretar órdenes [34].

Los *software* anteriormente enunciados representan la mayoría de los utilizados en el proyecto. Existen otros *software* específicos, con diferentes lenguajes, que no se describen.

6.3.2 Desarrollo de *Scripts*

La sección 6.3.1 hace referencia a las herramientas utilizadas. En la presente sección se describe la función que cumple la generación de código de programa necesarios para acceder al *hardware*. Estos código de programa, también se los llama *scripts*. Los *scripts* se refieren a un grupo de texto en un determinado orden que son interpretados por un programa para ser procesados. En nuestro caso, los programas que se utilizan son ***OpenOCD* y *xc3sprog***. La Figura 6.2 representa la vinculación entre los recursos de *Software* y *Hardware*.

La estructura de los *scripts* son definidos por el *software* con el cual procesar (*OpenOCD* o *xc3sprog*). Se dispone de gran información sobre estos programas en los

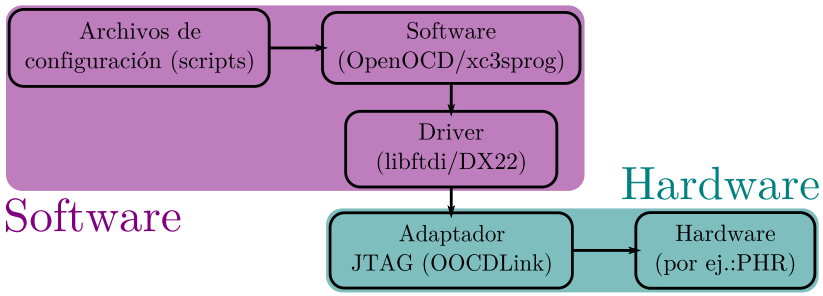


Figura 6.2 Diagrama en bloque de la conexión entre el *Software* y el *Hardware*.

sitios web oficiales como así también en listas de correos que se conforma por usuarios y desarrolladores de los *software*. En el *Manual de Usuario* se describen la forma en que se utilizan los programas.

6.3.3 Repositorio del proyecto

Para llevar adelante un proyecto de desarrollo de *hardware/software* (HW/SW), es sumamente importante disponer de un repositorio donde alojar toda la información. De forma que uno pueda acceder a ellos en forma segura y sin pérdidas de información. Además contar con la posibilidad de que se lleve un registro de los cambios que se vayan realizando sobre cada archivo. En nuestro caso se utiliza un repositorio *SVN* que se encuentra alojado en los servidores de *OpenCores*[3].
Existen muchos sitios en la red que ofrecen este tipo de servicios informáticos, tanto gratis como pagos. Una gran parte de los sitios que dan alojamiento a proyectos en forma gratuita son sustentados por empresas tecnológicas y aportes voluntarios de la comunidad. Además cada sitio tiene diferentes requerimientos para dar alojamiento a un nuevo proyecto (por ejemplo: tipo de licencia, tamaño de archivos, etc.). Como se ha dicho antes en la Sección 6.3.1, OpenCores es uno de los sitios web que hospeda a una gran cantidad de proyectos de HW/SW abiertos de todo el mundo. Otros sitios interesantes donde disponen de infraestructura para alojar proyectos abiertos son,

- SourceForge (https://sourceforge.net/)
- GitHub (https://github.com/)
- CodeGoogle (http://code.google.com/)

[3]http://opencores.org/project,phr

6.3.4 Observaciones

Los programas que se utilizan para interactuar con el *hardware*, *OpenOCD* y *xc3sprog*, no disponen de un entorno gráfico. La forma en que se manipulan es mediante lineas de comando *Bash*. Esto no genera imposibilidades alguna al usuario, pero sí la implementación de un entorno gráfico podría ayudar o facilitar su uso (por ejemplo: con el empleo de botones, visualización del estado del *hardware*, etc.).
Otra observación que se documenta es la posibilidad de integrar las herramientas del fabricante del dispositivo a programar, en nuestro caso Xilinx Inc., con las herramientas que disponemos (*OpenOCD* y *xc3sprog*). El ciclo completo por parte de los usuarios del proyecto arranca con el diseño de un circuito con un lenguaje de programación descriptivo (VHDL o Verilog), luego se sintetiza con una herramienta del fabricante del dispositivo y luego con el archivo que genera se lo debe cargar al *hardware*. Estos últimos pasos podrían realizarse con nuevos *scripts* que se agregan a los ya utilizados.

6.4 Costos finales del desarrollo

Al llevar adelante la fabricación de un desarrollo de prototipaje, los costos son elevados y más si se trabajo con tecnologías no comercializadas en la región. Es importante remarcar esta situación pues si se compara el valor que se ha invertido para poder armar siete o diez placas, con el valor que costaría comprar las plataformas comerciales, claramente es mucho más barato adquirirlas y por tratarse de tecnologías como los PLDs la única opción es importarla del extranjero. El gran problema aquí es el volumen que se maneja. No es lo mismo emprender el desafío de fabricar 10 placas que fabricar 10,000 placas. Un estudio futuro podría permitir evaluar los costos del proyecto para una cantidad mayor a 10,000 pero no solo se trata de ofrecer un producto a bajo precio, sino también se debe tener la certeza que el desarrollo tiene una demanda que amerita un gran volumen.

A continuación se detallan los gastos necesarios para el armado de las diferentes placas que componen el proyecto *Plataforma de Hardware Reconfigurable*.

6.4.1 Placa PHRBoard

Costos de componentes y placa

Componente	Encapsulado	Cantidad	precio p/u	precio total ($)
Capacitor 100nF	SM0603	10	0.00417	0.0417
Capacitor 18pF	SM0603	2	0.00910	0.01820
Capacitor 10μF	SM1206	8	0.15400	1.23200
Capacitor 1nF	SM0603	15	0.00510	0.07650
Capacitor 10nF	SM0603	16	0.00465	0.07440
Capacitor 100μF	SM1210	3	0.48240	1.44720
Capacitor 15pF	SM0603	1	0.01210	0.01210
Resistor	SM0603	28	0.00207	0.05796
Resistor	SM0805	11	0.0030	0.03300
Diodo LED	SM1206	9	0.0776	0.69840
Diodo Schottky	DO-106 A	1	0.32800	0.32800
Display	DIP-12	1	5.39000	5.39000
XCF02S	VOG20	1	4.25000	4.25000
Conector DB-9		1	0.80000	0.80000
Bornera		1	1.20000	1.20000
BC807DS	SOT457	2	0.31600	0.63200
Switch-DIP	DIP-16	1	0.90000	0.90000
XC3S200A	VQG100	1	12.37000	12.37000
MC74HC4060A	TSSOP16	1	0.50600	0.50600
ST3232	TSSOP16	1	0.90000	0.90000
Oscilador 50Mhz	DFN-6	1	1.31000	1.31000
Cristal 16Mhz	HC-49	1	0.30000	0.30000
PCB	-	1		40.872
			Total (en Dólares)	**73.44946**

6.4.2 Placa S3Power

Costos de componentes y placa

Componente	Encapsulado	Cantidad	precio p/u	precio total ($)
Resistor	SM0805	10	0.0030	0.03
Resistor	SM2512	2	1.27	2.54
Capacitor 100μF	SM1210	3	0.4824	1.4472
Capacitor 100nF	SM0805	3	0.0247	0.0741
Capacitor 1.5nF	SM0805	2	0.0244	0.0488
Capacitor 10nF	SM0805	1	0.00777	0.00777
Capacitor 10μF	SM1206	3	0.154	0.462
Capacitor 10pF	SM0805	1	0.02520	0.02520
Diodo LED	SM1206	2	0.0776	0.1552
Diodo Schottky	DO-214 AB	2	0.458	0.916
Diodo Zener	DO-214 AA	1		
Inductor		2	3.03	6.06
Mosfet	SOT23-3	2	0.73	1.46
TPS75003	20-QFN	1	4.902	4.902
PCB	-	1	8.16	8.16
			Total (en Dólares)	**25.872**

6.4.3 Placa OOCDLink

Costos de componentes y placa

Componente	Encapsulado	Cantidad	precio p/u	precio total ($)
Resistores	SM0603	19	0.00207	0.03933
Capacitor 10nF	SM0603	2	0.00465	0.0093
Capacitor 100nF	SM0603	10	0.00417	0.0417
Capacitor 2.2μF	SM0603	2	0.05600	0.112
Capacitor 33nF	SM0603	1	0.01540	0.01540
Capacitor 47pF	SM0603	6	0.14000	0.84
Diodo LED	SM0805	4	0.05900	0.236
Resonador 6Mhz	CSTCC	1	0.48800	0.48800
Inductor (choque)	SM0805	2	0.11720	0.23440
FT2232D	48-LQFP	1	6.99000	6.99000
93LC46BT	SOT23-6	1	0.19600	0.19600
SN74AUP1G125	SOT23-5	2	0.32880	0.6576
Conector mini-USB	USB-MB-H	1	1.26920	1.26920
PCB	-	1	7.665	7.665
			Total (en Dólares)	**18.79393**

6.5 Análisis de costos

El costo *Real* del proyecto completo resulta ser la suma de los costos de las diferentes placas armadas.

Placa	Costo ($)
PHRboard	73.44946
S3Power	25.87200
OOCDLink	18.79393
Total (en Dólares)	**118.11539**

Como se dijo anteriormente, el costo que aquí se presenta es sobre el valor del dispositivo prototipo, fabricando no más de 10 unidades completas. Una proyección a futuro de los costos de inversión para emprender la fabricación de estas placas puede centrar sus objetivos en:

- Reemplazar componentes que presenten varias prestaciones cuando en realidad se los utilice para una tarea definida. Por ejemplo, reemplazar el FT2232D por el FT232H.
- Diseñar un sistema de alimentación que cumpla las mismas funciones que el TPS75003. Esta observación se hace considerando la inversión que demanda utilizar este chip con excelentes prestaciones pero costoso.
- Considerar que los pedidos por futuras placas (PCBs) son generalmente más baratas que la primera orden. Esto es así ya que los fabricantes ya disponen de las matrices necesarias para el fabricado de los PCBs.

Con estas observaciones es posible recudir el costo total de la plataforma. De todas formas se debe tener en cuenta que *la única forma de realizar un producto competitivo a los desarrollos comerciales es fabricar una cantidad importante de forma tal que los costos en componentes electrónicos se vea reducido.*

Capítulo 7

Programación de la PHR

7.1 Introducción

Para transferir el diseño que realiza el usuario a la FPGA es necesario establecer una comunicación entre la computadora donde se sintetiza el proyecto y la FPGA de la placa PHR. En bajo nivel la computadora se comunica a través de un puerto USB con el chip FT2232D de la placa OOCDLink que luego se conecta mediante interfaz JTAG con la FPGA y la PROM. Subiendo en el nivel de abstracción es necesario un software que tome un archivo generado por la herramienta de síntesis y lo envíe a través de los drivers del sistema operativo al dispositivo concreto [35].

PHR GUI es una interfaz gráfica que permite al usuario transmitir los datos de una manera muy sencilla sin la necesidad de tener que sumirse en los pormenores de la transferencia del fichero. Si bien este es el software con el que interactúa el usuario, tiene que considerarse simplemente como un front-end para la aplicación que hace el verdadero trabajo. Esta es la denominada xc3sprog [36].

Xc3sprog[1] es en realidad un conjunto de aplicaciones de licencia libre que funciona en linea de comandos y sirve para programar mediante JTAG varios dispositivos. Su nombre hace referencia a que inicialmente fue diseñado para la familia de FPGA Spartan-3 de Xilinx. Sin embargo se ha extendido el manejo a otros tipos de dispositivos que incluyen otras FPGAs, CPLDs, XCF flash PROMs, microprocesadores AVRs de Atmel y memorias flash SPI. Xc3sprog soporta varios cables JTAG, incluyendo cables de puerto paralelo y programadores USB.

Es además capaz de funcionar con los drivers propietarios del chip FT2232D o

[1] http://xc3sprog.sourceforge.net/

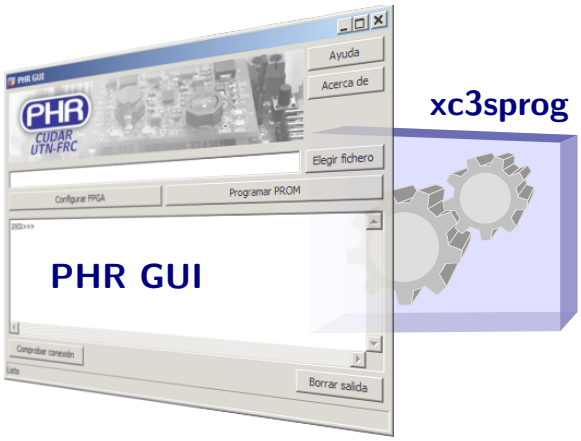


Figura 7.1 PHR GUI es una interfaz que facilita el uso de xc3sprog.

con la libreria libre libUSB. Se encuentra disponible para los sistemas GNU/Linux y Windows.

7.2 Instalación

7.2.1 MS Windows

Actualmente PHR GUI para Windows funciona con los drivers propietarios ofrecidos por el fabricante del chip FT2232D, Future Technology Devices International[2].

El procedimiento general para hacer correr la aplicación en los sistemas Windows consiste en primero descargar los drivers propietarios del FT2232D, luego instalarlos para poder conectar la placa OOCDLink y por último instalar PHR GUI v0.1 que distribuye los binarios de xc3sprog.

A continuación se desarrolla el procedimiento específico para Windows XP.

Windows XP

Necesitará el hardware PHR y los siguientes paquetes de software:

Drivers D2XX: Son los drivers provistos por FTDI y están disponibles en distintas formas. Este manual ejemplifica la instalación con la versión ejecutable. En particular se usa el archivo `CDM v2.10.00 WHQL Certified.exe`, pero se recomienda la versión mas nueva al momento de la descarga.

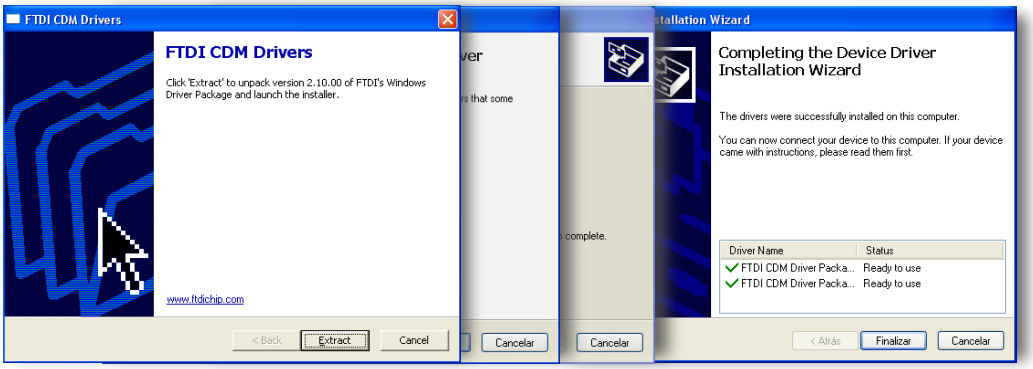
[2]http://www.ftdichip.com/

Para descargarlo consulte: `http://www.ftdichip.com/Drivers/`. En la misma página puede ver el procedimiento mas detallado de instalación.

Instalador de PHR GUI v0.1: Se puede obtener de la sección de descargas en el sitio web del proyecto: `http://opencores.org/project,phr,descargas`.

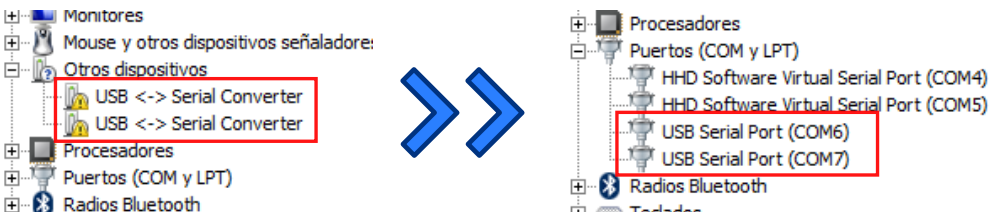
Procedimiento:

1. Ejecutar el asistente de instalación de los drivers D2XX. Esto prepara al sistema para que en el próximo paso se reconozcan los controladores de una forma mas sencilla.

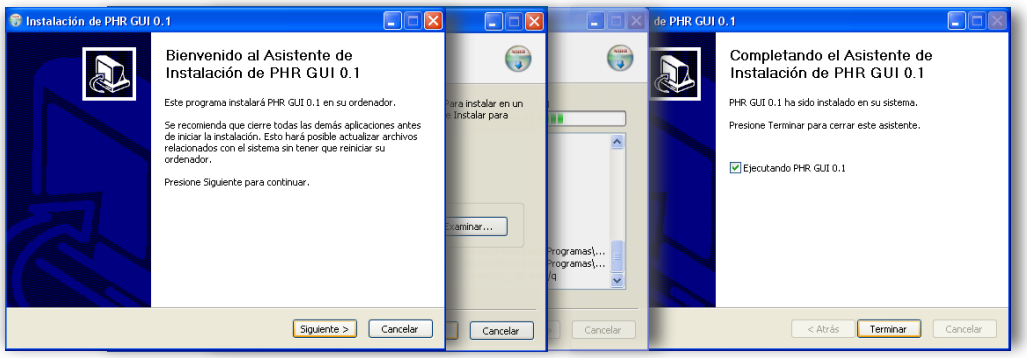


Si Windows le muestra una advertencia de que el software no supera la prueba de compatibilidad con Windows XP elija *Continuar*.

2. Conectar la PHR al puerto USB de la computadora, la cual debe usar los drivers pre-instalados para configurar dos nuevos dispositivos. A continuación se muestra como cambia el Administrador de dispositivos de Windows.



3. Ejecutar el instalador de PHR GUI v0.1. Una vez terminado el asistente puede optar por abrir el programa que debe estar listo para ser utilizado.



7.2.2 GNU/Linux

La instalación en los sistemas GNU/Linux no es necesaria ya que PHR GUI v0.1 en un software desarrollado en el lenguaje Python (versión 2.7.6) y las distribuciones GNU/Linux incluyen un interprete Python. Sin embargo hay que seguir una serie de pasos que están relacionados en su mayoría con el software xc3sprog. Se describe el procedimiento general:

1. Obtener los binarios de xc3sprog. Para esto hay dos opciones:

   (a) Descargar desde la web los binarios ya compilados para la arquitectura específica (`http://sourceforge.net/projects/xc3sprog/`).

   (b) Descargar desde la web o a través de svn el código fuente del proyecto y compilarlo manualmente (Método mas recomendado que el anterior).

   Nota: Xc3sprog hace uso de los paquetes *usblib* y *libftdi*.

2. Desde el sitio web de PHR descargar PHR GUI v0.1 para GNU/Linux.

3. Descomprimir el archivo anterior y moverse en los directorios hacia la carpeta descomprimida. El script que corre la aplicación es el `phr-gui.py`.

```
user@host:~/$ tar -xzf phr-gui-linux.tar.gz
user@host:~/$ cd phr-gui-linux
user@host:~/phr-gui-linux$ ls
crear-enlace.sh  ejemplos  grafica  phr-gui.py
```

4. PHR GUI buscará el software xc3sprog en una carpeta del mismo nombre. Se puede mover el directorio o crear un enlace:

```
user@host:~/phr-gui-linux$ ln -s /carpeta/de/los/binarios
   xc3sprog
```

5. Para una ejecución correcta de xc3sprog, éste debe tener permisos especiales de superusuario.

```
user@host:~/phr-gui-linux$ cd xc3sprog
user@host:~/phr-gui-linux/xc3sprog$ su
Password:
root@host:/home/user/phr-gui-linux/xc3sprog# chown root:root
   xc3sprog
root@host:/home/user/phr-gui-linux/xc3sprog# chmod 4755 xc3sprog
root@host:/home/user/phr-gui-linux/xc3sprog# exit
user@host:~/phr-gui-linux/xc3sprog$ cd ..
user@host:~/phr-gui-linux$
```

   Mediante `chown` se cambia el propietario del archivo al usuario root. Luego, con `chmod` se modifican los permisos:

   4: Hace que al ejecutarse el archivo se le asigne el UID del propietario (root).

   7: Confiere permisos de lectura, escritura y ejecución al propietario (root).

   5: Confiere permisos de lectura y ejecución al grupo.

   5: Confiere permisos de lectura y ejecución al resto de los usuarios.

6. Al asignar permisos de ejecución, `phr-gui.py` debe estar listo para ejecutarse:

```
user@host:~/phr-gui-linux$ chmod +x phr-gui.py
user@host:~/phr-gui-linux$ ./phr-gui.py
```

7. Para tener una experiencia totalmente gráfica se recomienda crear un lanzador. El script `crear-lanzador.sh` se encarga de esto.

```
user@host:~/phr-gui-linux$ chmod +x crear-enlace.sh
user@host:~/phr-gui-linux$ ./crear-lanzador.sh
```

   Se genera un nuevo archivo que puede moverse a un lugar conveniente del entorno gráfico.

7.3 Uso de la interfaz

Para poder transferir el diseño a la placa el usuario debe realizar la síntesis del proyecto y generar un archivo en formato BIT que es el formato por defecto de Xilinx.

Una ves obtenido el fichero, la PHR ofrece dos posibilidades al momento de descargar el diseño sobre la FPGA. Un método consiste en transferir los datos directamente a la FPGA (*configuración de la FPGA*) y el otro método consiste en transferirlos a la PROM (*programación de la PROM*). El primero es mucho mas rápido pero debido a la volatilidad de la FPGA tiene la desventaja de que se pierde el diseño una vez que el circuito es desenergizado. Contrariamente, el segundo es un procedimiento mas lento pero su ventaja radica en que los datos se conservan en los sucesivos ciclos de encendido y apagado.

La forma en que se setea la FPGA para que ésta tome los datos de una u otra fuente es mediante un selector en donde un jumper se cambia de posición. El hecho se ilustra en la Fig. 7.2. Luego el usuario debe interactuar con la interfaz PHR GUI (ver Fig. 7.3) para que la computadora efectivamente envíe los datos correspondientes.

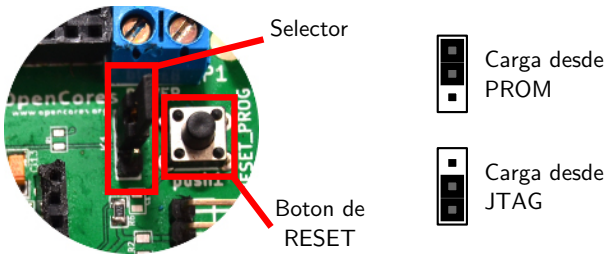


Figura 7.2 Selección de la fuente de carga.

7.3.1 Configuración de la FPGA

Procedimiento para transferir el diseño a la FPGA:

1. Coloque el jumper de la PHR para que la FPGA cargue desde JTAG (ver Fig. 7.2).
2. Elija el archivo BIT que desea transferir desde el diálogo que surge al presionar el botón *Elegir fichero*.
3. Presione el botón *Configurar FPGA*.
4. El software xc3sprog es invocado y PHR GUI muestra su salida.
5. El diseño debe comenzar a funcionar una vez terminado el proceso de descarga.

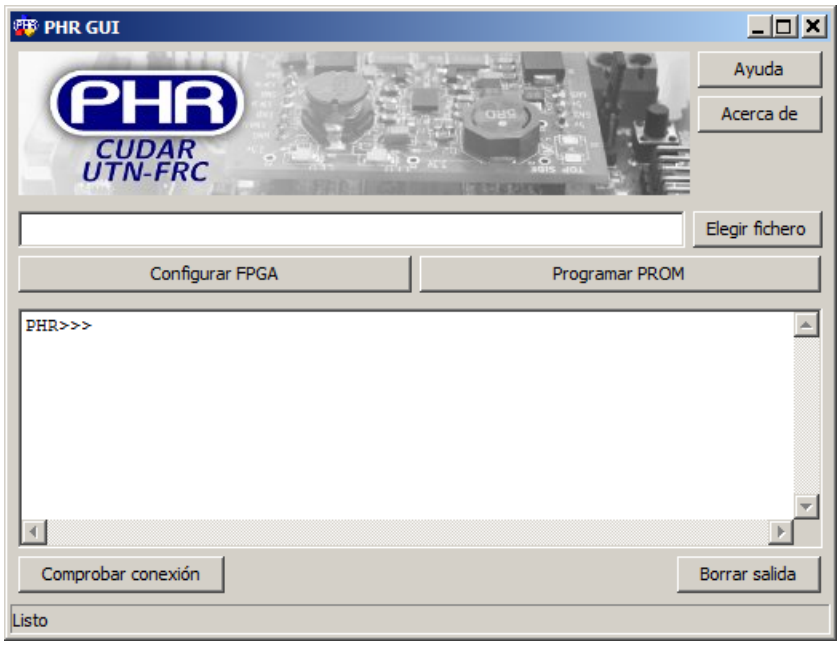


Figura 7.3 Interfaz de usuario PHR GUI.

7.3.2 Programación de la PROM

Procedimiento para transferir el diseño a la PROM:

1. Coloque el jumper de la PHR para que la FPGA cargue desde PROM (ver Fig. 7.2).
2. Elija el archivo BIT que desea transferir desde el diálogo que surge al presionar el botón *Elegir fichero*.
3. Presione el botón *Programar PROM*.
4. El software xc3sprog es invocado y PHR GUI muestra su salida.
5. Una vez terminado el proceso de programación, presione el botón de RESET en la placa PHR (ver Fig. 7.2) para que la FPGA cargue el diseño.

7.3.3 Interpretación de las salidas de texto

Comprobar conexión

Para verificar que el hardware esté bien conectado se puede recurrir al botón *Comprobar conexión*. Al presionarlo se espera una salida como la siguiente:

```
PHR>>> xc3sprog\xc3sprog -c ftdi -L -j -v
XC3SPROG (c) 2004-2011 xc3sprog project $Rev: 303 $ OS: Windows
Free software: If you contribute nothing, expect nothing!
Feedback on success/failure/enhancement requests:
        http://sourceforge.net/mail/?group_id=170565
Check Sourceforge for updates:
        http://sourceforge.net/projects/xc3sprog/develop

Using built-in device list
Using built-in cable list
Cable ftdi type ftdi VID 0x0403 PID 0x6010 dbus data 00 enable 0b
   cbus data 00 data 00
Using FTD2XX, Using JTAG frequency 1200000
JTAG chainpos: 0 Device IDCODE = 0x02218093     Desc: XC3S200A
JTAG loc.:   0  IDCODE: 0x02218093  Desc:      XC3S200A Rev: A  IR
   length:  6
JTAG loc.:   1  IDCODE: 0xd5045093  Desc:        XCF02S Rev: M  IR
   length:  8
USB transactions: Write 6 read 4 retries 0

PHR>>>
```

Lo importante son las dos últimas líneas en donde se reconocen 2 elementos en la cadena JTAG. En la posición 0 con ID 0x02218093 se encuentra la FPGA (XC3S200A). En la posición 1 con ID 0xd5045093 se encuentra la PROM (XCF02S).

Configurar FPGA

Al configurar la FPGA se espera esta salida:

```
PHR>>> xc3sprog\xc3sprog -v -L -c ftdi -p 0 fpgaFile:w:0:BIT
XC3SPROG (c) 2004-2011 xc3sprog project $Rev: 303 $ OS: Windows
Free software: If you contribute nothing, expect nothing!
Feedback on success/failure/enhancement requests:
        http://sourceforge.net/mail/?group_id=170565
Check Sourceforge for updates:
        http://sourceforge.net/projects/xc3sprog/develop

Using built-in device list
Using built-in cable list
Cable ftdi type ftdi VID 0x0403 PID 0x6010 dbus data 00 enable 0b
   cbus data 00 data 00
Using FTD2XX, Using JTAG frequency 1200000
```

```
JTAG chainpos: 0 Device IDCODE = 0x02218093     Desc: XC3S200A
Created from NCD file: test1_top.ncd;UserID=0xFFFFFFFF
Target device: 3s200avq100
Created: 2014/03/19 18:31:15
Bitstream length: 1196128 bits
DNA is 0x95cde80efb9ba0fe
done. Programming time 1044.1 ms
USB transactions: Write 84 read 9 retries 0

PHR>>>
```

La penúltima linea es la que indica que la grabación ha finalizado y muestra el tiempo que demoró el proceso.

Programar PROM

La salida que se lista está resumida puesto que suele ser muy larga:

```
PHR>>> xc3sprog\xc3sprog.exe -v -L -c ftdi -p 1 fpgaFile:w:0:BIT
XC3SPROG (c) 2004-2011 xc3sprog project $Rev: 303 $ OS: Windows
Free software: If you contribute nothing, expect nothing!
Feedback on success/failure/enhancement requests:
        http://sourceforge.net/mail/?group_id=170565
Check Sourceforge for updates:
        http://sourceforge.net/projects/xc3sprog/develop

Using built-in device list
Using built-in cable list
Cable ftdi type ftdi VID 0x0403 PID 0x6010 dbus data 00 enable 0b
   cbus data 00 data 00
Using FTD2XX, Using JTAG frequency 1200000
JTAG chainpos: 1 Device IDCODE = 0xd5045093     Desc: XCF02S
Erasing......done
Erase time 3129.2 ms
Programming does not end at block boundary (nbits = 1196128), padding

Programming block      0/   293 at XCF frame 0x0000.
...
Programming block    292/   293 at XCF frame 0x2480.done
Programming time 2802.2 ms

Verify block      0/   293 at XCF frame 0x0000
...
```

```
Verify block    278/   293 at XCF frame 0x22c0
VerSuccess! Verify time 1893.1 ms
USB transactions: Write 1815 read 906 retries 0

PHR>>>
```

Básicamente se efectúan tres operaciones. Primero se borra la memoria PROM, luego se programa la misma y después se verifican los datos. Para el caso que se muestra (depende de la versión de xc3sprog que se use) la línea que dice `VerSuccess!` indica que la programación fue exitosa.

7.4 Soluciones a problemas frecuentes

Problema: [Windows] Cualquiera de los tres botones, Configurar FPGA, Programar PROM o Comprobar conexión, invocan comandos pero no producen salida alguna

Solución: Si la placa OOCDLink no se conectó nunca en la computadora, posiblemente sea necesario instalar la librería para que xc3sprog funcione correctamente. Consulte la sección 7.2 para instalar la placa.

Problema: [GNU/Linux] Al presionar alguno de los tres botones Configurar FPGA, Programar PROM o Comprobar conexión, se produce una salida similar a:

```
PHR>>> ./xc3sprog/xc3sprog -c ftdi
/bin/sh: ./xc3sprog/xc3sprog: not found

PHR>>>
```

Solución: PHR GUI v0.1 busca los binarios de xc3sprog en el subdirectorio del mismo nombre y no lo encuentra. Debe descargar el software xc3sprog y ubicarlo en dicho directorio (consulte *Instalación en GNU/Linux*, página 101).

Problema: [GNU/Linux] Al presionar alguno de los tres botones Configurar FPGA, Programar PROM o Comprobar conexión, se produce una salida similar a:

```
HR>>> ./xc3sprog/xc3sprog -c ftdi
```

```
XC3SPROG (c) 2004-2011 xc3sprog project $Rev: 303 $ OS: Linux
Free software: If you contribute nothing, expect nothing!
Feedback on success/failure/enhancement requests:
        http://sourceforge.net/mail/?group_id=170565
Check Sourceforge for updates:
        http://sourceforge.net/projects/xc3sprog/develop

Could not open FTDI device (using libftdi): inappropriate permissions
    on device!
Unable to access FTDI device with either libftdi or FTD2XX

PHR>>>
```

Solución: Posiblemente xc3sprog no tenga permisos para acceder al puerto USB. Debe conceder los permisos (consulte *Instalación en GNU/Linux*, página 101).

Capítulo 8

Conclusiones

8.1 La experiencia del emprendimiento

El desarrollo del proyecto PHR ha requerido pasar por todas las etapas del proceso de producción de sistemas electrónicos. Desde los primeros diagramas en bloque, pasando por el diseño del esquemático y PCB de las diferentes placas. Además se realizó la compra de todos los componentes sin intermediarios debido al volumen requerido. Por cada una de estas etapas se realizaba documentación que permita afrontar proyectos similares o simplemente esta documentación sirva como referencias por parte de los estudiantes.

La posibilidad de un financiamiento externo a los recursos económicos con los que cuenta el CUDAR ha demandado por parte de los desarrolladores la necesidad de enfatizar las bondades a nivel regional tenía el ante-proyecto presentado en la postulación para recibir dicho financiamiento. Esto dejó una experiencia satisfactorio ya que no solo se logró el financiamiento por parte de la Agencia ADEC, sino que también se puedo intercambiar experiencias con emprendedores de varias instituciones académicas que buscan lograr hacerse de las becas del programa "Córdoba Innovadora".

8.2 Desarrollos reutilizables

Desde un concepto estratégico se consideró disponer de la etapa de alimentación (Sección 5.9) y la interfaz JTAG (Sección 5.8) en forma independientes a la placa principal PHR. Ambas placas pueden ser reutilizadas en otros proyectos por parte de los estudiantes que tengan acceso al proyecto PHR. Y es que la placa S3power está diseñada para alimentar cualquier sistema basado en las FPGAs Spartan-3 de Xilinx. De la

misma forma la placa OOCDLink soporta el protocolo JTAG que es muy utilizado en los microcontroladores actuales. La modularidad de las diferentes placas, en contraste con el párrafo anterior, presenta la desventaja del costo en la fabricación de los PCBs.

8.2.1 Casos de éxitos

Varios son los casos de éxitos que permiten comprobar las observaciones anteriormente hechas. Es decir, la modularidad de los desarrollos electrónicos son validos y sobre todo cuando el entorno donde se los transfiere son instituciones académicas o de I+D. A continuación se describirán resumidamente las circunstancias donde se implementó algunas de las placas que integran este proyecto. Se hace notar que nuestra intensión es resaltar las bondades que presenta el trabajo cooperativo y el uso de desarrollos abiertos.

Proyecto 1: Confidencialidad de la Comunicacion de Mensajes sobre Redes no Confiables

El primer proyecto en el que se ha colaborado es un Trabajo Final de la Carrera de Ingeniería Electrónica de nuestra facultad. El proyecto está orientado a realizar un sistema que permita la segurización mediante encripción basada en hardware, de mensajes confidenciales que se transmitirán a través de redes no confiables.

Aquí se colaboró con el asesoramiento sobre los recursos de hardware necesarios para llevar adelante el proyecto. Conjuntamente con el estudiante (tesista) se contactó con el Instituto Nacional de Tecnología Industrial donde se ha podido facilitar la adquisición de las placas que conforman el proyecto S3PROTO-MINI [14]. El estudiante, que finalmente se graduó en el año 2012.

Proyecto 2: Implementación de un Sistema en Chip con Microprocesador Soft-Core y Soporte Linux

El proyecto tiene como objetivo la implementación de un sistema embebido basado en hardware1 y software de código abierto. Dicho hardware sera un System on Chip (SoC) montado sobre una placa de desarrollo que deberá además soportar un sistema operativo libre con la finalidad de proveer un sistema integral FPGA-SoC-Sistema Operativo completamente funcional de código abierto para aplicaciones de pequeña envergadura [37]. Los integrantes del proyecto son estudiantes de Ingeniería en Computación de la Facultad de Ciencias Exacas, Físicas y Naturales de la Universidad

Nacional de Córdoba. Estos estudiantes se han graduado en el año 2014.

En este proyecto se colaboró con soporte de la placa OOCDLink 5.8 y las herramientas de software con las que ya se tenía experiencia en la integración de la placa OOCDLink con el proyecto PHR.

Y muchos otros más!

Actualmente se continúan haciendo pruebas de integración de las placas que conforman el proyecto *Plataforma de Hardware Reconfigurable*, con varias otras plataformas y periféricos.

8.3 Un diseño integramente abierto y libre

El proyecto se realizó en su totalidad con herramientas de software libre/abiertas. Por cada etapa del desarrollo se buscó alternativas libres que cubrieran los requerimientos del caso. Se tenía referencias sobre proyectos de las mismas envergadura pero la plataforma PHR requería nuevas tecnologías a implementar que han sido resueltas con herramientas desarrolladas por la comunidad de software/ hardware libre/abierto. Son varias las ventajas por la que se optó y segura optando.

8.3.1 En la ética

El Software Libre tiene sus bases en una ideología que dice el software no debe tener dueños, es un asunto de libertad: la gente debería ser libre de usarlo en todas las formas que sean socialmente útiles.

De esta forma, el movimiento del Software Libre pone lo que es beneficioso para la sociedad por encima de los intereses económicos o políticos.

Entre los beneficios que percibe la sociedad podemos mencionar:

- Tecnologías transparentes, confiables y seguras.
- Tecnologías como bien público.
- Promoción del espíritu cooperativo, en el que el principal objetivo es ayudar a su vecino.
- Precios justos.

8.3.2 En la práctica

El Software Libre ofrece a las personas la posibilidad de utilizar, estudiar, modificar, copiar y redistribuir el software. Para hacer efectivas estas libertades, el código fuente de los programas debe estar disponible.

Gracias a estas libertades obtenemos muchos beneficios prácticos:

- Podemos ejecutar el software cuando queramos y para lo que queramos.
- Podemos aprender de los programas existentes.
- Podemos mejorarlos.
- Podemos adaptarlos para que se ajusten a nuestras necesidades.
- Podemos basarnos en ellos, de forma que evitamos los costos adicionales de empezar un programa desde 0.
- Podemos formar negocios alrededor de la creación, distribución, soporte y capacitación de programas libres.

Y el efecto de todos estos beneficios es la formación de Comunidades enormes alrededor de proyectos de software libre, gracias a las cuales tenemos acceso a desarrolladores, documentadores y testers de todo el mundo.

8.4 Transferencia del proyecto

La transferencia del desarrollo se encuentra en ejecución. En principio se tiene una plataforma funcional. Los pasos siguientes son realizar la construcción de las siete unidades que se pretende armar y para las cuales se dispone de todos los dispositivos electrónicos.

8.4.1 Aporte del sector privado

Recientemente se está realizando comunicación con varias empresas que puedan participar como Sponsors del proyecto, sobre todo con la fabricación de una nueva versión de la placa principal PHRBoard.

Además se participó de la *Expotrónica 2014*, en el Stand de la Universidad Tecnológica Nacional, donde se contactó con varias instituciones académicas que podrían estar interesadas en recibir el desarrollo.

8.4.2 Aporte del sector estatal

Se continúa documentando el proyecto con la intensión de institucionalizar el proyecto para poder contar con recursos económicos específicos para la inversión en la mejora del proyecto. Además se pretende dar mayor difusión del proyecto (PHR) para lograr un apoyo por parte de áreas del Estado relacionadas a la promoción de desarrollos innovadores y las denominadas Tecnologías de la Informática y las Comunicaciones.

Referencias

[1] Volnei A. Pedroni. *Circuit Design with VHDL*. MIT Press, 2004.

[2] Richard Haskell and Darrin Hanna. *Introduccion to Digital Design – Using Digilent FPGA Boards*. LBE Books, 2009.

[3] Gaye Lightbody Roger Woods, John McAllister and Ying Yi. *FPGA-based Implementation of Signal Processing Systems*. Wiley, 2008.

[4] Pablo Cayuela. Actualización de la currícula - incorporación de la lógica programable en ingenierías. *JIDIS 2007*, 2007.

[5] Xilinx. Xilinx university program. `http://www.xilinx.com/support/university/index.htm`, June 2009. Accedido: 2014-05-19.

[6] Altera. Altera university program – learning through innovation. `http://www.altera.com/education/univ/unv-index.html`, June 2009. Accedido: 2014-05-19.

[7] Pereyra y Manfredi Olmedo. Kit de desarrollo educativo con cpld. *FPGA Based Systems. 2nd. Southern Conference on Programmable Logic*, 2006.

[8] Avnet. *Xilinx Spartan-6 FPGA LX9 MicroBoard – User Guide*. Avnet Inc., 2011.

[9] Altera and terasIC. *DE0-Nano – User Guide*. Altera Inc., 2012.

[10] Digilent. *Spartan-3 Starter Kit Board – User Guide*. Digilent & Xilinx., 2005.

[11] Ministerio de Defensa. Instituto de investigación científica y técnicas para al defensa (citedef). `http://www.citedef.gob.ar/i-d/laser/`, June 2009. Accedido: 2014-05-20.

[12] G. Jaquenod J. Siman and H. Mascialino. Fpga-based transmit/receive distributed controller for the tr modules of an l band antenna (sar). *4th. Southern Conference on Programmable Logic*, 2008.

[13] Juan Borgna Diego Brengi, Salvador Tropea. Tarteja de diseño abierto para desarrollo y educación. *Southern Conference on Programmable Logic SPL'07*, 2007.

[14] M. Visentin C. Huy Diego Brengi, Salvador Tropea and R. Melo. S3proto-mini: Tarjeta de hardware libre con fpga de encapsulado bga. *XVIII Workshop Iberchip 2012*, 2012.

[15] Luis A. Guanuco Alexis M. Quinteros and Sergio D. Olmedo. Plataforma de hardware reconfigurable para el diseño de sistemas digitales. *V Congreso de Microelectrónica Aplicada*, 2014.

[16] Xilinx. *Spartan-3 Generation FPGA User Guide*. Xilinx., 2009.

[17] Xilinx. *Spartan-3A FPGA Family Data Sheet*. Xilinx., 2010.

[18] Xilinx. *Platform Flash PROM User Guide*. Xilinx., 2009.

[19] Alexis M. Quinteros and Luis A. Guanuco. *Plataforma de Hardware Reconfigurable – Manual de Usuario*. CUDAR, 2014.

[20] Xilinx. *Spartan-3 Generation Configuration User Guide – Extended Spartan-3A, Spartan-3E and Spartan-3 FPGA Families*. Xilinx Inc., 2009.

[21] Future Technology Devices International Ltd. *FT2232D – Dual USB to Serial UART/FIFO IC*. FTDI Chip., 2010.

[22] Christian Huy and Diego Brengi. Módulo de alimentación para placas con dispositivos fpga. *Instituto Nacional de Tecnología Industrial*, 2011.

[23] Luis A. Guanuco. Plataforma de hardware reconfigurable – armado - testeo y documentación de las placas de prototipaje. Technical report, CUDAR, 08 2012.

[24] ADEC. Anexo iv – carta compromiso. `http://www.desarrolloterritorial.adec.org.ar/cordoba-innovadora/images/Anexo-IV-Carta-Compromiso-2013.doc`, August 2011. Accedido: 2014-05-20.

[25] ADEC. Córdoba innovadora – fondo para la promoción de innovación. `http://www.desarrolloterritorial.adec.org.ar/cordoba-innovadora/images/Bases-y-Condiciones-4-Convocatoria.doc`, August 2011. Accedido: 2014-05-20.

[26] Comunidad Wikipedia. Debian. `http://es.wikipedia.org/wiki/Debian`, May 2014. Accedido: 2014-05-20.

[27] Comunidad Wikipedia. Kicad. `http://es.wikipedia.org/wiki/KiCad`, May 2014. Accedido: 2014-05-20.

[28] Comunidad Wikipedia. Emacs. `http://es.wikipedia.org/wiki/Emacs`, May 2014. Accedido: 2014-05-20.

[29] Comunidad Wikipedia. Subversion (software). `http://es.wikipedia.org/wiki/Subversion`, May 2014. Accedido: 2014-05-20.

[30] Comunidad Wikipedia. Opencores. `http://en.wikipedia.org/wiki/OpenCores`, May 2014. Accedido: 2014-05-20.

[31] Comunidad Wikipedia. LaTeX. `http://en.wikipedia.org/wiki/Latex`, May 2014. Accedido: 2014-05-20.

[32] Comunidad Wikipedia. Inkscape. `http://en.wikipedia.org/wiki/Inkscape`, May 2014. Accedido: 2014-05-20.

[33] Comunidad Wikipedia. Libreoffice. `http://en.wikipedia.org/wiki/Libreoffice`, May 2014. Accedido: 2014-05-20.

[34] Comunidad Wikipedia. Bash. `http://en.wikipedia.org/wiki/Bash`, May 2014. Accedido: 2014-05-20.

[35] Luis A. Guanuco. Plataforma de hardware reconfigurable – jtag – configuración oocd-links, (hardware & software). Technical report, CUDAR, 03 2013.

[36] Alexis M. Quinteros and Luis A. Guanuco. *Interfaz gráfica de usuario PHR GUI v0.1 – Guía de Instalación y uso*. CUDAR, 2014.

[37] Valeria Lovaisa Michelini Roberto Pablo Gomez. Implementación de un sistema en chip con microprosesador soft-core y soporte linux. Master's thesis, Facultad de Ciencias Exactas Físicas y Naturales - Universidad Nacional de Córdoba, Argentina, 2013.

[38] Free Software Foundation. The free software definition - gnu project - free software foundation(fsf). `http://www.gnu.org/philosophy/free-sw.html`, May 2010. Accedido: 2014-05-20.

[39] Free Software Foundation. Free software is a matter of liberty, not price - free software foundation - working together for free software. `http://www.fsf.org/about/`, May 2010. Accedido: 2014-05-20.

[40] TAPR. The tapr open hardware license. `http://www.tapr.org/ohl.html`, May 2011. Accedido: 2014-05-20.

[41] Ars Technica. Tapr introduces open-source hardware license, osi skeptical. `http://arstechnica.com/old/content/2007/02/8911.ars`, May 2011. Accedido: 2014-05-20.

Anexo A

Licencias

A.1 Software Libre y Código Abierto

El *software libre*, o en inglés *free software*, ofrece al usuario la libertad para ejecutarlo, copiarlo, modificarlo y distribuirlo. Las características que tiene que cumplir el software la Fundación para el Software Libre (FSF) [38]. La FSF fue fundada por Richard Stallman en 1985 para promover la libertad del usuario y defender los derechos de todo software libre [39]. En particular, el proyecto GNU es patronizado por la FSF. El software libre permite al usuario el ejercicio de cuatro libertades básicas a saber:

- *Libertad 0*: El software *libre* puede estudiarse y adaptarse a las necesidades de quién lo use. Esta libertad requiere acceso al código fuente y tiene la ventaja de por ejemplo permitir descubrir funciones ocultas o saber como se realiza determinada tarea. Además, al poder adaptar el programa a las necesidades del usuario se pueden suprimir partes que no sean de interés, agregar otras partes que considere importantes, copiar una parte que realiza una tarea y/o asignarla a otro programa.

- *Libertad 1*: El software, sus copias y las modificaciones se pueden distribuir libremente. Esto significa que se posee la libertad de redistribuir el programa ya sea gratis o con algún costo.

- *Libertad 2*: Es posible mejorarlo y hacer pública esas mejoras. La libertad de hacer un programa mejor, implica que se puede hacer menores los requerimientos de hardware para funcionar, que tenga mayores prestaciones, que sus requerim-

ientos no sean tan altos, que tenga menos errores. El poder liberar las mejoras al público quiere decir que si se realiza una mejora que permita un requerimiento menor de hardware, o que haga que ocupe menos espacio, se puede redistribuir ese programa mejorado o simplemente proponer la mejora en lugar público (un foro de noticias, una lista de correo, un sitio web, un FTP, un canal de chat).

- *Libertad 3*: El usuario al poseer el código fuente tiene poder de decisión, ya que podrá elegir quién puede modificar los programas que ha adquirido para mejorarlos (o bien mejorarlos el mismo). Es decir, esto permite que no exista un monopolio, porque en el caso de que un software sea discontinuado el usuario podrá, elegir a un desarrollador para continuar utilizando el software que fue discontinuado. Además el usuario no estará completamente a merced de tener que renovar su hardware y software constantemente según ocurre a menudo con las políticas de las empresas que producen software privativo y también será libre de vender o redistribuir software libre.

Cabe destacar que las libertades 1 y 3 tienen como prerrequisito que se tenga acceso al código fuente. La libertad 2 hace referencia a la libertad de modificar y redistribuir el software libremente licenciado bajo algún tipo de licencia de software libre que beneficie a la comunidad.

El término *free* posee en inglés el doble significado de "gratuidad" y "libertad" del software dependiendo del contexto. Por tal motivo se comenzó a utilizar en 1998 el término *Código Abierto*, o en inglés "Open Source" para intentar aclarar que la palabra libre, se refiere a la libertad y no al precio. Este software de *Código Abierto* permite a las personas usar y modificar el software, compatibilizarlo con otros sistemas operativos o arquitecturas de hardware, compartirlo con otras personas y comercializarlo. Desde el punto de vista de una "traducción estrictamente literal", el significado textual de "código abierto" es que "se puede examinar el código fuente", por lo que puede interpretarse como un término más débil y flexible que el del *software libre*.

La definición oficial de *código abierto* se encuentra en el documento Open Source Definition (OSD) publicado por Open Source Initiative (OSI). El OSI es un movimiento diferente al movimiento del software libre que, aunque es incompatible con este último desde el punto de vista filosófico, es completamente equivalente desde el punto de vista práctico. El propósito de establecer una definición oficial de software de *código abierto* es asegurar que los programas distribuidos con "Licencia de código abierto" estarán disponibles para su continua revisión y mejora. Brindando además a los usuarios la

posibilidad de adaptarlo a sus necesidades y corregir errores rápidamente.

La principal diferencia entre los términos *código abierto* y *libre* es que éste último tiene en cuenta los aspectos éticos y filosóficos de la libertad mientras que el *código abierto* se basa únicamente en los aspectos técnicos. En un intento por unir los mencionados términos, que se refieren a conceptos semejantes, se intentó extender el uso del termino Free and Open Source Software (FOSS) el cual se utiliza para referirse al software que se adhiere al OSD y FSF.

A.2 Licencias de Software Libre

Mediante la *licencia* un autor permite el uso de su creación a otras personas. Es decir que la licencia es el instrumento que regula la forma en que el usuario puede utilizar el software. Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciatario. Desde el punto de vista del software *libre*, existen distintas variantes del concepto o grupos de licencias:

- *General Public License (GNU GPL)*: Es la licencia más usada, garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación. Esta licencia fue creada originalmente por Richard Stallman para el proyecto GNU (GNU project). Su finalidad es proteger los derechos de los usuarios finales (usar, compartir, estudiar, modificar). Esta es la primera licencia copyleft para uso general. Copyleft significa que los trabajos derivados sólo pueden distribuirse bajo los términos de la misma licencia. Bajo esta filosofía, la licencia GPL garantiza a los destinatarios de un programa los derechos-libertades reunidos en la definición de software libre y usa copyleft para asegurar que el software está protegido cada vez que el trabajo es distribuido, modificado o ampliado.

  El software bajo licencia GPL puede aplicarse bajo todos los propósitos. Esto incluye los propósitos comerciales e incluso el uso como herramienta de creación de software propietario. En uso puramente privativo o interno, sin venta ni distribución, no es necesario hacer publico el código fuente para los usuarios. Sólo si un programa utiliza fragmentos de código GPL y el programa es distribuido, el código fuente en su totalidad debe estar disponible bajo la misma licencia. Lo

mencionado es independiente de la plataforma utilizada.

Los usuarios o compañías que distribuyen sus trabajos bajo licencias GPL pueden hacerlo gratuitamente o cobrar por ellos. En particular, la GPL establece explícitamente que las obras GPL se puede vender a cualquier precio. Esto se deriva de la FSF, la cual argumenta que no se debe restringir la distribución comercial del software incluyendo la redistribución. La FSF permite al público crear nuevas licencias basadas en la GPL siempre y cuando las licencias derivadas no utilicen GPL sin permiso. Sin embargo, esto último no se recomienda ya que tal licencia puede ser incompatible con la GPL.

- *Lesser General Public License (GNU LGPL)*: La licencia LGPL de software fue creada por la FSF para tener derechos menos restrictivos que GPL. Garantiza la libertad de compartir y modificar el software cubierto por ella y asegura que el software es libre para todos sus usuarios. Esta licencia permisiva se aplica a cualquier programa o trabajo que contenga una nota puesta por el propietario de los derechos del trabajo en la cual se establece que su trabajo puede distribuirse bajo los términos de está. El termino "programa", utilizado en lo subsecuente, se refiere a cualquier programa o trabajo original. Por otro lado, el "trabajo basado en el programa" se refiere al programa o cualquier trabajo derivado del mismo bajo la ley de derechos de autor. Esto es, un trabajo que contenga el programa o alguna porción de él, ya sea íntegra o con modificaciones o traducciones a otros idiomas. Otras actividades que no sean copia, distribución o modificación no están cubiertas en esta licencia y están fuera de su alcance. El acto de ejecutar el programa no está restringido y la salida de información del programa está cubierta sólo si su contenido constituye un trabajo basado en el programa, es independiente de si fue resultado de ejecutar el programa. Si esto es cierto o no depende de la función del programa. En resumen, si se tiene un programa que utiliza fragmentos de código LGPL no es necesario liberar el código de dicho programa.

- *Berkeley Software Distribution (BSD)*: La licencia BSD se denomina de esta forma porque se utiliza en gran cantidad del software distribuido junto a los sistemas operativos BSD. Es una licencia de software libre permisiva que tiene menos restricciones en comparación con otras como la GPL. La licencia BSD, al contrario que la GPL, permite el uso del código fuente en software no libre. El autor, bajo tal licencia, mantiene la protección de copyright únicamente para

la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados. Sin embargo, permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Esta licencia asegura verdadero software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software.

A.3 Desarrollos de Código Abierto

Tras la adopción creciente de software de código abierto en la década del 90, la organización OSI propuso que el software *libre*, como fue conocido en ese entonces, tenía un lugar en la industria comercial. El éxito del modelo sorprendió a muchas personas y demostró que era un modelo de desarrollo viable. El aumento en la popularidad y utilidad de Internet proporciono el inicio para las grandes comunidades de código abierto.

Con el éxito del software *libre* como GNU/Linux, Servidor HTTP Apache, Mozilla Firefox y OpenOffice.org, muchas empresas comenzaron a interactuar con la comunidad del software *libre*. Esto trajo aparejado una nueva dificultad a la hora de elegir las licencias de software libre y en la selección de qué software se liberará. Un ejemplo de una relativamente exitosa entrada a la comunidad del software libre fue que "Sun Microsystems" liberó StarOffice como OpenOffice.org bajo la GNU LGPL. Esto fue muy bien recibido por la comunidad de software *libre* que no tenía una suite ofimática madura en ese momento. Un ejemplo de una entrada más difícil para la comunidad del software libre es el de Real Networks, qué escribió su propia licencia, y puso en libertad sólo una parte de su suite de software. En particular, el códec de los programas informáticos necesarios para ver archivos Real Video, no fue puesto en libertad.

El sitio web *OpenCores* con sus comienzos en el año 2000, proporciona un sitio para la comunidad de hardware de código abierto. Fue una de las primeras comunidades de desarrollo de hardware y actualmente la más grande, con más de cien mil usuarios y cerca de mil proyectos. El principal objetivo de OpenCores es diseñar y publicar diseños de núcleos bajo una licencia de hardware de código abierto siguiendo el modelo de Licencia LGPL usada para el software.

A.3.1 Código abierto en la industria

Actualmente exciten un gran número de aplicaciones de código abierto. Por ejemplo se dispone de software multimedia, productividad de oficina, herramientas de gráficos, sistemas operativos y de comunicaciones. Los gobiernos y las grandes empresas están aprovechando cada vez más el software de código abierto que existe y se están convirtiendo en importantes contribuyentes a los proyectos del software que adoptan. Las tendencias recientes han hecho mucho para disipar la imagen de solitarios desarrolladores de código abierto como los únicos contribuyentes de trabajos. Al estar las entidades comerciales aumentando la adopción y utilización de los proyectos de software libre, se están convirtiendo en los contribuyentes más frecuentes. Actualmente el kernel de Linux tiene la mayor parte de sus contribuciones de código proveniente de entidades comerciales. Esto se debe en gran parte a que están trabajando con el núcleo Linux en sus productos o porque desean asegurar el soporte de su hardware en el kernel. Entre dichas empresas se pueden mencionar a Intel, IBM y AMD. Lo mismo pasa para proyectos como Apache y MySQL.

La gran cantidad de software de código abierto disponible ofrece la adopción de licencias FOSS con la posibilidad de elegir entre: la publicación del trabajo derivado, el desarrollo de una solución interna o la compra de una solución propietaria manteniendo el código en privado.

Las entidades comerciales interesadas en mantener sus plataformas en buenas condiciones son en gran parte las que ofrecen mayor soporte y desarrollo a herramientas de programación como el compilador GCC y binutils del proyecto GNU. El desarrollo de las herramientas de software de código abierto es impulsado por GNU brindando soporte a diferentes plataformas con el fin de fomentar el uso de un compilador-optimizador de clase global, que atraiga a desarrolladores entregando herramientas que funcionen en diferentes arquitecturas.

El aumento del uso comercial del software de código abierto trae como consecuencia que las empresas aporten recursos a estos proyectos, disminuyendo por consiguiente el desarrollo y mantenimiento de sus propios conjuntos de compiladores y herramientas. Por ejemplo en el caso del GCC. Uno de los objetivos del código abierto es proporcionar una base de software donde los desarrolladores se encuentren cómodos para adoptarlo. Esto se debe al aumento del software existente producto de la publicación de trabajos derivados, lo que evita la tarea de empezar a desarrollar desde cero o comprar una solución propietaria. Estas son solo algunas de las motivaciones para la adopción y contribución de código abierto, pero podrían ser muchas más y variadas dependiendo

de la funcionalidad del proyecto.

Para las entidades comerciales la motivación de desarrollar código abierto proviene de cobrar por servicios extras al proyecto. Normalmente se requiere de conocimientos técnicos y experiencia para implementar y/o personalizar un proyecto en particular. Las empresas que adoptan y mantienen desarrollos de código abierto no tiene el costo de regalías o licencias, solo el de conocimiento necesarios para hacerlo.

A pesar de que el software de código abierto no es tradicionalmente un generador continuo de productos innovadores, no se opone a la estrategia de desarrollo elegida para una tecnología innovadora. Cualquier diseño innovador de código abierto suele tener retornos valiosos para la inversión requerida. Es una opción interesante para los desarrolladores empezar con una base que les ahorre tiempo y les permita competir frente a implementaciones propietarias, además de proporcionar soporte técnico a largo plazo para una aplicación.

Cuando el ciclo de vida de un producto llega a su fin, liberar el código fuente permite a otros desarrolladores aportar conocimiento brindando soluciones a problemas derivados de las actualizaciones inevitable de las plataformas.

Esta breve mirada a la situación actual de código abierto ha indicado que el mismo se ha convertido en una fuerza a tener en cuenta en el mundo del software. Sin embargo, este no es el caso en la actualidad para el desarrollo de hardware de código abierto.

A.3.2 Desventajas de la adopción del código abierto en software

Son muchos los que no están de acuerdo con el desarrollo de código abierto en la actualidad a pesar del gran aumento en su popularidad y uso. El software o hardware privativo puede ser desplazado del mercado por una alternativa no privativa, sin costo. Los que obviamente van a estar en desacuerdo son las entidades que se encuentren amenazadas por una alternativa lo suficientemente innovadora que provoque una disminución en sus ganancias. Una alternativa para los desarrolladores menos favorecidos es mejorar el código abierto existente el cual generalmente no es suficientemente innovador. Esto les permite competir con cualquier producto del mercado al mismo tiempo que adquieren conocimientos, ahorran tiempo y desarrollan un negocio. Además, tal mejora contribuye a mejorar el proyecto original.

El inconveniente de mostrar el código del proyecto mejorado es que los desarrolladores exponen sus técnicas innovadoras. Sin embargo, esto se compensado por la

reducción de la inversión en el desarrollo de su producto. Por otro lado, la mejora introducida en el proyecto de código abierto atrae a otros contribuyentes a trabajar en él.

Las implementaciones de código abierto varían en calidad y funcionalidad dependiendo de la colaboración de los desarrolladores. No hay dudas de que hay software de código abierto de gran utilidad. Sin embargo, no debe ser visto sólo como un producto innovador sino también como un paso en el camino hacia el avance tecnológico.

A.4 Código Libre para Hardware

El OpenRisc y otros IP publicados en OpenCore no sólo muestran el estado del proyecto sino que también permiten a los desarrolladores poder continuar el desarrollo de los núcleos. La aceptación de los proyectos de código abierto en RTL por parte de las entidades comerciales de IP no es la misma que para el desarrollo de software de código abierto.

El problema en la implementación de los proyectos de hardware de código abierto es que la falta de herramientas de "Entorno de Desarrollo Integrado" (IDE) es inaceptable para desarrollos de ASIC dado el elevado costo de corregir errores. Para brindar una solución a este problema se tendría que incluir dicha herramienta junto con el core. Las herramientas que se encuentran disponibles en la industria tienen un elevado precio y pueden variar de acuerdo al proveedor. Esto representa un gran obstáculo para la entrada al mercado de los desarrolladores de IP.

La falta de alternativas libres en EDA de código abierto limita al desarrollo de núcleos de código abierto. Esto se debe en parte a que la tecnología utilizada para el desarrollo de hardware se considera secreto industrial y esto limita el desarrollo de herramientas libres. Por tal motivo, actualmente, para minimizar el riesgo las implementaciones basadas en hardware de código abierto se realizan en dispositivos como FPGA donde generalmente se pueden hacer cambios del código RTL a un menor costo.

A.4.1 Problema para implementar y desarrollar hardware de código abierto

La necesidad de una plataforma donde implementar el diseño de hardware y la falta de herramientas de desarrollo libres son un gran obstáculo que no enfrenta el software

de código abierto. Estas plataformas se comercializan por empresas de hardware ofreciendo herramientas de compilación, simulación, síntesis y descarga para sus FPGA. La complejidad de las herramientas de diseño y la pronunciada curva de aprendizaje perjudica a los principiantes.

Otra dificultad a la hora de realizar un diseño para FPGA es que el mismo se debe describir a un nivel de abstracción muy bajo. Para lograr un resultado "útil" se requiere por lo general un gran trabajo que atraviese varios niveles de abstracción para poder utilizarlo cómodamente en una computadora.

Como ejemplo de una aplicación en hardware podría ser el desarrollo de un core que realice transacciones de I/O de un sensor. Dicho core podría proporcionar información a una aplicación que controle la temperatura en un espacio determinado. Esto requeriría el desarrollo y prueba del modelo de hardware y la implementación en FPGA. Por ejemplo, sea un microprocesador el que se encuentra corriendo sobre la FPGA, el cual brinda los servicios de red a través de un sistema operativo de tiempo real (RTOS). Este módulo personalizado debería requerir el desarrollo de una capa de software, lo que significa la necesidad de un driver para que permita al sistema operativo en tiempo real interactuar con el periférico, haciendo una abstracción del hardware y proporcionando una interfaz para usarlo. El sistema operativo en tiempo real se conecta con la aplicación que se ejecuta en el microprocesador de la FPGA para proporcionan lo datos a través del enlace de red. De esta forma, los datos de este sensor se encontrarán disponibles para la aplicación de nivel superior. Este es sólo un ejemplo en el cual el diseñador posiblemente haya seleccionado una solución que utilice un bus estándar. Sin embargo, existen algunos casos donde el diseñador desarrolla nuevos cores de interfaces o controladores en FPGA para proveer acceso a sistemas heredados (legacy) o nuevos buses estándar, donde vale destacar el trabajo extra requerido al escribir código RTL que provea la interfaz física. Viendo la cantidad de desarrollo y pruebas requeridas para poner en práctica estas soluciones, es fácil sentirse abrumado por la cantidad de trabajo necesario para completar una tarea tan aparentemente trivial.

Al comparar esto último con la implementación de software de código abierto. En el segundo, simplemente se descarga el código fuente, se lo compila, se lo ejecuta y se lo prueba todo usando la misma plataforma. Hay una gran ventaja en la facilidad con que se accede la plataforma de desarrollo (el host) y las herramientas de desarrollo son mucho más simples (gcc, make en el sistema host). Además, las pruebas son más cortas y más fáciles (se ejecutan en el equipo host a través de un shell).

A medida que se incremente la cantidad de proyectos de hardware de código abierto y disminuya la complejidad de las herramientas de desarrollo es de esperar que se superen los obstáculos mencionados. Las barreras iniciales a las que se enfrentó el software de código abierto parecían igual de complicadas de superar. Es de esperar que con el tiempo y el aumento de participantes, el hardware de código abierto alcance el mismo éxito. En ese momento, los diseños serán tan grandes que no cabrán en los dispositivos programables actuales. La reconfiguración será un elemento imprescindible. Las fronteras entre el hardware y el software se harán cada vez más difusas.

A.5 Licencias de Hardware

Una cuestión que queda por resolver es la concesión de licencias para el diseño de hardware de código abierto. Por ejemplo, el proyecto OpenRISC utiliza licencias públicas del proyecto GNU. Sin embargo, estas se refieren específicamente a software y no se sabe bien que se aplica al hardware. El sitio web del proyecto GNU contiene una sección con preguntas frecuentes (FAQ) que afirma lo siguiente.

Cualquier material que puede ser licenciado con derechos de autor puede ser licenciado bajo la GPL.

Una indicación de la naciente idea del desarrollo de hardware de código abierto proviene de la publicación reciente (febrero de 2011) de un conjunto de principios para los participantes de la comunidad de hardware de código abierto. El siguiente es el código abierto Hardware (OSHW) Declaración de Principios 1.0 de FreedomDefined.org. Estas publicaciones proporcionan un punto de referencia para saber si el diseño puede estar bajo licencia de "hardware de código abierto".

Cualquier licencia de hardware de código abierto se puede utilizar para restringir (o en este caso, para no restringir) los planos de un diseño, pero no el uso del dispositivo fabricado. Estos son conceptos que ya se encuentran a menudo en las licencias de software de código abierto, pero de nuevo, no es tan claro como se aplica para el diseño del hardware de código abierto.

Una de las primeras licencias de hardware de código abierto es la Amateur Packet Radio Licencia Open Hardware Tucson (TAPR OHL). Los autores de TAPR OHL identifican el problema con las licencias de software existentes. En las mismas, los derechos de autor protegen la documentación de copias, modificaciones y distribuciones, pero esto tiene poco que ver con el derecho de hacer, distribuir o usar un producto basado en la documentación [40]. En consecuencia, la TAPR OHL esta

siendo adoptada por varios aficionados y empresas comerciales.

En general, las licencias actuales de hardware de código abierto han recibido críticas del OSI, entre otras cosas, por la adopción de un significado diferente de la palabra "distribución" [41]. Sin embargo, es de esperar que en el futuro próximo surjan licencias alternativas de hardware de código abierto que resuelvan las ambigüedades presentes en las licencias actuales.

Anexo B

Esquemáticos

En sucesivas páginas se muestran los esquemáticos completos para las tres placas utilizadas en el proyecto. Las hojas aparecen en el mismo orden del siguiente listado que explica brevemente cada esquemático:

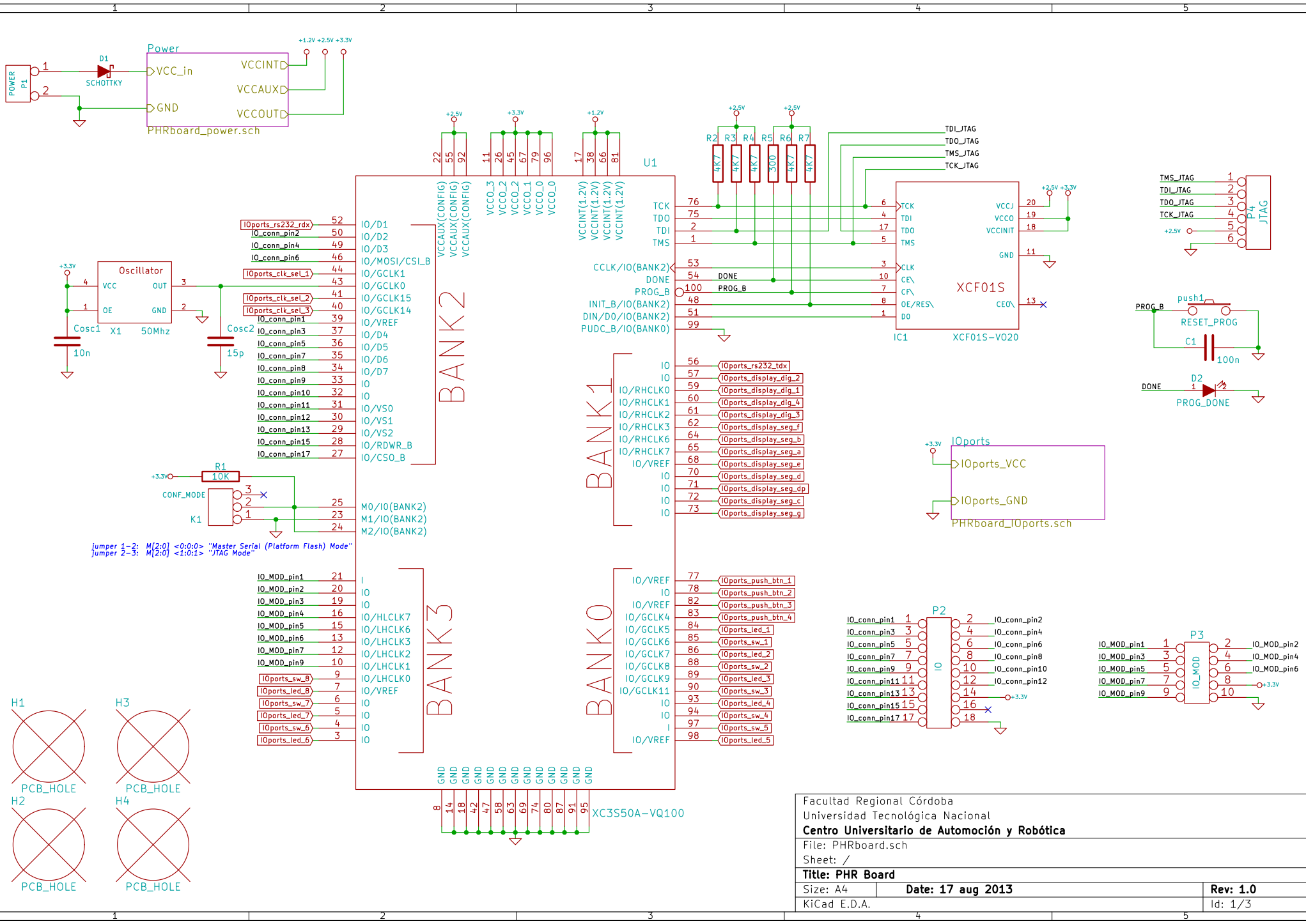
PHRboard.sch Muestra el chip FPGA y la conexión de sus pines.

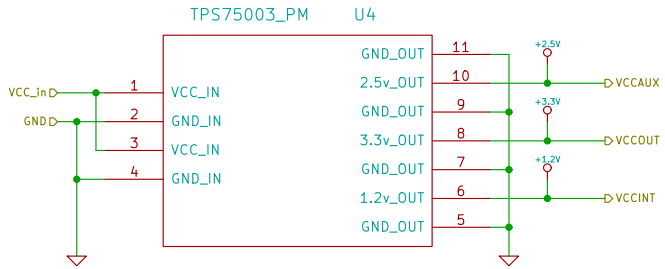
PHRboard_power.sch Muestra la red de capacitores de desacople y bypass.

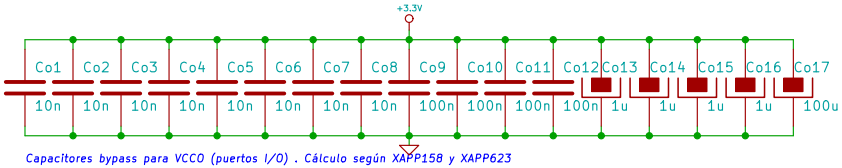
PHRboard_IOports.sch En esta hoja se describen mayormente los perífericos de la placa PHR.

OOCD_placa.sch Es la placa *OOCDLink* encargada de la comunicación con la PC.

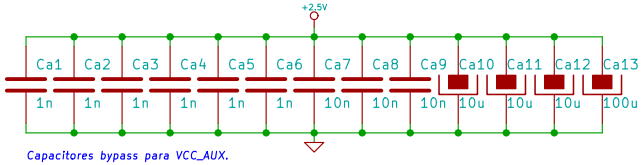
S3Proto_Power.sch Es la placa *S3Power* que provee la energía para el resto de los circuitos.


Power
PHRboard_power.sch
SCHOTTKY
VCC_in
GND
VCCINT
VCCAUX
VCCOUT
U1
BANK2
BANK1
BANK3
BANK0
XC3S50A-VQ100
Oscillator
50Mhz
Cosc1
Cosc2
10n
15p
X1
R1
10K
CONF_MODE
K1
jumper 1-2: M[2:0] <0:0:0> "Master Serial (Platform Flash) Mode"
jumper 2-3: M[2:0] <1:0:1> "JTAG Mode"
XCF01S
XCF01S-VO20
IC1
TDI_JTAG
TDO_JTAG
TMS_JTAG
TCK_JTAG
JTAG
RESET_PROG
PROG_DONE
C1
100n
IOports
PHRboard_IOports.sch
IOports_VCC
IOports_GND
P2
P3
IO_MOD
H1
H2
H3
H4
PCB_HOLE
Facultad Regional Córdoba
Universidad Tecnológica Nacional
Centro Universitario de Automoción y Robótica
File: PHRboard.sch
Sheet: /
Title: PHR Board
Size: A4
Date: 17 aug 2013
Rev: 1.0
KiCad E.D.A.
Id: 1/3

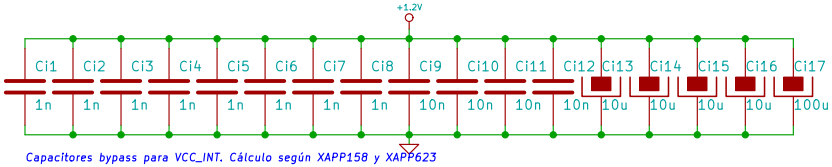




Capacitores bypass para VCCO (puertos I/O) . Cálculo según XAPP158 y XAPP623

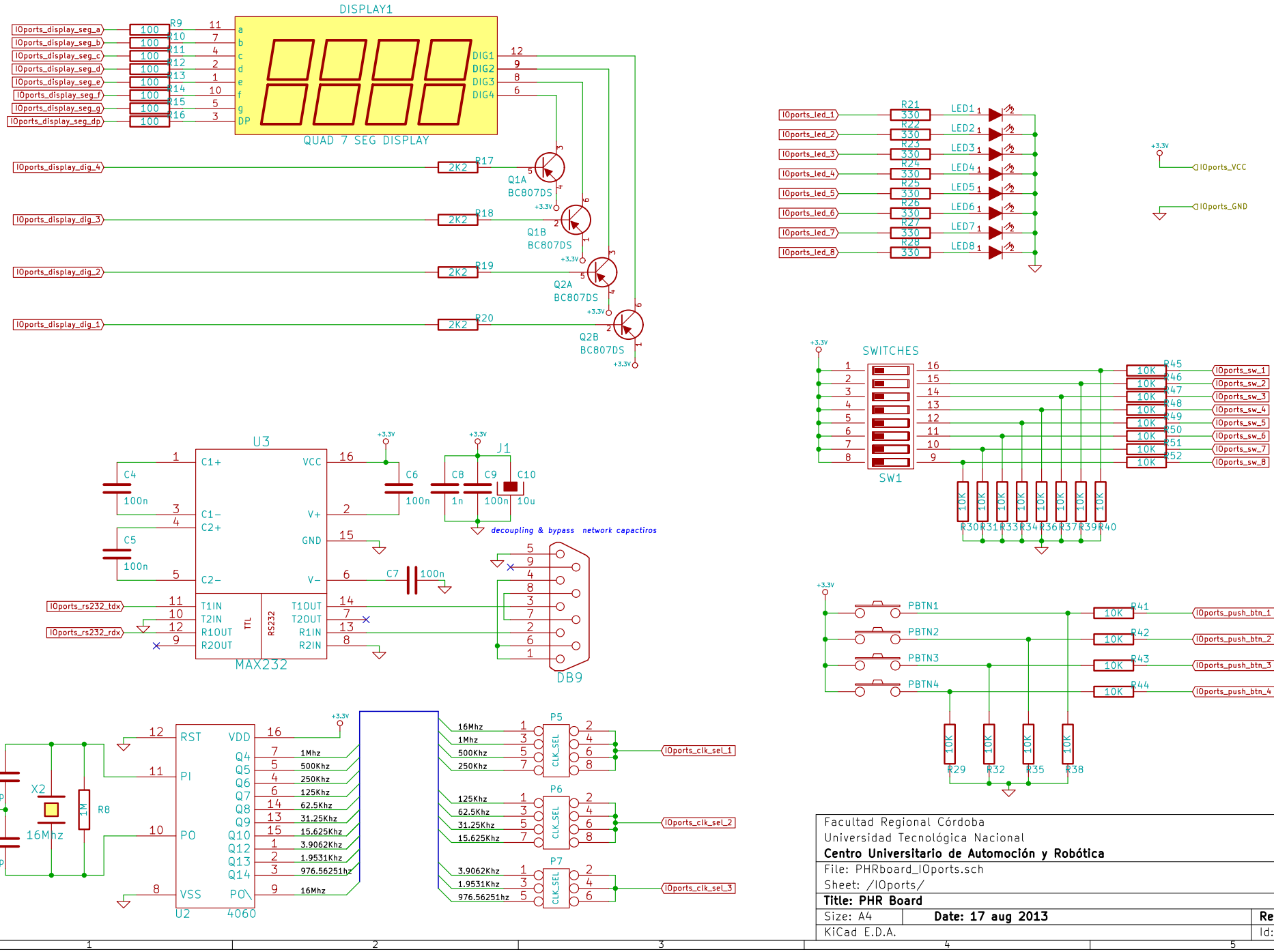


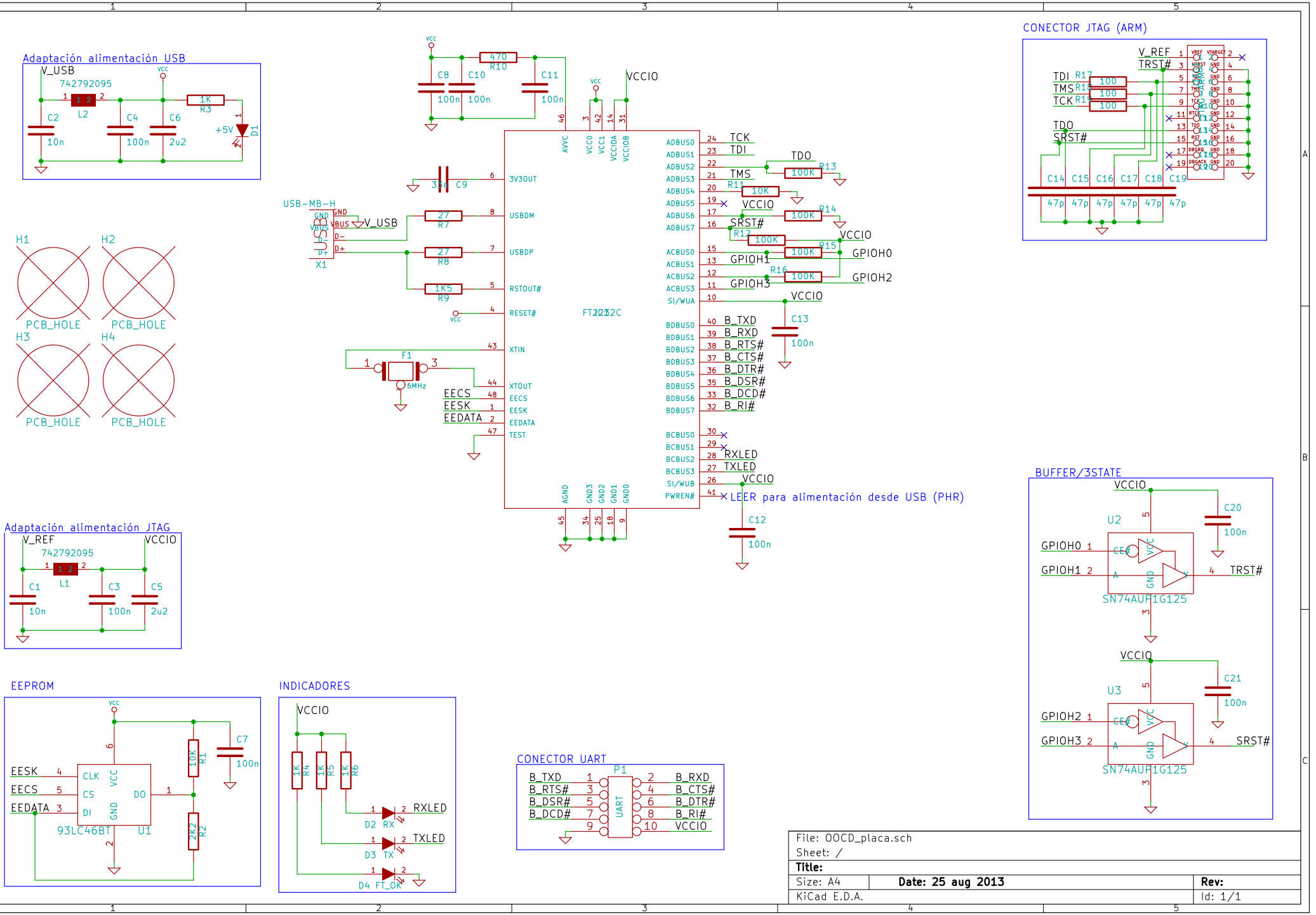
Capacitores bypass para VCC_AUX.

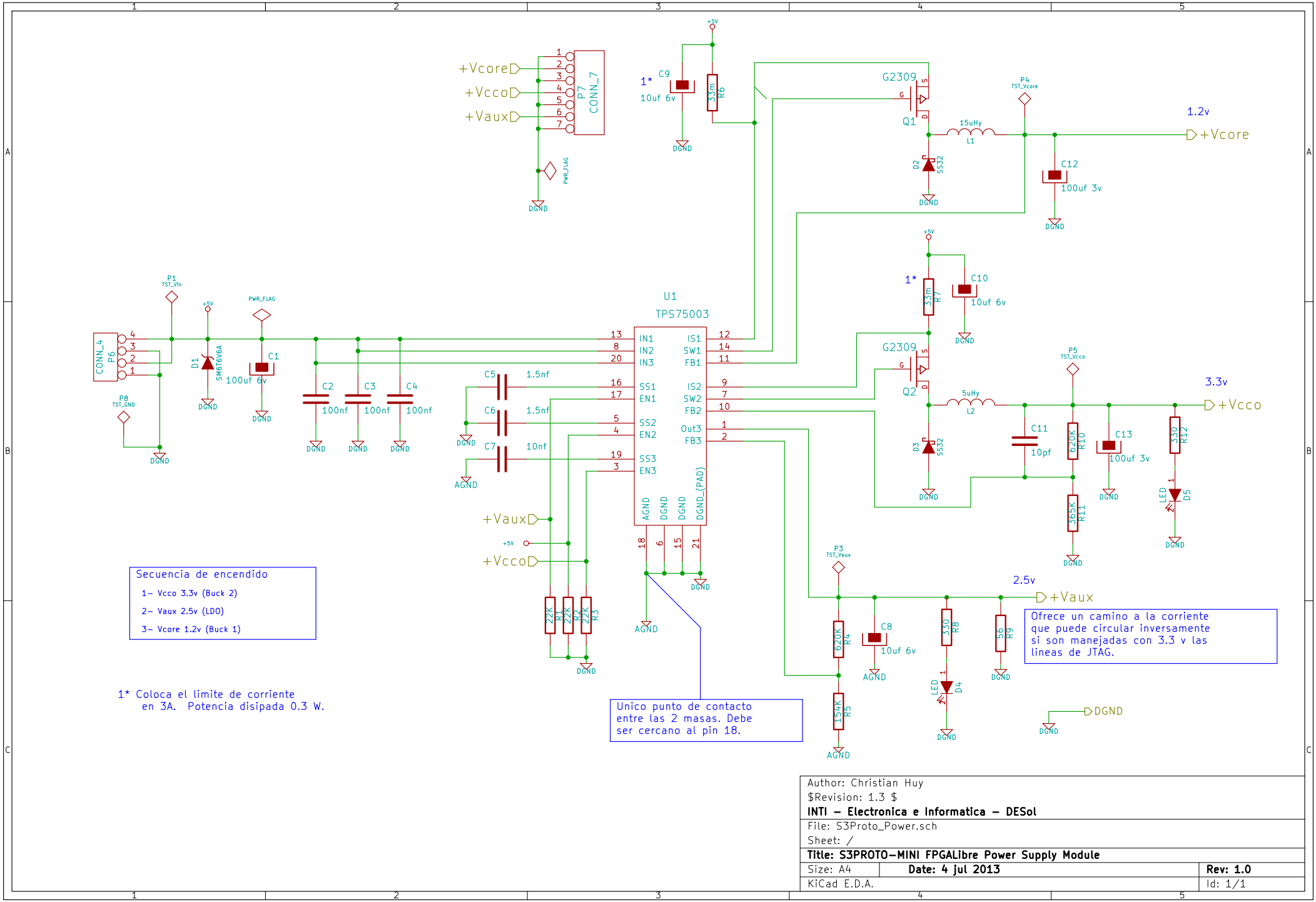


Capacitores bypass para VCC_INT. Cálculo según XAPP158 y XAPP623

File: PHRboard_power.sch		
Sheet: /Power/		
Title:		
Size: A4	**Date: 17 aug 2013**	**Rev:**
KiCad E.D.A.		Id: 2/3


DISPLAY1
QUAD 7 SEG DISPLAY
IOports_display_seg_a
IOports_display_seg_b
IOports_display_seg_c
IOports_display_seg_d
IOports_display_seg_e
IOports_display_seg_f
IOports_display_seg_g
IOports_display_seg_dp
R9
R10
R11
R12
R13
R14
R15
R16
100
DIG1
DIG2
DIG3
DIG4
IOports_display_dig_4
IOports_display_dig_3
IOports_display_dig_2
IOports_display_dig_1
R17
R18
R19
R20
2K2
Q1A
Q1B
Q2A
Q2B
BC807DS
+3.3V
R21
R22
R23
R24
R25
R26
R27
R28
330
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
IOports_led_1
IOports_led_2
IOports_led_3
IOports_led_4
IOports_led_5
IOports_led_6
IOports_led_7
IOports_led_8
IOports_VCC
IOports_GND
U3
MAX232
C1+
C1-
C2+
C2-
VCC
V+
GND
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
TTL
RS232
C4
C5
C6
C7
C8
C9
C10
100n
1n
10u
J1
decoupling & bypass network capactiros
IOports_rs232_tdx
IOports_rs232_rdx
DB9
SWITCHES
SW1
R30
R31
R33
R34
R36
R37
R39
R40
R45
R46
R47
R48
R49
R50
R51
R52
10K
IOports_sw_1
IOports_sw_2
IOports_sw_3
IOports_sw_4
IOports_sw_5
IOports_sw_6
IOports_sw_7
IOports_sw_8
PBTN1
PBTN2
PBTN3
PBTN4
R29
R32
R35
R38
R41
R42
R43
R44
IOports_push_btn_1
IOports_push_btn_2
IOports_push_btn_3
IOports_push_btn_4
U2
4060
RST
PI
PO
VSS
VDD
PO
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q12
Q13
Q14
C2
C3
18p
X2
16Mhz
1M
R8
1Mhz
500Khz
250Khz
125Khz
62.5Khz
31.25Khz
15.625Khz
3.9062Khz
1.9531Khz
976.56251hz
16Mhz
P5
P6
P7
CLK_SEL
IOports_clk_sel_1
IOports_clk_sel_2
IOports_clk_sel_3
Facultad Regional Córdoba
Universidad Tecnológica Nacional
Centro Universitario de Automoción y Robótica
File: PHRboard_IOports.sch
Sheet: /IOports/
Title: PHR Board
Size: A4
Date: 17 aug 2013
Rev: 1.0
KiCad E.D.A.
Id: 3/3


Adaptación alimentación USB
V_USB
742792095
L2
C2
10n
C4
100n
C6
2u2
1K
R3
+5V
D1
H1
H2
H3
H4
PCB_HOLE
PCB_HOLE
PCB_HOLE
PCB_HOLE
Adaptación alimentación JTAG
V_REF
VCCIO
742792095
L1
C1
10n
C3
100n
C5
2u2
EEPROM
93LC46BT
U1
EESK
EECS
EEDATA
10K
R1
2K2
R2
C7
100n
INDICADORES
VCCIO
1K
R4
1K
R5
1K
R6
D2 RX
RXLED
D3 TX
TXLED
D4 FT_OK
CONECTOR UART
P1
UART
B_TXD
B_RTS#
B_DSR#
B_DCD#
B_RXD
B_CTS#
B_DTR#
B_RI#
VCCIO
USB-MB-H
V_USB
27
R7
27
R8
1K5
R9
X1
F1
6MHz
C8
100n
C10
100n
470
R10
C11
100n
C9
3n
FT2232C
3V3OUT
USBDM
USBDP
RSTOUT#
RESET#
XTIN
XTOUT
EECS
EESK
EEDATA
TEST
AVVC
VCC0
VCC1
VCCIOA
VCCIOB
AGND
GND3
GND2
GND1
GND0
ADBUS0
ADBUS1
ADBUS2
ADBUS3
ADBUS4
ADBUS5
ADBUS6
ADBUS7
ACBUS0
ACBUS1
ACBUS2
ACBUS3
SI/WUA
BDBUS0
BDBUS1
BDBUS2
BDBUS3
BDBUS4
BDBUS5
BDBUS6
BDBUS7
BCBUS0
BCBUS1
BCBUS2
BCBUS3
SI/WUB
PWREN#
TCK
TDI
TDO
TMS
SRST#
GPIOH0
GPIOH1
GPIOH2
GPIOH3
B_TXD
B_RXD
B_RTS#
B_CTS#
B_DTR#
B_DSR#
B_DCD#
B_RI#
RXLED
TXLED
LEER para alimentación desde USB (PHR)
R11
10K
R12
100K
R13
100K
R14
100K
R15
100K
R16
100K
C12
100n
C13
100n
CONECTOR JTAG (ARM)
V_REF
TRST#
TDI
TMS
TCK
TDO
SRST#
100
100
100
C14
C15
C16
C17
C18
C19
47p
47p
47p
47p
47p
47p
BUFFER/3STATE
U2
SN74AUP1G125
GPIOH0
GPIOH1
TRST#
C20
100n
U3
SN74AUP1G125
GPIOH2
GPIOH3
SRST#
C21
100n
File: OOCD_placa.sch
Sheet: /
Title:
Size: A4
Date: 25 aug 2013
Rev:
KiCad E.D.A.
Id: 1/1



Author: Christian Huy		
$Revision: 1.3 $		
INTI – Electronica e Informatica – DESol		
File: S3Proto_Power.sch		
Sheet: /		
Title: S3PROTO-MINI FPGALibre Power Supply Module		
Size: A4	**Date: 4 jul 2013**	**Rev: 1.0**
KiCad E.D.A.		Id: 1/1