



**UNIVERSIDAD MAYOR  
FACULTAD DE INGENIERIA**

**“DESARROLLO E IMPLEMENTACION DE SISTEMA DE COMUNICACION  
RAW SOCKET TCP/IP Y SERVIDOR HTML EMPOTRADOS, BASADOS EN  
PROGRAMACION DE TECNOLOGIA FPGA  
MODELO VIRTEX-5 XILINX.”**

**Proyecto de Título para Optar al Título de Ingeniero Civil Electrónico**

**NICOLAS EDUARDO PEÑA RALPH**

**SANTIAGO DE CHILE  
NOVIEMBRE 2010**



**UNIVERSIDAD MAYOR  
FACULTAD DE INGENIERIA**

**“DESARROLLO E IMPLEMENTACION DE SISTEMA DE COMUNICACION  
RAW SOCKET TCP/IP Y SERVIDOR HTML EMPOTRADOS, BASADOS EN  
PROGRAMACION DE TECNOLOGIA FPGA  
MODELO VIRTEX-5 XILINX.”**

**Proyecto de Título para Optar al Título de Ingeniero Civil Electrónico**

**Alumno: Nicolás Peña R.  
Profesor Guía: Gonzalo Téllez**

**SANTIAGO DE CHILE  
NOVIEMBRE 2010**

## **Dedicatoria**

A mi Padres, quienes me apoyaron constantemente.

## **Agradecimientos**

Quiero agradecer el apoyo incondicional prestado por personas importantes en mi vida, que me han acompañado por medio de su comprensión para acometer mis metas. Espero en alguna oportunidad devolver a Cristina Ralph y Hugo Peña, todo el afecto y la ayuda que como padres me han brindado.

Quiero agradecer también a mi profesor guía Gonzalo Téllez, debido que a pesar de algunas dificultades surgidas de su parte para atender mis consultas, estuvo siempre dispuesto a cooperar en este tema. También por su tiempo dedicado y esfuerzo.

# Índice de contenidos

i.	Índice de figuras.....	8
ii.	Resumen.....	10
iii.	Abstract.....	11
	Presentación General.....	12
1.1.	Introducción .....	12
1.2.	Alcance .....	14
1.3.	Objetivos Generales.....	15
1.4.	Objetivos Específicos.....	15
1.5.	Motivación del proyecto .....	16
1.6.	Aplicación en la Industria .....	17
1.7.	Tecnología utilizada .....	18
1.8.	Limitaciones .....	18
1.9.	Material de entrega .....	19
2.	Análisis del Problema.....	20
2.1.	Situación Inicial del Sistema .....	20
2.2.	Propuesta Técnica del Sistema .....	21
2.3.	Propuesta programación de Hardware .....	22
2.4.	Propuesta programación de Software.....	25
2.5.	Costos del Proyecto .....	26
3.	Desarrollo de Hardware .....	27
3.1.	Resumen de Actividades .....	27
3.2.	Selección de Módulos .....	31
3.3.	Comprensión funcional de Módulos.....	33
3.4.	Puertos o Señales de los Módulos.....	39

3.5.	Definir Arquitectura .....	43
3.6.	Integración Módulo / buses del sistema .....	44
3.7.	Ajuste de Memoria Principal .....	46
3.8.	Ajuste de Memoria Flash .....	51
3.9.	Memoria Block Ram.....	53
3.10.	Mapeo de Memoria del Procesador.....	54
3.11.	Implementación de Microblaze .....	55
3.12.	Frecuencias del Sistema .....	57
3.13.	Interrupciones.....	59
3.14.	Contador.....	60
3.15.	Reseteos .....	61
3.16.	Interconexión de módulos .....	62
3.17.	Puertos Externos.....	65
3.18.	Tarjeta de Red.....	68
4.	Desarrollo Software.....	70
4.1.	Resumen de Actividades .....	70
4.2.	Sistema Operativo.....	73
4.3.	Librerías .....	74
4.4.	Máscaras .....	75
4.5.	Drivers.....	77
4.6.	Cabeceras.....	78
4.7.	Contador .....	79
4.8.	Contador/LWIP .....	83
4.9.	Interrupciones .....	84
4.10.	IP/MAC .....	87
4.11.	Socket TCP/IP RAW.....	88
4.12.	Devolución de Llamados (Callbacks) .....	91
4.13.	HTML.....	93
4.14.	Función principal (Main) .....	94
5.	Pruebas.....	95

6.	Resultados .....	98
7.	Conclusiones.....	99
8.	Bibliografía .....	101
9.	Anexos .....	102
9.1.	Tarjeta Haps-51 .....	102
9.2.	Componentes Haps-51 – Cara superior.....	103
9.3.	Componentes Haps-51 – Cara inferior.....	104
9.4.	Especificaciones técnicas Haps-51.....	105
9.5.	Puerto Hapstrack II número A2.....	106
9.6.	Virtex-5 LX330 y otros modelos .....	107
9.7.	Tarjeta GEPHY y especificaciones técnicas .....	108
9.8.	Código de Archivo MHS.....	109
9.9.	Archivo UCF.....	121
9.10.	Servidor HTML Embebido .....	125
9.11.	Tiempos de Acceso módulo MCH EMC .....	125
9.12.	FPGAs en Sistemas Embebidos .....	126
9.13.	Lista de acrónimos .....	126

## i. Índice de figuras

<i>Figura 2-1 : Diagrama Tarjeta Haps-51 .....</i>	20
<i>Figura 2-2 : Propuesta Hardware del Sistema.....</i>	21
<i>Figura 2-3 : Propuesta de Hardware VHDL.....</i>	22
<i>Figura 2-4 : Propuesta Software .....</i>	25
<i>Figura 3-1 : Actividades Hardware.....</i>	30
<i>Figura 3-2 : Módulos VHDL incorporados .....</i>	32
<i>Figura 3-3 : Esquemático de módulo controlador de memoria principal .....</i>	35
<i>Figura 3-4 : Esquemático de módulo generador de reloj.....</i>	35
<i>Figura 3-5 : Esquemático de módulo controlador de memoria flash y ssram....</i>	37
<i>Figura 3-6 : Puertos disponibles para Microblaze .....</i>	40
<i>Figura 3-7 : Puertos disponibles para controlador de memoria principal .....</i>	40
<i>Figura 3-8 : Puertos disponibles para controlador de memoria flash y ssram... </i>	41
<i>Figura 3-9 : Puertos disponibles para generador de reloj.....</i>	41
<i>Figura 3-10 : Puertos disponibles para controlador de resets .....</i>	41
<i>Figura 3-11 : Puertos disponibles para controlador de interrupciones .....</i>	41
<i>Figura 3-12 : Puertos disponibles de controlador Ethernet.....</i>	42
<i>Figura 3-13 : Puertos disponibles interfaz de block ram .....</i>	42
<i>Figura 3-14 : Puertos disponibles modulo programador .....</i>	42
<i>Figura 3-15 : Buses de comunicación con Microblaze .....</i>	43
<i>Figura 3-16 : Buses de comunicación con el Sistema.....</i>	45
<i>Figura 3-17 : Parámetros de controlador de memoria principal.....</i>	48
<i>Figura 3-18 : Implementación de puertos controladores de memoria principal. </i>	50
<i>Figura 3-19 : Implementación de latencias de memoria principal .....</i>	51
<i>Figura 3-20 : Ajuste de controlador de memoria flash.....</i>	53
<i>Figura 3-21 : Mapa de memoria del procesador.....</i>	55
<i>Figura 3-22 : Esquemático del módulo Microblaze .....</i>	56
<i>Figura 3-23 : Tabla de frecuencias del módulo administrador de reloj.....</i>	58
<i>Figura 3-24 : Tabla de interrupciones.....</i>	59
<i>Figura 3-25 : Parámetros de operación del contador .....</i>	61
<i>Figura 3-26 : Parámetros de operación del controlador de resets.....</i>	62
<i>Figura 3-27 : Conexión intermodular 1.....</i>	63

<i>Figura 3-28 : Conexión intermodular 2</i> .....	63
<i>Figura 3-29 : Conexión intermodular 3</i> .....	64
<i>Figura 3-30 : Conexión intermodular 4</i> .....	64
<i>Figura 3-31 : Puertos externos</i> .....	65
<i>Figura 3-32 : Extracto de archivo .ucf</i> .....	67
<i>Figura 3-33 : Puertos Haps-51</i> .....	68
<i>Figura 3-34 : GEPHY</i> .....	69
<i>Figura 4-1 : Diagrama de actividades de software</i> .....	70
<i>Figura 4-2 : Librerías disponibles para Microblaze</i> .....	74
<i>Figura 4-3 : Xparameters interrupciones</i> .....	76
<i>Figura 4-4 : Xparameters contador</i> .....	76
<i>Figura 4-5 : Xparameters memoria principal</i> .....	76
<i>Figura 4-6 : .MSS de contador, Ethernet e Interrupciones</i> .....	77
<i>Figura 4-7 : Archivos cabecera incluidos</i> .....	78
<i>Figura 4-8 : Máscaras de configuración de contador</i> .....	80
<i>Figura 4-9 : Función Timer Handler</i> .....	82
<i>Figura 4-10 : Función platform setup interrupts</i> .....	86
<i>Figura 4-11 : IP / MAC</i> .....	87
<i>Figura 4-12 : Función seteo del socket</i> .....	90
<i>Figura 4-13 : Devolución de llamados y Primitivas</i> .....	91
<i>Figura 4-14 : Función HTTP ACCEPT</i> .....	92
<i>Figura 4-15 : Función HTTP RECV</i> .....	92
<i>Figura 4-16 : Contenido HTML</i> .....	93
<i>Figura 4-17 : Función Main</i> .....	94
<i>Figura 5-1: Configuración de Prueba</i> .....	96
<i>Figura 9-1: Tarjeta Haps-51</i> .....	102
<i>Figura 9-2: Detalles de Haps-51 (cara superior)</i> .....	103
<i>Figura 9-3: Detalles Haps-51 (cara inferior)</i> .....	104
<i>Figura 9-4: Pines Hapstrack II (Bus A2)</i> .....	106
<i>Figura 9-5: Familia Virtex-5</i> .....	107
<i>Figura 9-6: Archivo .UCF</i> .....	124
<i>Figura 9-7: Prototipo Servidor HTML Embebido</i> .....	125
<i>Figura 9-8: FPGA en Sistemas Embebidos</i> .....	126

## ii. Resumen

Un FPGA es un dispositivo de lógica programable cuya función digital se determina por el usuario. Son utilizados para reemplazar circuitos integrados de bajo a medio nivel de integración con el propósito de mejorar la implementación y el escalamiento.

Un proveedor excepcional de FPGAs es Xilinx, fabricante precursor de estos chips, donde debido a su prestigio, calidad y compromiso, se ubica dentro de los favoritos al momento de seleccionar uno de estos dispositivos.

El objetivo de este proyecto consiste en desarrollar un prototipo de servidor embebido HTML programado en un chip FPGA de Xilinx, sustentado en TCP/IP Ethernet. El dispositivo puede ser utilizado posteriormente para supervisar, mediante acceso remoto, variables de interés mediante señales digitales a la entrada de la tarjeta, siempre y cuando se realicen las adaptaciones que requiera el sistema.

La tecnología empleada es un chip FGPA Virtex-5 que se presenta en una tarjeta de desarrollo HAPS-51. Esta tarjeta cumple las necesidades básicas de programación y entre otros aspectos cuenta con memorias, puertos de entrada/salida de datos y reloj de sistema. Además se utilizó una tarjeta adicional para complementar la interfaz Ethernet.

La metodología usada consiste en desarrollar un Firmware donde es necesario programar la lógica que cubre aspectos tales como incorporación de un procesador, buses, memoria principal, contadores y controlador de interrupciones entre otros. Posteriormente se debe programar el Software que da sustento a las comunicaciones por medio de sockets, controlando también las interrupciones y los periféricos.

### **iii. Abstract**

An FPGA is a programmable logic device which digital function is configured by the user. It is used to replace small-scale and medium-scale integrated circuits to enhance implementation and growth.

An exceptional FPGA provider is Xilinx, first researcher and developer of those microchips. Related within its prestige, quality and compromise is one of the most favorite and preferred FPGA device for buyers.

The objective of the project is to develop an embedded HTML server prototype, programmed into a FPGA Xilinx microchip running with TCP/IP Ethernet. The device could be further solved as a remote supervisor, that perform checking outside-system variables. The variables are guided to the microchip entrance gates as digital signals while the user must have fitted that correctly. The fitting-out prototype of any industrial plant is out of the purpose of the project.

FPGA Virtex-5 is the technology welded on the HAPS-51 mother development board. This mother board satisfied the basic programming requirements, and also come with memories, both input-output data ports and a clock system reference. In addition, it was selected a daughter board to provide Ethernet interface.

The project method is about a Firmware development. First, it is necessary to program logic structures related with microprocessor, buses, main memory, counters, and an interruption controller among others. Then, the software must be engineered to support socket communications, controlling interrupts and peripherals.

# Presentación General

## 1.1. *Introducción*

Los sistemas embebidos han sido evaluados en diversos países respecto de su uso en la industria y en personas, lo que ha demostrado que tal cantidad ha ido aumentando con el paso del tiempo y seguirá por la misma senda a lo largo de los próximos años. Se pronostica además, que cobrarán importancia en esto las comunicaciones.

Debido a esta tendencia, ha surgido la preocupación e interés por estos sistemas modernos, y tanto en países desarrollados como emergentes se traduce en destinar mayores recursos económicos para el estudio de estas tecnologías y capacitación de personal indicado.

Las FPGAs<sup>1</sup>, son uno de los dispositivos empleados para implementación de sistemas embebidos. Estos se ubican dentro la última generación de PLDs que existen hoy en día. La compañía Xilinx es precursora en crear estos elementos y es favorita en cuanto al gran nivel de demanda de estos recursos.

A raíz de esta información surge la motivación del proyecto, cuyo objetivo comprende el desarrollo de un sistema embebido basado en una FPGA de Xilinx, que cumplirá la función de proveer comunicación mediante un servidor prototipo. El servidor brindará un servicio de envío de datos de hipertexto, al que tendrán acceso los clientes que accedan al dispositivo.

Los clientes son computadoras convencionales, los que a través de un navegador web solicitan al servidor del proyecto el contenido HTML que es un archivo. La solicitud se realiza ingresando la dirección IP asignada a la interfaz Ethernet del prototipo, en el navegador preferido. Finalmente el cliente al recibir el archivo, interpreta el código mostrando el resultado en su interfaz gráfica.

---

<sup>1</sup> Consultar lista de acrónimos

La metodología usada consta de dos etapas. La primera es la descripción de lógica tal como procesadores, buses, contadores y controladores de periféricos entre otros, cuya finalidad es emular la estructura de un computador ajustado a las necesidades del servidor embebido. La segunda es programar el software tal como los sockets, el manejo de interrupciones y los controladores entre otros. Ambas partes en conjunto conformarán lo que se llama Firmware.

La elaboración del Firmware que cumpla este objetivo es una tarea compleja. Para eso, las tarjetas de desarrollo, en especial de FPGAs o bien cualquier otro dispositivo electrónico programable, brindan la ayuda necesaria para la programación e implementación rápida y fiable de funciones determinadas.

En este proyecto, se cuenta con la tarjeta HAPS-51, la cual ahorra tiempo y trabajo en problemas particulares como diseño de la placa, soldado, programación del chip, incorporación de tarjetas adicionales, memorias, y reloj de sincronismo, en donde tales ventajas ayudarán a acometer eficientemente el objetivo.

La HAPS-51 incorpora una FPGA Virtex-5 Xilinx, modelo de alta tecnología, versátil y de una gran capacidad para generar lógica y software. Es en este chip donde estará contenida la información que creará al prototipo servidor HTML.

## 1.2. Alcance

Este proyecto proporcionará un prototipo de comunicación TCP/IP y un servidor HTML, para un chip FPGA de la marca Xilinx familia Virtex-5, integrado en una tarjeta modelo HAPS-51 de la marca Simplicity.

La documentación necesaria para el desarrollo de este proyecto se compone de los siguientes elementos:

- Guía de usuario de chip FPGA familia Virtex-5 (*Fuente: UG Xilinx.com*)
- Manual de configuración de la tarjeta HAPS-51 (*Fuente: Symplicity.com*)
- Hoja de datos de HAPS GEPHY 1x1 (*Fuente: DS GEPHY1x1*)
- Hoja de datos de memoria FLASH (*Fuente: EM39LV040*)
- Hoja de datos de memoria DDR2 SDRAM (*Fuente: HYS72T128020GR*)
- Hoja de datos de memoria SSRAM (*Fuente: K9K1208U0M-YCB0*)
- Guía de usuario de IPCores de Xilinx (*Fuente: Xilinx.com*)

El software que será utilizado para el desarrollo e implementación será el siguiente:

- Xilinx Platform Studio, XPS (*Desarrollo de Proyecto Embebido*)
- Xilinx Impact (*Software que programa la FPGA con archivo binario*)
- Eclipse Galileo (*Desarrollo de software C*)
- Mozilla FireFox (*Navegador Web*)

Cabe mencionar que se han creado previamente servidores embebidos prototipos. Sin embargo, esto ha sido utilizando otros modelos de FPGAs, lo que para un modelo Xilinx Virtex-5 bajo una tarjeta de desarrollo HAPS-51, este es un proyecto completamente original. Además es original en cuanto a que incorpora un servicio fundamentalmente de tipo HTML.

### **1.3. Objetivos Generales**

El objetivo general de este proyecto es:

- Desarrollar un servidor HTML prototipo, basado en comunicación TCP/IP Ethernet y embebido sobre un chip tipo FPGA Virtex-5.

### **1.4. Objetivos Específicos**

Los objetivos específicos del proyecto son:

- Construir un prototipo de servidor HTML basado en comunicaciones TCP/IP Ethernet, embebido en una FPGA Virtex-5 LX330 de una tarjeta HAPS-51 Simplicity.
- Entregar a los ingenieros o técnicos que desarrollen Firmware para sistemas embebidos, conocimientos para implementar una comunicación basada en sockets TCP/IP, mediante tecnología FPGA de Xilinx.
- Brindar conocimientos para el desarrollo de un proyecto Firmware para la FPGA Virtex-5 de Xilinx, tal que permita programar hardware adecuado, y manejo de proyectos de software.

## **1.5. Motivación del proyecto**

La motivación del proyecto surge a partir de:

- Tendencia en uso de Sistemas Embebidos

Este proyecto se adapta a la tendencia actual de innovación y desarrollo de sistemas embebidos<sup>1</sup>, quedando como base para una posible aplicación en la industria. Se calcula que el mercado mundial de estos sistemas tendrá un valor en el mundo durante 2020 de mucha inversión en recursos. Actualmente ya representa el 14% de la inversión en I+D en Europa.

- Tendencia en uso de las Comunicaciones

TCP/IP es un protocolo de red que es utilizado en innumerables áreas tales como redes de área local, dispositivos de control industrial, desarrollo de tecnología y muchos otros. Desde tal punto de vista, este proyecto ofrece un medio de comunicación ampliamente utilizado hoy en día, montado además sobre un dispositivo programable capaz de ser adaptado a situaciones requeridas en la industria.

- Traspaso de Conocimientos

Se ha demostrado que el rumbo hacia el uso de FPGAs en sistemas embebidos ha ido aumentando considerablemente con el tiempo<sup>2</sup>. Este proyecto les brinda a los ingenieros la posibilidad de adaptarse a las nuevas tendencias y tecnologías.

---

<sup>1</sup> Consultar la Bibliografía N°8.

<sup>2</sup> Véase un Gráfico de tendencia en el Anexo 9.12, para ver la evolución del uso de este dispositivo en el desarrollo de sistemas embebidos.

## **1.6. Aplicación en la Industria**

El uso de sistemas embebidos en la industria eléctrica ha ido aumentando a lo largo de los años y se espera que siga la misma senda en curso. El uso de FPGAs, comunicaciones y sistemas embebidos han proporcionando soluciones tanto a personas particulares como a la industria, a raíz de esto se ha estimado que la utilización de estos tres elementos continúe evolucionando considerablemente.

Una aplicación importante que podrá surgir de este proyecto es la de adaptar el prototipo a una planta industrial determinada, esto con el fin de revisar parámetros importantes y ser enviados para supervisión remota. En general esta adaptación se logra habilitando un bus de entrada del dispositivo para captar señales que brindarán innumerables datos digitales o eléctricos tales como temperatura, alarmas, sensores y notificaciones entre otros.

Este proyecto podría funcionar como sistema de control inclusive. Tal objetivo sería alcanzado habilitando también un puerto de salida y de entrada, mientras que se actualiza el software y la lógica programada para recibir comandos de configuración vía Ethernet y que el dispositivo sea capaz de tomar buenas decisiones automatizadas. Finalmente deberá emitir señales eléctricas o digitales para controlar hardware.

Todas las adaptaciones mencionadas están fuera del alcance del proyecto. Esto debido a que una hay innumerables plantas industriales, donde cada una de ellas posee características individualizadas, por lo que la adaptación deberá ser realizada como actividad posterior luego de finalizar este trabajo. Sin embargo, un ingeniero que posea experiencia y conocimientos en sistemas embebidos podrá encaminarse con facilidad para sacarle provecho a este novedoso proyecto y darle uso como sistema de supervisión o de control de plantas industriales.

## **1.7. Tecnología utilizada**

Fue seleccionada una FPGA de la familia Virtex-5 de Xilinx, la cual es un chip que proporciona gran capacidad para desarrollo de lógica, alto rendimiento y novedoso diseño<sup>1</sup>. El modelo LX330 posee gran capacidad, lo cual asegura el espacio suficiente para programar el chip con la totalidad del Firmware desarrollado.

Fue seleccionada también la tarjeta de desarrollo HAPS-51<sup>2</sup>, que admite insertar tarjetas adicionales compatibles con la marca Simplicity de HAPS. La tarjeta GEPHY<sup>3</sup> 1x1 es un ejemplo de ello, lo cual brinda una interfaz Ethernet que será utilizada para capa física en las comunicaciones.

## **1.8. Limitaciones**

Los distintos proveedores de FPGAs ofrecen familias o modelos de acuerdo a las necesidades de almacenamiento, velocidades y funciones que se requieran.

En este proyecto se ha seleccionado el modelo LX330 de la Familia Virtex-5, el cual posee una gran capacidad de almacenamiento, una cantidad elevada de puertos de entrada y salida y soporte para altas velocidades.

En caso de que se realice una adaptación de este proyecto a un modelo distinto al LX330 de la familia Virtex 5, tendrá que considerarse la capacidad de tal chip FPGA para almacenar el Firmware desarrollado.

Otro factor preponderante corresponde a los retardos mínimos que deberán presentarse en las señales lógicas de la FPGA. Cuando no se cumplan estas condiciones mínimas, el software de desarrollo de Xilinx se preocupará de informar que existe tal problema. En la mayoría de los casos, la solución será

---

<sup>1</sup> Existen otras Familias como Virtex-4 y Spartan3 previas a Virtex-5.

<sup>2</sup> Ver Anexos 9.1, 9.2 y 9.3

<sup>3</sup> Ver Anexo 9.7

bajar la frecuencia del reloj que alimenta al sistema; en caso contrario, habrá que reformular el diseño. La frecuencia máxima soportada por la FPGA LX330 es de 200 MHz para el control de las Memoria Principal y 100 MHz para el Procesador Microblaze, que ponen un límite en la frecuencia de operación al sistema.

Por otro lado, el socket RAW TCP no realiza fragmentación de paquetes TCP. Además, su data máxima de envío es de 1 KBytes, lo cual limita el tamaño del archivo HTML.

### **1.9. Material de entrega**

Al finalizar el proyecto se podrá contar con los siguientes elementos. Estos son archivos que conforman el proyecto del software XPS. El software XPS es utilizado para compilar el archivo binario a cargar en las FPGAs de Xilinx.

- Archivo proyecto\_html.XPS (*archivo de proyecto Xilinx*)
- Archivo proyecto\_html.UCF (*archivo de asignación de pines*)
- Archivo proyecto\_html.MHS (*archivo de representación de hardware*)
- Archivo proyecto\_html.MSS (*archivo de drivers*)
- Archivos fuente \*.C (*archivos de proyecto software C*)
- Archivos cabecera \*.H (*archivos cabeceras del software*)
- Archivo proyecto\_html.BIT (*archivo binario de programación*)

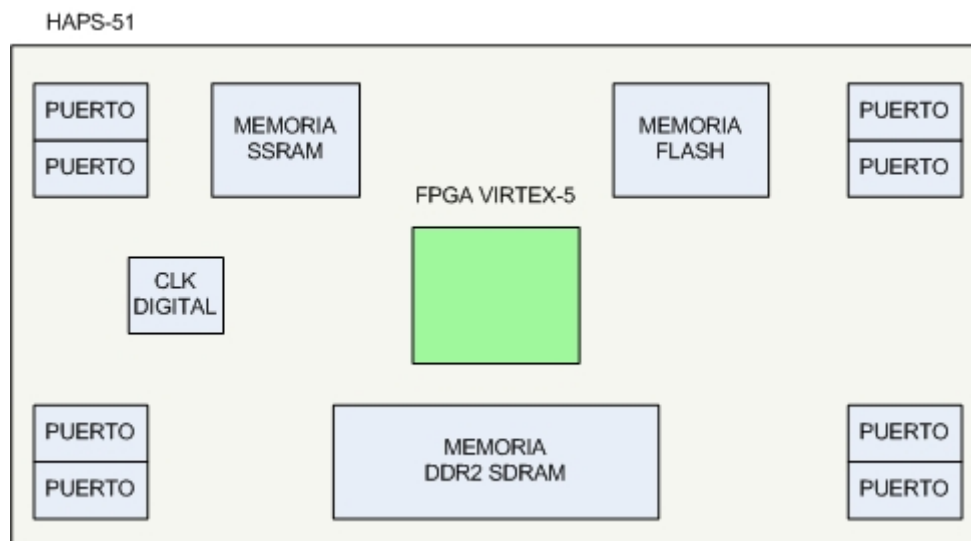
La tarjeta HAPS-51 no será proporcionada al finalizar el proyecto, pero puede ser adquirida por medio de la marca y proveedor de tecnología Simplicity.

## 2. Análisis del Problema

### 2.1. Situación Inicial del Sistema

Al comenzar este proyecto, se cuenta con una tarjeta de desarrollo HAPS-51, la cual dispone de una configuración previa básica, cuyos elementos importantes de mencionar son los siguientes:

- FPGA Virtex-5
- Puertos disponibles para periféricos
- Generador de Clock Digital sintonizable
- Memoria Flash
- Memoria DDR2 SDRAM
- Memoria SSRAM



*Figura 2-1 : Diagrama Tarjeta Haps-51  
Fuente: Propia (2010)*

La tarjeta de desarrollo se compone también de pistas eléctricas, las cuales unen pines del hardware con la Virtex-5, lo que posibilita el uso de las memorias Principal y Flash, el reloj digital o los 8 puertos Hapstrack II.

Para desarrollar un servidor HTML y tomar el control de los elementos de hardware necesarios, se requiere construir un Firmware en la FPGA.

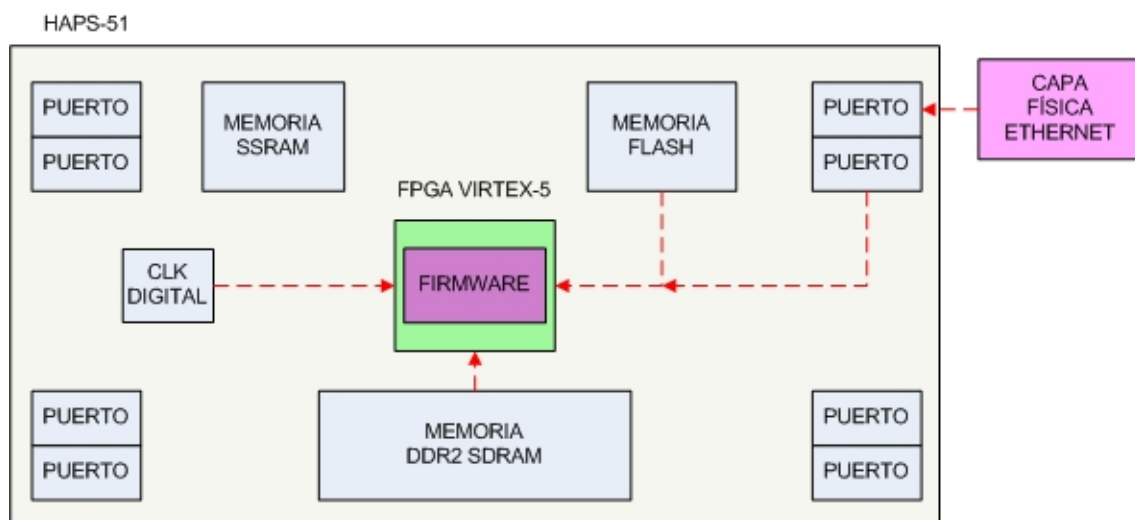
## 2.2. Propuesta Técnica del Sistema

La propuesta técnica del sistema está conformada por el desarrollo de un Firmware a programar en la FPGA, y el uso de recursos disponibles en la tarjeta HAPS-51.

Los recursos a utilizar serán las memorias principal y flash, la utilización del generador de señal de reloj, y un puerto para controlar una tarjeta externa de capa física Ethernet. La memoria de datos SSRAM no será utilizada ya que no se consideró necesario.

El Firmware, por un lado estará compuesto por lenguaje descripción de hardware VHDL para implementar la lógica, y por otro lado lenguaje de programación en C, para la implementación del software.

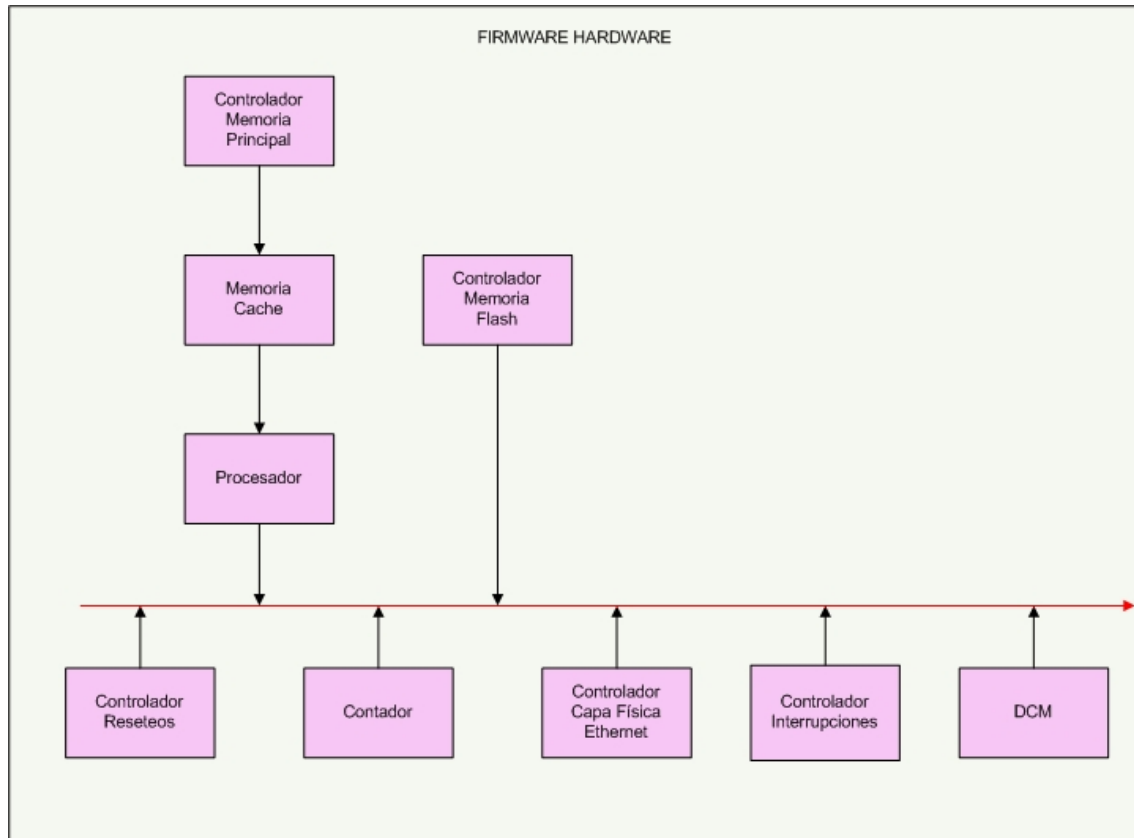
Esta propuesta se muestra a través del diagrama de la *Figura 2-2*.



*Figura 2-2 : Propuesta Hardware del Sistema*  
Fuente: Propia (2010)

### 2.3. Propuesta programación de Hardware

Para la realización de este proyecto se requiere implementar un hardware VHDL, el cual tendrá que dar soporte a los requerimientos para manejar una interfaz de comunicaciones Ethernet y un servidor HTML.



*Figura 2-3 : Propuesta de Hardware VHDL  
Fuente: Propia (2010)*

Para tal efecto se ha determinado que son necesarios los siguientes elementos básicos de hardware, los cuales también puede apreciarse en la *Figura 2-3*:

- Procesador
- Controlador de Capa física Ethernet (PHY, Physical Layer)
- Módulo controlador de memoria DDR2 SDRAM
- Módulo controlador de memoria Flash
- Memoria VHDL interna
- Buses de interconexión

- Periféricos
- Controlador de interrupciones
- Memoria de acceso rápido Cache
- Contador
- Generador de señales de clock digital
- Unidad depuradora
- Controlador de Reseteo del sistema

La función que cumplirán los elementos básicos será la siguiente:

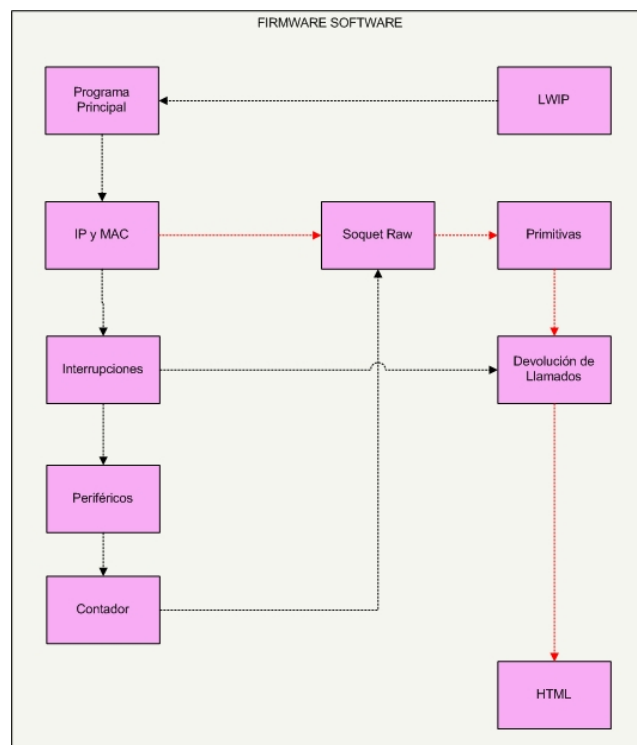
- Procesador: Brindará la capacidad de ejecutar las aplicaciones creadas en lenguaje de programación C (sockets e interrupciones entre otros).
- Controlador de Capa física Ethernet: Proporciona el control de la capa física Ethernet. Además deberá incorporar las características del estándar IEEE 802.3
- Módulo controlador de memoria principal DDR2 SDRAM: Brindará la capacidad de interactuar con la memoria DDR2 SDRAM la cual almacenará el programa en lenguaje de programación C que contiene la aplicación de los soques.
- Módulo controlador de memoria Flash: Brinda la capacidad de almacenar el programa C, incluso después de que sea apagada la tarjeta Xilinx y se desactive la memoria DDR2 SDRAM.
- Memoria VHDL interna: Memoria de acceso rápido, y de mayor velocidad respecto a los demás, pero de limitadas dimensiones. Esta será una memoria reservada para el futuro.
- Buses de interconexión: Proporcionarán la interconexión entre la unidad de procesamiento, las memorias y los periféricos.

- Controlador de interrupciones: Administrará las interrupciones que puedan existir para la unidad de procesamiento
- Memoria de acceso rápido Caché: Mejorará el rendimiento de la ejecución de la aplicación requerida para crear y ejecutar los sockets.
- Contador: Realizará la tarea de limpiar cada 250 milisegundos la memoria dinámica, por medio de una interrupción al procesador.
- Administrador de reloj digital: Será el encargado de generar las señales de reloj necesarias para todo el sistema.
- Unidad depuradora: Permitirá la descarga del proyecto hardware hacia el FPGA, de la descarga de la aplicación en C y depuración en caso de que sea necesario
- Controlador de Reseteo del Sistema: Será el módulo encargado de administrar los Reseteo requeridos por el sistema, ya sea a nivel de hardware o a nivel de software. Este deberá distribuir las señales a los módulos incorporados que lo requieran, como el procesador, las memorias y los periféricos al detectar errores.

## 2.4. Propuesta programación de Software

El software requiere disponer de un conjunto de instrucciones que permitan el manejo de las comunicaciones a nivel TCP/IP. Tales instrucciones deben estar orientadas a sistemas embebidos o aplicaciones de hardware a programar en una FPGA. La solución óptima es utilizar la librería LWIP, la cual proporciona Xilinx, brindando los requerimientos necesarios para administrar las comunicaciones por medio de primitivas de sockets TCP/IP.

Una vez habilitada la librería en el proyecto, será necesario realizar un levantamiento de la interfaz de capa física Ethernet presente en la tarjeta HAPS, incorporada en alguno de sus puertos. Se deberá también tomar control del contador de hardware, el cual es necesario debido a que se requiere realizar una limpieza de memoria dinámica del programa cada 250 milisegundos. La propuesta de software se muestra en la *Figura 2-4*.



*Figura 2-4 : Propuesta Software*  
Fuente: Propia (2010)

## **2.5. Costos del Proyecto**

La inversión del proyecto esta relacionada con los costos de adquirir los dispositivos electrónicos, horas hombre y licencias de softwares de programación.

Tales costos se muestran a continuación, mostrando el valor en pesos chilenos:

- Tarjeta HAPS-51:	\$ 600.000
- Cable programador de FPGAs JTAG:	\$ 50.000
- Tarjeta de red GEPHY 1x1:	\$ 50.000
- Licencia Software XPS:	\$ 220.000
- Horas Hombre:	\$ 1.000.000

Las Horas Hombre corresponden la labor de un ingeniero o técnico altamente capacitado para realizar rápidamente labores de programación y puesta en marcha del Firmware involucrado en la FPGA, tanto en el hardware como en el software.

En total el desarrollo del proyecto asume un costo total de \$ 1.920.000.

## 3. Desarrollo de Hardware

### 3.1. Resumen de Actividades

Las actividades a seguir para el desarrollo y la implementación del Hardware, tienen como objetivo dar sustento lógico a la parte Software.

La descripción de estas actividades es la siguiente:

- Selección de Módulos:

Se elige de manera adecuada la lógica a incorporar al proyecto, siendo seleccionada a partir del catálogo de módulos que proporciona la herramienta XPS (catálogo IPCores) y basada en los requerimientos de hardware.

- Comprensión funcional de los Módulos:

Rescatar la utilidad y estructura funcional de los módulos previamente seleccionados a partir de la documentación proporcionada por la herramienta XPS.

- Identificar Puertos o Señales de cada Módulo:

Proceso en el cual en cada módulo se identifican las señales que posteriormente se integran manualmente al sistema y los futuros pines externos.

- Definir Arquitectura:

Seleccionar la arquitectura del sistema (Harvard o alternativas), lo que posteriormente definirá el rumbo de la implementación lógica.

- Integración Módulo / Buses del Sistema:

Realizar la conexión entre los buses, módulos y el procesador del sistema.

- Ajuste de Memoria Principal:

Establecer la configuración apropiada de los tiempos de acceso, largo de bits y especificación de la memoria DDR2 (de la tarjeta HAPS-51), para una correcta comunicación del controlador MPMC.

- Ajuste de Memoria FLASH:

Establecer la configuración apropiada de los tiempos de acceso, largo de bits y especificación de la memoria FLASH (de la tarjeta HAPS-51), para una correcta comunicación del controlador XPS MCH EMC.

- Memoria Block Ram:

Especificar el tamaño y el uso que se le dará a la memoria Block Ram en el sistema.

- Mapeo de Memoria del Procesador:

Definir una dirección de memoria para cada uno de los módulos conectados al bus PLB del sistema, logrando que el procesador pueda acceder a tal lógica y controlarla por medio de lectura o escritura a sus registros.

- Implementación de Microblaze:

Definir las características adicionales que tendrá el procesador Microblaze, tales como unidad de punto flotante, divisora, interfaz de memoria cache entre otras.

- Frecuencias del Sistema:

Establecer la cantidad relojes usados, su frecuencia de operación y desfases.

- Interrupciones:

Crear la tabla de interrupciones del sistema y sus prioridades.

- Contador:

Configurar el contador del sistema para que comience a realizar cuentas.

- Reseteos

Especificar el funcionamiento de los reseteos del sistema, por medio del módulo controlador de Reseteos.

- Interconexión de Módulos

Se deberán conectar entre si los módulos, por medio de los puertos.

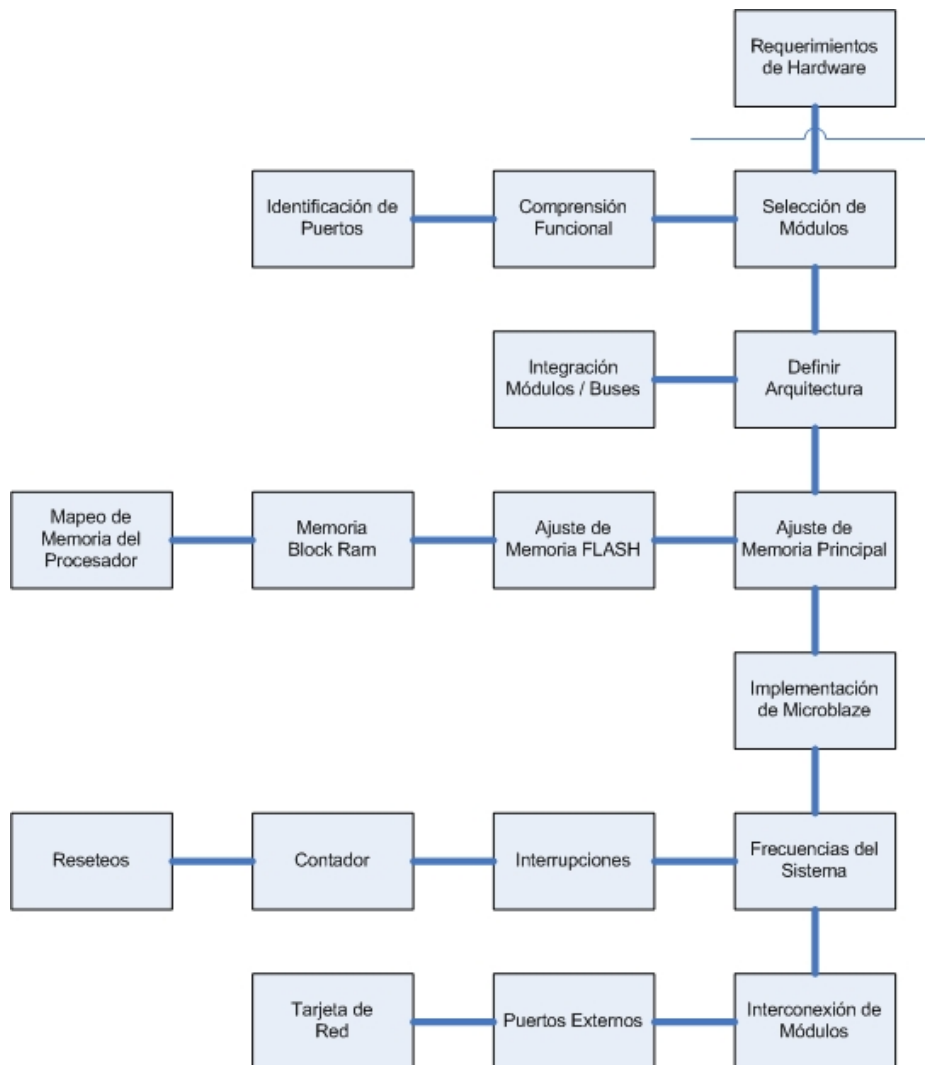
- Puertos Externos

Seleccionar las señales de cada módulo que serán convertidas en puertos externos y a su vez en pines, por medio del archivo UCF.

- Tarjeta de Red

Decidir y ajustar lo necesario para integrar la tarjeta GEPHY de red Ethernet a la tarjeta de desarrollo HAPS-51.

La *Figura 3-1* muestra un diagrama de tales actividades.



*Figura 3-1 : Actividades Hardware*  
*Fuente: Propia (2010)*

Para el desarrollo de estas actividades de hardware, se hará uso del software XPS proporcionado por Xilinx, en donde a lo largo de las etapas se irán mostrando imágenes de la aplicación XPS.

### **3.2. Selección de Módulos**

De todos los módulos disponibles por el software XPS, solo serán seleccionados los que más satisfacen los requerimientos de hardware. Estos son los siguientes:

- Procesador Microblaze  
(*Microblaze*)
- Bus de memoria local  
(*Local Memory Bus, lmb\_v10*)
- Procesador de bus local  
(*Processor Local Bus, plb\_v46*)
- Controlador de interfaz local de Bloque Ram  
(*Local Memory Bus controller interfaze, lmb\_bram\_if\_cntrl*)
- Controlador de memoria multipuertos  
(*Multi Protocol Memory Controller, MPMC*)
- Controlador de memoria externa de multi canal  
(*Multi Channel External Memory Controller, xps\_mch\_emc*)
- Bloque de BRAM  
(*BRAM Block*)
- Modulo depurador de Microblaze  
(*Microblaze Debug Module, MDM*)
- Controlador de Acceso al Medio  
(*Ethernet Lite Media Access Controller, xps\_ethernetlite*)
- Controlador de interrupciones  
(*Interrupt Controller, xps\_intc*)
- Contador Xilinx  
(*Timer, xps\_timer*)
- Generador de Reloj

(Clock Generator)

- EDK\_bufr\_r
- Controlador de reinicio del sistema

(Processor System Reset, *proc\_sys\_rst*)

Los últimos dos módulos llamados EDK\_bufr\_rx, EDK\_bufr\_tx no son proporcionados por el software XPS de Xilinx, y deben ser desarrollados en lenguaje VHDL. La implementación de estos módulos corresponde a la creación de una instancia BUFG para las señales RX y TX del módulo controlador Ethernet en el proyecto Xilinx. Esta actividad esta fuera del alcance del proyecto.

La *Figura 3-2* muestra los módulos incorporados. La columna IP Type corresponde al Tipo de hardware y la columna Name corresponde al nombre asignado.

Name	IP Type
External Ports	
<i>microblaze_0</i>	microblaze
<i>dlimb</i>	lmb_v10
<i>ilmb</i>	lmb_v10
<i>mb_plb</i>	plb_v46
<i>dlimb_cntrlr</i>	lmb_bram_if_cntrlr
<i>ilmb_cntrlr</i>	lmb_bram_if_cntrlr
<i>SDRAM_128M_64</i>	mpmc
<i>FLASH_32M_16</i>	xps_mch_emc
<i>lmb_bram</i>	bram_block
<i>debug_module</i>	mdm
<i>xps_ethernetlite_0</i>	xps_ethernetlite
<i>xps_intc_0</i>	xps_intc
<i>xps_timer_0</i>	xps_timer
<i>clock_generator_0</i>	clock_generator
<i>edk_bufr_ether_rx</i>	edk_bufr
<i>edk_bufr_ether_tx</i>	edk_bufr
<i>proc_sys_reset_0</i>	proc_sys_reset

*Figura 3-2 : Módulos VHDL incorporados*  
*Fuente: Propia (2010)*

### **3.3. Comprensión funcional de Módulos**

Es importante disponer de información de referencia de los módulos que se están utilizando, ya que muchos de estos ofrecen múltiples funcionalidades con la facultad de ser modificados.

A continuación se realizará una breve descripción de los módulos más relevantes, previo a realizar la implementación del sistema.

#### MICROBLAZE

Microblaze es un procesador que está organizado a través de una arquitectura Harvard con unidades de interfaz de acceso de instrucciones y datos, las cuales no son separadas, sino que se mapean en memoria distinta.

Es un procesador liviano de 32 bits, el cual fue diseñado para trabajar con sistemas embebidos. Posee un conjunto reducido de instrucciones, optimizadas para trabajar especialmente con FPGAs de Xilinx. Además, Microblaze utiliza el formato Big-endian para representar los datos.

Microblaze soporta reseteo, interrupciones, excepciones de usuario, excepciones de hardware, y break, cada una de las cuales cuenta con una prioridad predeterminada.

Microblaze puede trabajar con una interfaz depuradora como JTAG y aplicaciones depuradoras tal como Microprocessor debug Tool o bien conocido como XMD.

### XPS\_ETHERNETLITE

Está diseñado para incorporar las características del estándar IEEE 802.3 el cual puede ser conectado a una capa física de velocidades de 10 / 100 Mbps.

Puede conectarse al bus local del procesador y además incorpora puertos independientes de 2 K bytes para almacenar el paquete de datos tanto para recepción como para transmisión. Puede ser ampliable a 4 K bytes.

### MPMC

Este Core es un controlador de memorias DDR y DDR2 de tipo SDRAM, el cual soporta una cantidad total de 8 puertos seleccionables y configurables para comunicar la memoria.

La ventaja que aporta este Core al sistema está en los múltiples accesos disponibles que se pueden crear para acceder a la memoria desde diversos componentes, tales como dispositivos hardware conectados al bus PLB y procesadores.

A continuación se muestra un diagrama en el cual se aprecia todo el potencial de hardware que es capaz de generar este IPCore, haciendo efectiva la capacidad de acceder a una memoria externa tipo SDRAM por medio de múltiples dispositivos, incluso a través de dos arquitecturas distintas.

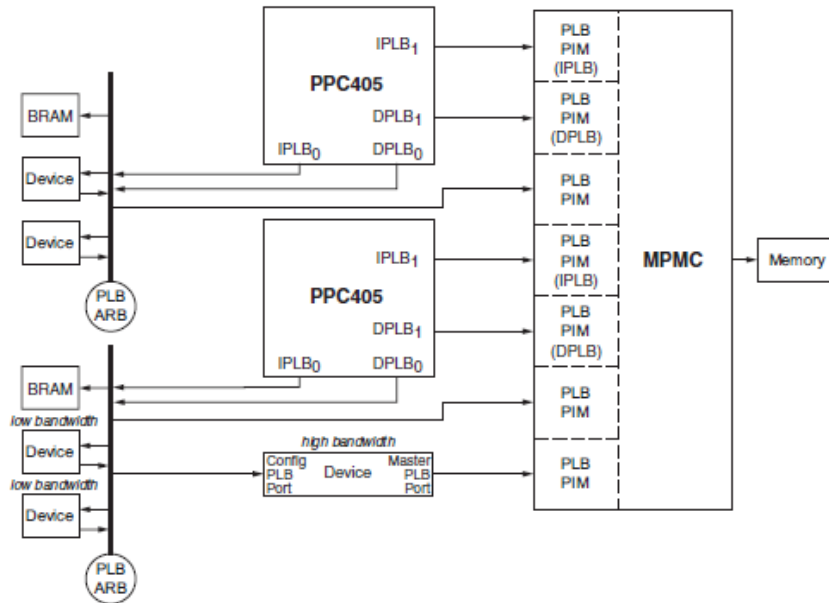


Figura 3-3 : Esquemático de módulo controlador de memoria principal  
Fuente, Xilinx DS643 (2010)

### CLOCK GENERATOR

Este IPCore llamado Clock Generator proporciona señales de reloj de acuerdo a las necesidades del uso de relojes en el sistema. Su capacidad implica generar múltiples salidas de reloj, con distintas frecuencias y fases, de acuerdo a una entrada análoga usada como referencia.

El siguiente diagrama de bloques muestra la composición interna del módulo.

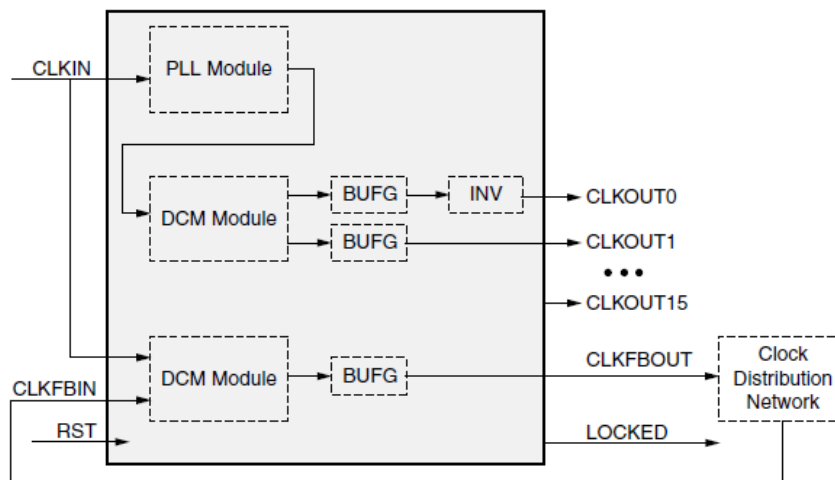


Figura 3-4 : Esquemático de módulo generador de reloj  
Fuente, Xilinx DS614 (2010)

Está compuesto por un módulo de Lazos Enganchados en Fase PLL (Phase-Locked Loop), por un administrador de clocks digital DCM (Digital Clock Manager) y buffers secundarios BUFG como entidades básicas.

Está disponible una señal llamada Locked la cual estará desactivada en caso de que alguna de las señales de reloj de salida no alcance los requerimientos necesarios.

### XPS MCH EMC

Este módulo proporciona la interfaz de control para memorias externas de tipo SRAM asíncrono y síncrono, y memorias Flash, por medio del bus PLB.

Sus principales características son:

- Posee un número parametrizable de interfaces controladoras de memoria desde 0 hasta 4 puertos, para controlar hasta 4 memorias del tipo mencionado anteriormente.
- Soporta memorias con ancho de datos de 32 bits, 16 bits y de 8 bits.
- Pueden configurarse los tiempos (timing) para la operación de los ciclos de escritura y de lectura.

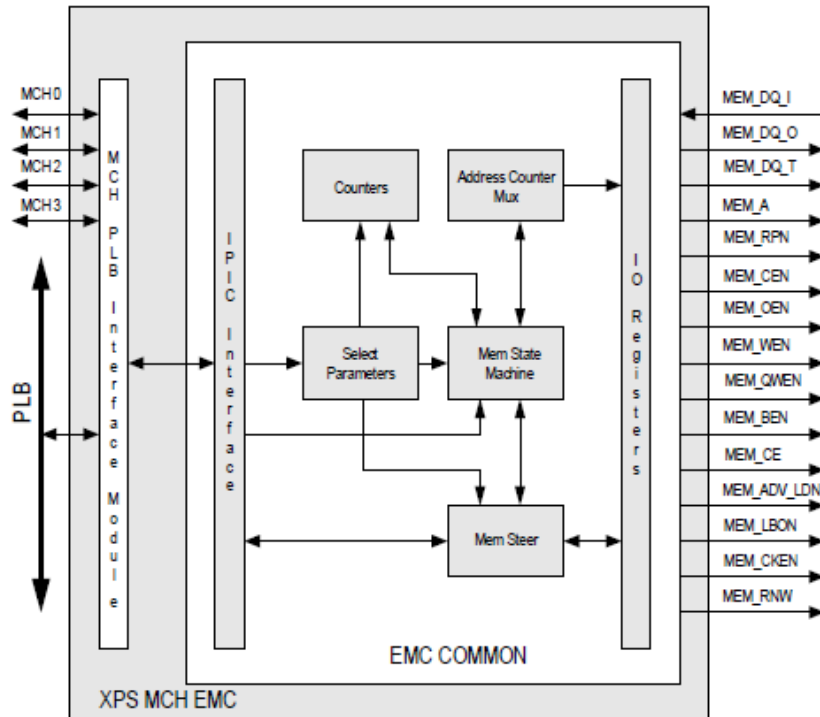


Figura 3-5 : Esquemático de módulo controlador de memoria flash y ssram  
Fuente: Xilinx DS575 (2010)

### BRAM BLOCK

Esta es una memoria tipo VHDL configurable y reducida en tamaño, pero con una altísima velocidad.

Para su control se requieren módulos especializados en BRAMs, los que son proporcionados por Xilinx.

La memoria posee dos puertos A y B, los que se pueden conectar a controladores de tipo LMB (Local Memory Bus) o bien PLB (Processor Local Bus).

### XPS PROC SYS RST

Proporciona un control de reset llevado a cabo por alguna señal externa de tipo asíncrona la cual se sincroniza con el reloj del sistema. Esta señal se puede configurar en lógica negativa o en lógica positiva para su activación.

Algunas de sus características son:

- Administra el reseteo del sistema completo de la tarjeta hacia los módulos.
- Presenta una entrada de enganche LOCK para los módulos generadores de frecuencias digitales.
- Configuración del ancho mínimo de pulso para señales de reseteo que serán reconocidas.

### XPS ETHERNET LITE

Este es un Core que incorpora las características del estándar IEEE.802.3 y posee una interfaz para el bus PLB. Puede ser configurado para una velocidad de 10 Mbps o de 100 Mbps en transferencia de datos. Su finalidad es proporcionar las funciones que establecen una interfaz Ethernet, por medio de un uso mínimo de recursos.

Las señales de salida y entrada de este hardware sirven para ser conectadas directamente a un MII (Media Independent Interface), la cual sirve de interfaz entre un Fast Ethernet y una PHY (Physical Layer).

Una de sus principales características es la de poseer una señal de interrupción para la transmisión y para la recepción de mensajes.

### **3.4. Puertos o Señales de los Módulos**

La siguiente etapa que consiste en reconocer las señales disponibles de cada módulo llamadas Puertos, los que se identifican de acuerdo a las siguientes propiedades:

- Nombre del Puerto
- Largo del Puerto
  - Cantidad de Bits
- Dirección del Puerto:
  - Entrada : I
  - Salida : O
  - Entrada-Salida IO
- Tipo de Puerto:
  - Reloj (*CLK*)
  - Interrupción (*INTERRUPT*)
  - Reseteo (*RST*)

A su vez estos pueden desconectarse o conectarse a otros módulos, o incluso convertirse en pines externos.

- Estado:
  - Desconectado (*No Connection*)
  - Conectado (*Nombre del Puerto*)

A continuación se muestran los puertos para cada módulo del proyecto.

microblaze_1			
MB_Halted	No Connection	▼	O
DBG_STOP	No Connection	▼	I
INTERRUPT	No Connection	▼	I INTERRUPT
MB_RESET	No Connection	▼	I RST

Figura 3-6 : Puertos disponibles para Microblaze  
Fuente: Propia (2010)

mpmc_0			
DDR2_DQS_n	No Connection	▼	IO
DDR2_DQS	No Connection	▼	IO
DDR2_DM	No Connection	▼	O
DDR2_DQ	No Connection	▼	IO
DDR2_Addr	No Connection	▼	O
DDR2_BankAddr	No Connection	▼	O
DDR2_WE_n	No Connection	▼	O
DDR2_CAS_n	No Connection	▼	O
DDR2_RAS_n	No Connection	▼	O
DDR2_ODT	No Connection	▼	O
DDR2_CS_n	No Connection	▼	O
DDR2_CE	No Connection	▼	O
DDR2_Clk_n	No Connection	▼	O CLK
DDR2_Clk	No Connection	▼	O CLK
MPMC_InitDone	No Connection	▼	O
MPMC_Idelayctrl_Rdy_O	No Connection	▼	O
MPMC_Idelayctrl_Rdy_I	No Connection	▼	I
MPMC_Rst	No Connection	▼	I RST
MPMC_Clk_200MHz	No Connection	▼	I CLK
MPMC_Clk90	No Connection	▼	I CLK
MPMC_Clk0_DIV2	No Connection	▼	I CLK
MPMC_Clk0	No Connection	▼	I CLK

Figura 3-7 : Puertos disponibles para controlador de memoria principal  
Fuente: Propia (2010)

xps_mch_emc_0			
Mem_RNW	No Connection	0	
Mem_CKEN	No Connection	0	
Mem_LBON	No Connection	0	
Mem_ADV_LDN	No Connection	0	
Mem_CE	No Connection	0	
Mem_BEN	No Connection	0	
Mem_QWEN	No Connection	0	
Mem_WEN	No Connection	0	
Mem_DEN	No Connection	0	
Mem_CEN	No Connection	0	
Mem_RPN	No Connection	0	
Mem_A	No Connection	0	
Mem_DQ_T	No Connection	0	
Mem_DQ_O	No Connection	0	
Mem_DQ_I	No Connection	1	
Mem_DQ	No Connection	10	
RdClk	No Connection	1	CLK

Figura 3-8 : Puertos disponibles para controlador de memoria flash y ssram  
Fuente: Propia (2010)

clock_generator_1			
LOCKED	No Connection	0	
RST	No Connection	1	RST
CLKOUT0	No Connection	0	CLK
CLKIN	No Connection	1	CLK

Figura 3-9 : Puertos disponibles para generador de reloj  
Fuente: Propia (2010)

proc_sys_reset_1			
Peripheral_Reset	No Connection	0	RST
Bus_Struct_Reset	No Connection	0	RST
MB_Reset	No Connection	0	RST
Dcm_locked	No Connection	1	
MB_Debug_Sys_Rst	No Connection	1	RST
Aux_Reset_In	No Connection	1	RST
Ext_Reset_In	No Connection	1	RST
Slowest_sync_clk	No Connection	1	CLK

Figura 3-10 : Puertos disponibles para controlador de reinicios  
Fuente: Propia (2010)

xps_intc_1			
Irq	No Connection	0	INTERRUPT
Intr	L to H: No Connection	1	INTERRUPT

Figura 3-11 : Puertos disponibles para controlador de interrupciones  
Fuente: Propia (2010)

xps_ethernetlite_1		
IP2INTC_Irpt	No Connection	INTERRUPT 0
PHY_tx_data	No Connection	0
PHY_tx_en	No Connection	0
PHY_rst_n	No Connection	0
PHY_rx_er	No Connection	1
PHY_col	No Connection	1
PHY_rx_data	No Connection	1
PHY_dv	No Connection	1
PHY_crs	No Connection	1
PHY_rx_clk	No Connection	1
PHY_tx_clk	No Connection	1

Figura 3-12 : Puertos disponibles de controlador Ethernet  
Fuente: Propia (2010)

lmb_v70_0		
SYS_Rst	No Connection	1
LMB_Clk	No Connection	1 CLK

Figura 3-13 : Puertos disponibles interfaz de block ram  
Fuente: Propia (2010)

mcm_0		
Debug_SYS_Rst	No Connection	0
Interrupt	No Connection	0 INTERRUPT

Figura 3-14 : Puertos disponibles modulo programador  
Fuente: Propia (2010)

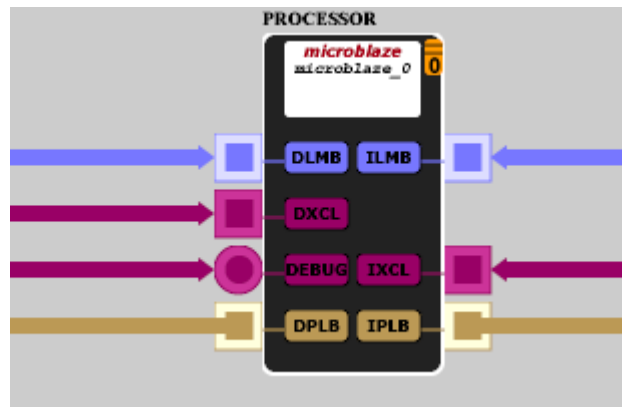
A modo de ejemplo, la *Figura 3-14* muestra el módulo xps\_ethernetlite\_1 el cual posee 11 puertos, donde el primero de ellos (Puerto 1) comprende lo siguiente:

- Nombre del Puerto: IP2INT\_Irpt
- Largo del Puerto
  - Cantidad de Bits: 1
- Dirección del Puerto:
  - Salida : 0
- Tipo de Puerto:
  - Interrupción (*INTERRUPT*)
- Estado
  - Desconectado (*No Connection*)

### 3.5. Definir Arquitectura

La implementación de la arquitectura será tipo Harvard<sup>1</sup>, ya que presenta una ventaja al separar el bus de datos y el de instrucciones. No es óptima la implementación Von Neumann<sup>2</sup>, debido a que junta en un solo bus a ambos, lo que resta eficiencia al sistema.

La *Figura 3-15* muestra el procesador Microblaze. Al lado derecho se ven las interfaces de los buses de instrucciones y en el lado izquierdo las interfaces de los buses de datos.



*Figura 3-15 : Buses de comunicación con Microblaze*  
*Fuente: Propia (2010)*

Las líneas de color café, púrpura y celeste (*Figura 3-15*) representan los buses PLB, XCL y LMB respectivamente.

<sup>1</sup> Harvard utiliza un bus para datos y un bus para instrucciones

<sup>2</sup> Von Neumann utiliza un mismo bus para datos y para instrucciones

### **3.6. Integración Módulo / buses del sistema**

Para la integración del sistema se requiere partir conectando el procesador Microblaze y los módulos (solo algunos módulos son capaces) a los buses seleccionados, esto para lograr comunicación entre el procesador y esta lógica incorporada. XPS ofrece múltiples formas de implementar un sistema de este tipo, donde la implantación final esta en manos del usuario.

La integración realizada al Bus PLB y LMB es:

- Conexionado al Bus PLB
  - Microblaze: Interfaz IPLB (Instrucciones), Maestro
  - Microblaze: Interfaz DPLB (Datos), Maestro
  - MPMC: Interfaz SPLB0, Esclavo
  - XPS\_MCH\_EMU: Interfaz SPLB, Esclavo
  - MDM: Interfaz SPLB, Esclavo
  - XPS\_Ethernetlite: Interfaz SPLB, Esclavo
  - XPS\_INTC: Interfaz SPLB, Esclavo
  - XPS\_TIMER: Interfaz SPLB, Esclavo
  
- Conexionado al Bus LMB 1
  - Microblaze: Interfaz ILMB (Instrucciones), Maestro
  - LMB\_BRAM\_IF\_CNTRLR: Interfaz SLMB, Esclavo
  
- Conexionado al Bus LMB 2
  - Microblaze: Interfaz DLMB (Datos), Maestro
  - LMB\_BRAM\_IF\_CNTRLR: Interfaz SLMB, Esclavo

La Figura 3-16 muestra la conexión realizada. Al costado izquierdo están los buses PLB, LMB y el canal XCL, mientras que en el costado derecho están los módulos incorporados.

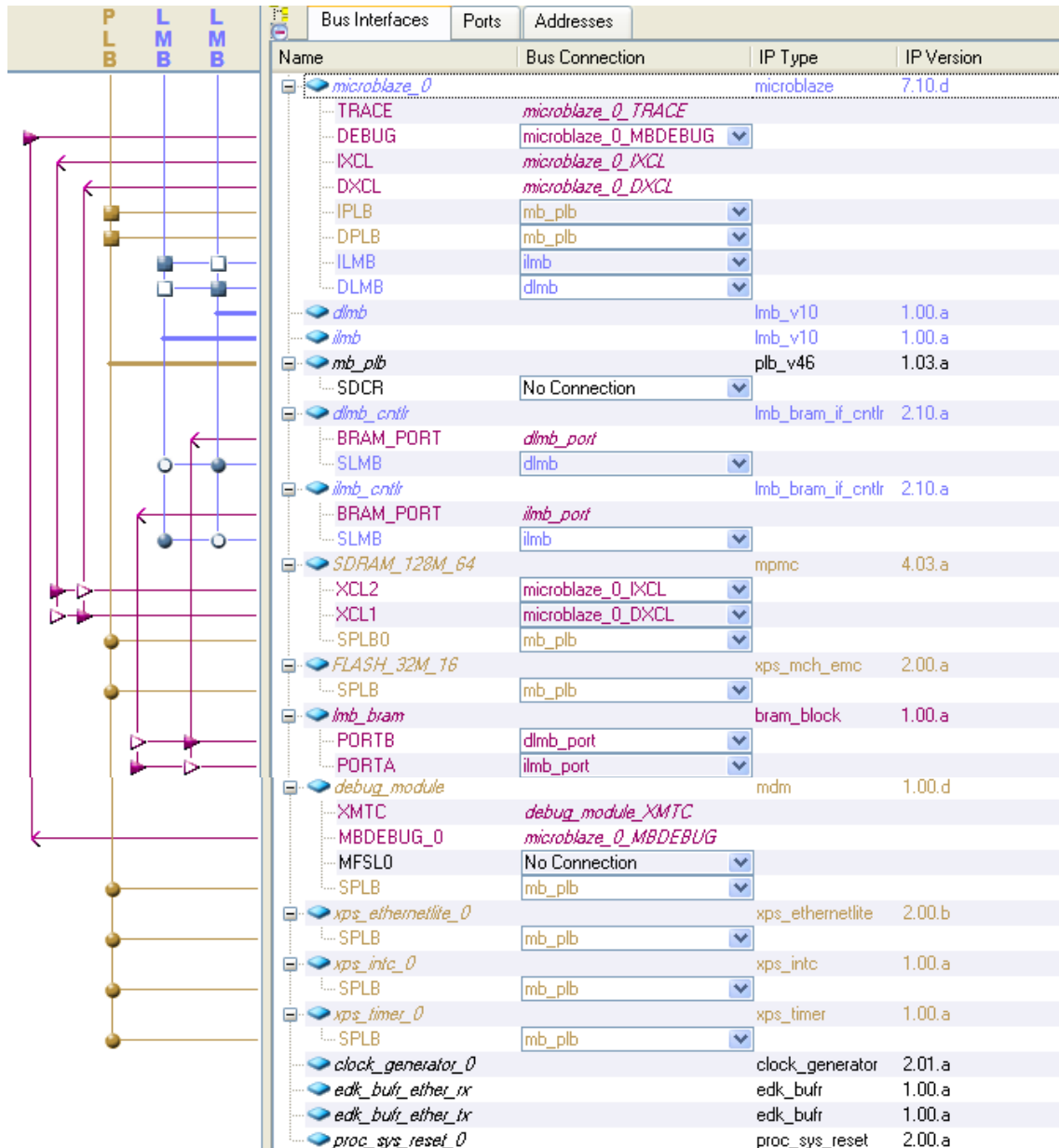


Figura 3-16 : Buses de comunicación con el Sistema  
Fuente: Propia (2010)

Los módulos clock\_genetator, edk\_buf\_ether\_rx, edk\_bufether\_tx y proc\_sys\_reset\_0 no pueden ser conectados a los buses debido a que no cuentan con la interfaz necesaria.

### **3.7. Ajuste de Memoria Principal**

El módulo controlador de la memoria principal (MPMC) proporciona una interfaz de control configurable para memorias ubicadas fuera de las FPGA.

Dentro de esta configuración es fundamental la especificación de los tiempos de acceso, ya que en caso de estar mal configuradas, el acceso provocará un error en el proceso de lectura y escritura de los datos o instrucciones. También es necesario disponer de información tal como la frecuencia de operación, el tamaño, la cantidad de bits y la cantidad de direcciones entre otras cosas.

La información técnica de las memorias se encuentra disponible en las hojas de datos del fabricante que corresponda, donde la memoria principal en este caso es una DDR2 SDRAM KINGSTON modelo KVR667D2U5/1G presente en la HAPS-51y disponible para ser controlada por el firmware o el módulo MPMC incorporado al proyecto.

A continuación se muestra la implementación realizada para los siguientes parámetros:

- Tiempo de Acceso
- Configuración de Parámetros
- Selección de Puertos

## TIEMPOS DE ACCESO:

Respecto al controlador MPMC, la configuración disponible de los tiempos de acceso es la siguiente:

- Frecuencia máxima para CAS latencia más baja (CASFA<sup>1</sup>)
- Latencia CAS más baja aplicable a la memoria(CASA)
- Segunda latencia CAS más baja aplicable a la memoria(CASB)
- Tercera latencia CAS más baja aplicable a la memoria(CASC)
- Tercera latencia CAS más baja aplicable a la memoria (CASD)
- Tiempo mínimo de Activo a Precarga (tRAS)
- Tiempo mínimo de Lectura a Escritura (tWTR)
- Tiempo mínimo de banco A al banco B (tRRD)
- Tiempo medio entre refrescamiento (tREFI)
- Tiempo máximo de Activo a Precarga (tRASMAX)
- Tiempo mínimo de Activo a Activo en mismo banco (tRC)

El ajuste realizado corresponde a los siguientes datos:

- CASFA: 333MHz
- CASA: 5
- CASB: 1
- CASC: 1
- CASD: 1
- tRAS: 45000 ps<sup>2</sup>
- tWTR: 7500 ps
- tRRD: 10000 ps
- tREIF: 7800000 ps
- tRASMAX: 7000000 ps
- tRC: 60000 ps

---

<sup>1</sup> Consultar Acrónimos

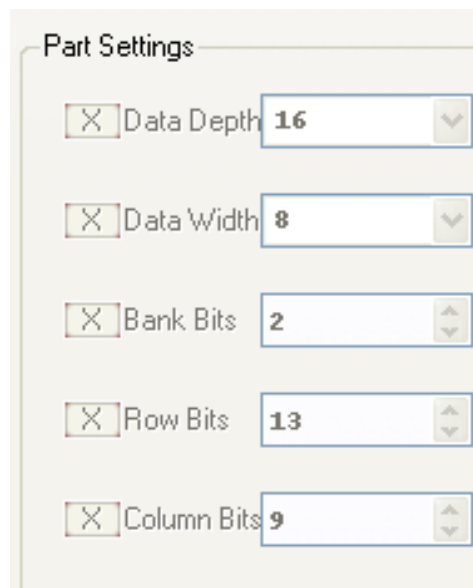
<sup>2</sup> Consultar Acrónimos

## CONFIGURACIÓN DE PARÁMETROS

Los parámetros que adaptan el controlador a la memoria son:

- Profundidad de los datos en bits (*Data Depth*)
- Ancho de los datos en bits (*Data Width*)
- Número de bancos de bits (*Bank Bits*)
- Número de filas en bits (*Row Bits*)
- Número de columnas en bits (*Column Bits*)
- Tipo de memoria

Los valores ingresados se aprecian en la *Figura 3-17*, la cual corresponde a la interfaz del software XPS para incorporar los valores de los parámetros. El tipo de memoria ingresado fue DDR2 SDRAM.



*Figura 3-17 : Parámetros de controlador de memoria principal  
Fuente: Propia (2010)*

## SELECCION DE PUERTOS

Es posible seleccionar el tipo y la cantidad de puertos que conformarán el control entre las señales del controlador MPMC y la memoria principal DDR2.

Las opciones disponibles por el controlador MPMC son las siguientes:

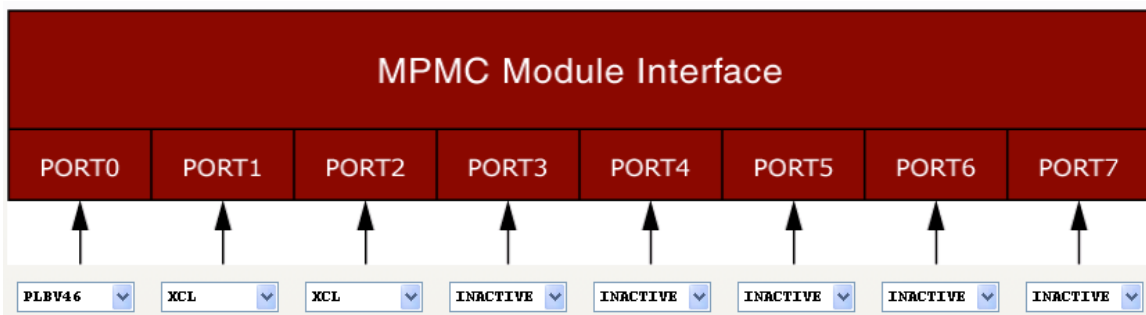
- Número de puertos disponibles para controlar la memoria son:
  - 1 a 8 Puertos.
  
- Tipo de Puerto disponible:
  - Inactivo
  - XCL: Xilinx Cache Link
  - PLB: Processor Local Bus
  - SDMA: Soft Direct Memory Access
  - NPI: Native Port Interface
  - PPC440MC: PowerPC 440 Processor
  - VFBC: Video Frame Buffer Controller

La implementación realizada fue la siguiente:

- Número de Puertos seleccionados:
  - 3

- Tipo de Puertos:
  - Puerto 0: PLB
  - Puerto1: XCL
  - Puerto2: XCL

La *Figura 3-18* muestra la implementación a nivel del software XPS realizada para el tipo y la cantidad de puertos.



*Figura 3-18 : Implementación de puertos controladores de memoria principal  
Fuente: Propia (2010)*

La implementación realizada indica que el controlador MPMC posee un total de tres puertos asignados y cinco deshabilitados. El puerto 0 (PORT0) brinda acceso al bus PLB, mientras que los puertos 1 y 2 (PORT1, PORT2) al bus XCL.

Todos los puertos asignados podrán comunicarse con la memoria DDR2 principal, por medio de este módulo multicontrolador de interfaz.

### 3.8. Ajuste de Memoria Flash

Al igual que la implementación realizada respecto del módulo MPMC, este módulo controlador de memorias Flash llamado MCH EMC deberá proporcionar la interfaz entre la FPGA y una memoria Flash K9F6408Q0C incorporada en la tarjeta de desarrollo HAPS-51.

A continuación se muestra la implementación realizada para:

- Tiempos de acceso
- Modo de operación
- Configuración de parámetros

#### TIEMPOS DE ACCESO

La implementación de los tiempos de acceso, se muestra en la *Figura 3-19*, donde la unidad de tiempo es ps (picosegundo). Esta corresponde a una ventana de configuración del software XPS para un módulo MCH EMC general.

TCEDV of Bank 0	15,000ps
TAVDV of Bank 0	15,000ps
THZCE of Bank 0	7,000ps
THZOE of Bank 0	7,000ps
TWC of Bank 0	15,000ps
TWP of Bank 0	12,000ps
TLZWE of Bank 0	0ps

*Figura 3-19 : Implementación de latencias de memoria principal  
Fuente: Propia (2010)*

En el ANEXO 9.11, se explican los parámetros de la *Figura 3-19*.

## MODO DE OPERACION

La memoria Flash ofrece dos modos de configuración. Estos modos son el Byte y el Word, en donde el primero ajusta el bus de datos a 8 bits, mientras que en el segundo lo ajusta en 16.

La selección modo, define a su vez la forma de implementar el bus de direcciones de esta memoria. En modo WORD el bit menos significativo (LBS) de las direcciones se deja sin conectar, realizando un direccionamiento cada 16 bits. En modo BYTE, en cambio, el bus de datos se implementa usando el LBS conectado, donde las direcciones serán cada en 8 bits.

El modo elegido es tipo WORD, debido a que para lectura y la escritura ofrece mayor rapidez.

## CONFIGURACIÓN DE PARAMETROS

Los parámetros de interés disponibles para este controlador son los siguientes:

- Largo del bus de datos  
*(Data Bus Width of Bank 0)*
- Múltiple acceso al Bus PLB para acceder a la memoria  
*(Execute Multiple Memory Accesses To Match Bank 0)*
- Memoria Asíncrona o Síncrona  
*(Bank Synchronous)*
- Latencia de Pipeline  
*(Pipeline Latency)*

La configuración implementada se aprecia por medio de la *Figura 3-20*, que muestra la configuración de los parámetros por medio del software XPS.

Data Bus Width of Bank 0	16
Execute Multiple Memory Accesses To Match Bank 0 Data Bus Width To PLB Data Bus Width	TRUE
Bank 0 is Synchronous	Asynchronous
Pipeline Latency of Bank 0	2

*Figura 3-20 : Ajuste de controlador de memoria flash*  
*Fuente: Propia (2010)*

### **3.9. Memoria Block Ram**

La memoria Block Ram es de un tamaño variable y se ajusta según los requerimientos de diseño. Esta memoria puede utilizarse para guardar información digital, de datos de software o de instrucciones de programa.

Su capacidad se encuentra limitada (para las FPGAs Virtex-5) a un rango que va de un mínimo de 4 KBytes a un tamaño máximo de 512 KBytes por cada memoria Block Ram incorporada. Lo anterior también depende de la capacidad total de memoria disponible en el proyecto.

La implementación de este proyecto no requiere una memoria Block Ram de tamaño amplio, sin embargo, es necesario un tamaño mínimo adecuado para que sirva de memoria Cache en el sistema.

La incorporación de una memoria Block Ram logra un aumento notablemente en la velocidad de ejecución de lectura o escritura de datos o instrucciones en los programas, lo que le da una ventaja a los sistemas que se estén desarrollando.

Además, es necesario la incorporación de un módulo Block Ram para la inicialización del programa, debido a que se requiere realizar la carga del software cada vez que se energiza la tarjeta HAPS-51. Esta carga se realiza almacenando un pequeño programa llamado Bootloader en esta memoria Block Ram, el cual se encarga de ejecutar las primeras líneas de instrucciones, las que traen el software desde la memoria Flash, llevándolo a la memoria principal y comenzar así el Microblaze a ejecutarlo.

El tamaño a implementar para el modulo Block Ram será de 16 KBytes. Este se dividirá en 2 partes, 8 KBytes para el Cache de datos y 8 KBytes para el Cache de instrucciones.

### ***3.10. Mapeo de Memoria del Procesador***

Para que el procesador Microblaze pueda tener acceso a los registros de los módulos implementados y de esa manera controlarlos, es necesario realizar un mapa de memoria de los módulos. Este mapa es un archivo de texto que especifica la ubicación de cada módulo conectado al procesador Microblaze mediante direcciones hexadecimales, lo que permite saber donde ir a buscarlos en el sistema.

Estos módulos deben tener la capacidad de permitir acceso por software, ya que algunos no poseen la característica de ser conectados a ningún bus en específico, lo que es requisito para estos casos el poseer interfaz con el bus PLB del sistema.

Para cada módulo en el archivo de texto del mapa de direcciones, se representa su ubicación mediante una dirección base y una dirección final o alta. La diferencia entre la dirección base y la dirección alta, mostrará la cantidad de memoria de mapeo que el procesador Microblaze destina para acceder a tales registros.

La *Figura 3-21* muestra la implementación efectuada al mapa de memoria. A modo de referencia, las Block Ram de datos (dlmb\_cntrl) y de instrucciones (ilmb\_cntrl) tienen un tamaño de 8KBytes, donde coinciden con la capacidad previamente seleccionada.

Instance	Name ▲	Base Address	High Address	Size	Bus Interface(s)
dlmb_cntrl	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB
ilmb_cntrl	C_BASEADDR	0x00000000	0x00001fff	8K	SLMB
debug_module	C_BASEADDR	0x84400000	0x8440ffff	64K	SPLB
mb_plb	C_BASEADDR			U	Not Applicable
xps_ethernetlite_0	C_BASEADDR	0x81000000	0x8100ffff	64K	SPLB
xps_intc_0	C_BASEADDR	0x81800000	0x8180ffff	64K	SPLB
xps_timer_0	C_BASEADDR	0x83c00000	0x83c0ffff	64K	SPLB
FLASH_32M_16	C_MEM0_BASEADDR	0xa4000000	0xa7ffffff	64M	SPLB
SDRAM_128M_64	C_MPMC_BASEADDR	0xc0000000	0xffffffff	1G	SPLB0:XCL1:XCL2

*Figura 3-21 : Mapa de memoria del procesador*  
Fuente: Propia (2010)

### **3.11. Implementación de Microblaze**

El procesador Microblaze es flexible de acuerdo a que puede ser implementado incorporando o quitando algunas características adicionales. La *Figura 3-22* presenta un diagrama de la estructura interna del procesador, en donde se resalta con color gris los atributos opcionales que pueden ser incorporados a sus capacidades.

Los atributos opcionales para el procesador Microblaze son:

- Bus de datos de tipo cache XCL para instrucciones
- Bus de instrucciones tipo cache XCL para datos
- Unidad multiplicadora
- Unidad divisora
- Conexión al bus PLB para datos
- Conexión al bus PLB para instrucciones
- Conexión al canal FSL
- Unidad de punto flotante.

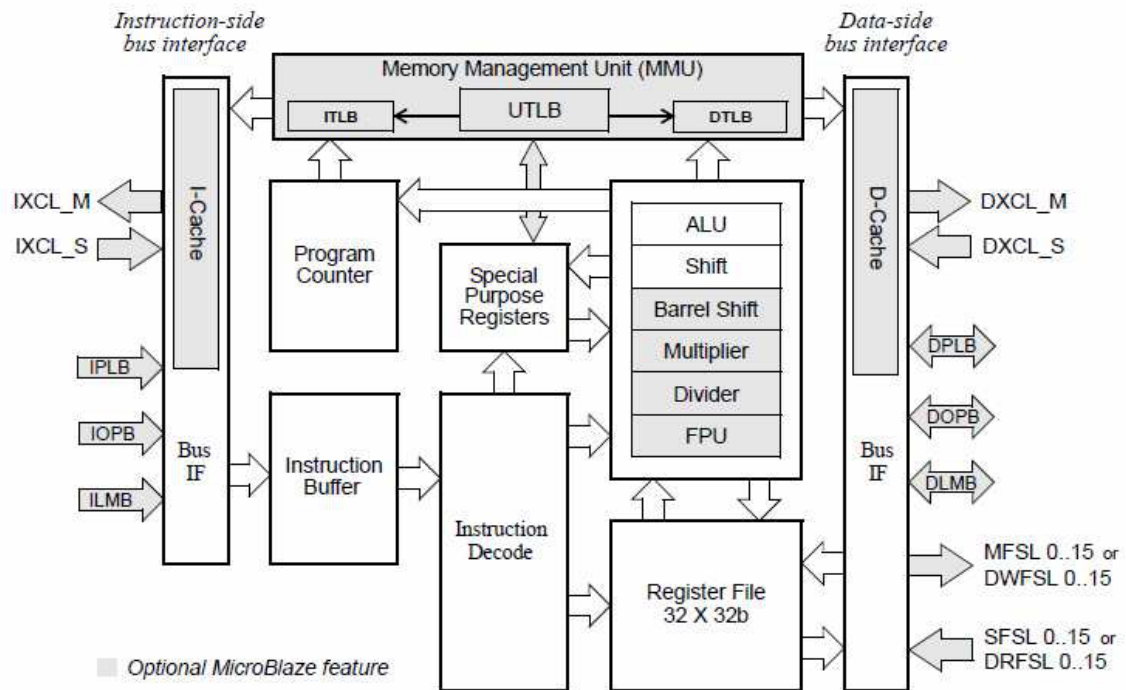


Figura 3-22 : Esquemático del módulo Microblaze  
Fuente: Xilinx UG081 (v9.0)

De acuerdo a los requerimientos del proyecto se determinó utilizar:

- Bus cache XCL para la interconexión directa entre el procesador y la memoria RAM tanto para datos como para instrucciones.
- Bus PLB para la interconexión entre el procesador y los periféricos tanto para datos como para instrucciones

No fue necesario utilizar lo siguiente:

- Unidad multiplicadora
- Unidad divisora
- Unidad de punto flotante
- Bus FSL

### **3.12. Frecuencias del Sistema**

La siguiente es una lista de algunos de los parámetros de interés que pueden utilizarse en el módulo administrador de Reloj Digital:

- Frecuencia de la señal de reloj de entrada:  
CLK\_IN\_F
- Incorporación de un buffer a la señal reloj de entrada:  
CLK\_IN\_BUF
- Numero de señales de reloj a ocupar, de salida:  
CLK\_N\_PORTS
- Frecuencia de cada una de las señales de reloj de salida:  
CLKN\_F
- Fase de cada una de las señales de salida:  
CLKN\_P
- Incorporación de un buffer a alguna de las señales de reloj de salida:  
CLKN\_BUF
- Relojes de salida serán generadas por PLL o por DCM:  
CLKN\_INF

Luego, se presenta la configuración seleccionada para cada uno de los parámetros y opciones previamente mencionadas:

- CLK\_IN\_F: 100.000.000 Hz
- CLK\_IN\_BUF: FALSO
- CLK\_N\_PORTS: 3

- CLK0\_F: 100.000.000 Hz
- CLK0\_P: 0
- CLK0\_BUF: FALSO
- CLK0\_INF: NADA
  
- CLK1\_F: 200.000.000 Hz
- CLK1\_P: 90
- CLK1\_BUF: VERDAD
- CLK1\_INF: PLL
  
- CLK2\_F: 200.000.000 Hz
- CLK2\_P: 0
- CLK2\_BUF: VERDAD
- CLK2\_INF: PLL

La *Figura 3-23*, presenta una tabla del software XPS para el módulo Reloj Digital. Las cuatro primeras filas (1, 2, 3 y 4) corresponden a los relojes generados, en donde se muestra el nombre de las señales (Port Name), su frecuencia (Frequency MHz) de operación y desfase (Phase Shift).

	Port Name	Connected to	Frequency (MHz)	Phase Shift	Group
1	CLKIN	dcm_clk_s	100.000000		
2	CLKOUT0	sys_clk_s	100.000000		
3	CLKOUT1	clock_200_90	200.000000	90	PLLO
4	CLKOUT2	clock_200	200.000000		PLLO
5	RST	net_gnd			
6	LOCKED	Dcm_all_locked			

*Figura 3-23 : Tabla de frecuencias del módulo administrador de reloj  
Fuente: Propia (2010)*

### 3.13. Interrupciones

El controlador de interrupciones es implementado realizando asignaciones de Puertos del tipo interrupción a un registro de tabla de prioridades. Estos Puertos involucrados son provenientes de otros módulos, y se caracterizan porque generan una interrupción cada vez que se gatilla un evento determinado.

La salida de este módulo será conectada al procesador Microblaze, en donde el procesador se detendrá y atenderá las peticiones generadas de acuerdo a las prioridades establecidas en la tabla.

La *Figura 3-24*, muestra la tabla de prioridades implementada a través del software XPS, la cual se conforma por las señales de interrupción de los módulos Ethernet, Contador y MDM.



Connected Interrupts	Priority
debug_module_Interrupt	Low
xps_timer_0_Interrupt	
xps_ethernetlite_0_IP2INTC_Irpt	High

*Figura 3-24 : Tabla de interrupciones  
Fuente: Propia (2010)*

La primera prioridad fue asignada al módulo Ethernet, en donde la interrupción es generada con la llegada de un dato al buffer.

La segunda se asignó al Contador, el cual genera una interrupción periódica con cada nuevo ciclo (Termine de Contar). El evento deberá asociarse con la limpieza de registros de software utilizados por la librería LWIP y de los sockets.

La última prioridad corresponde al módulo MDM, evento que servirá para descargar los programas y probar el procesador Microblaze una vez terminado el desarrollo de hardware y software.

### **3.14. Contador**

Para el módulo contador es necesario ingresar la información de los siguientes parámetros:

- Cantidad de bits del registro que almacena el número del contador  
*(The Width of Counter in Timer)*
- Cantidad de Contadores presentes  
*(Only One Timer is present)*
- La lógica negativa o positiva de las señales de interrupción
  - TRIG0 *(Se activa al terminar la cuenta del contador número 0)*
  - TRIG1 *(Se activa al terminar la cuenta del contador número 1)*
  - GEN0 *(Se activa al generarse la cuenta del contador número 0)*
  - GEN1 *(Se activa al generarse la cuenta del contador número 1)*

La *Figura 3-25* muestra la implementación para el módulo Contador, realizada a través del software XPS.

The Width of Counter in Timer	32
Only One Timer is present	<input type="checkbox"/>
TRIG0 Active Level	1
TRIG1 Active Level	1
GEN0 Active Level	1
GEN1 Active Level	1

Figura 3-25 : Parámetros de operación del contador  
Fuente: Propia (2010)

La señal de interrupción se genera cuando el contador termina su cuenta. Esta cuenta se establece por medio de un Driver en el Software.

### 3.15. Reseteos

Los reseteos del sistema se comportan de acuerdo a los siguientes parámetros, que corresponden a la configuración del módulo controlador de Resets:

- Número de ciclos de reloj antes que la entrada de reseteo externa sea reconocida (*Number of Clocks Before Input Change is Recognized On The External Reset Input*)
- Número de ciclos de reloj antes que la entrada de reseteo auxiliar sea reconocida (*Number of Clocks Before Input Change is Recognized On the Auxiliary Reset Input*)
- Puerto de reseteo externo en lógica positiva o negativa (*External Reset Active High*)
- Puerto de reseteo auxiliar en lógica positiva o negativa (*Auxiliary Reset Active High*)

- Número de salidas de reseteo para entradas de Bus  
(*Number of Bus Structure Reset Registered Outputs*)
- Número de salidas de reseteo para entradas de Periféricos  
(*Number of Peripheral Reset Registered Outputs*)

La implementación de este controlador se muestra en la *Figura 3-26*.

Number of Clocks Before Input Change is Recognized On The External Reset Input	4
Number of Clocks Before Input Change is Recognized On The Auxiliary Reset Input	4
External Reset Active High	0
Auxiliary Reset Active High	1
Number of Bus Structure Reset Registered Outputs	1
Number of Peripheral Reset Registered Outputs	1

*Figura 3-26 : Parámetros de operación del controlador de reseteos*  
*Fuente: Propia (2010)*

### **3.16. Interconexión de módulos**

Los módulos requieren ser interconectados por medio de sus puertos y se deberán cumplir las condiciones necesarias. Esto implica que si por un lado un puerto es de salida, por otro lado deberá ser de entrada. Además ambas señales tendrán que poseer el mismo largo (la misma cantidad de bits).

La *Figura 3-27*, *Figura 3-28*, *Figura 3-29* y *Figura 3-30* muestran la implementación mediante el software XPS de la conexión realizada entre los módulos por medios de los puertos.

microblaze_0			microblaze
MB_Halted	No Connection	0	
DBG_STOP	No Connection	I	
Interrupt	Interrupt	I	INTERRUPT
MB_RESET	mb_reset	I	RST
dlmb			lmb_v10
SYS_Rst	sys_bus_reset	I	
LMB_Clk	sys_clk_s	I	CLK
ilmb			lmb_v10
SYS_Rst	sys_bus_reset	I	
LMB_Clk	sys_clk_s	I	CLK
mb_plb			plb_v46
Bus_Error_Det	No Connection	0	INTERRUPT
SYS_Rst	sys_bus_reset	I	RST
PLB_Clk	sys_clk_s	I	CLK
dlmb_cntlr			lmb_bram_if_cntlr
ilmb_cntlr			lmb_bram_if_cntlr

Figura 3-27 : Conexión intermodular 1  
Fuente: Propia (2010)

SDRAM_128M_64			mPMC
DDR2_DQS_n	SDRAM_128M_64_DDR2_DQS_n	I/O	
DDR2_DQS	SDRAM_128M_64_DDR2_DQS	I/O	
DDR2_DM	SDRAM_128M_64_DDR2_DM	0	
DDR2_DQ	SDRAM_128M_64_DDR2_DQ	I/O	
DDR2_Addr	SDRAM_128M_64_DDR2_Addr	0	
DDR2_BankAddr	SDRAM_128M_64_DDR2_BankAddr	0	
DDR2_WE_n	SDRAM_128M_64_DDR2_WE_n	0	
DDR2_CAS_n	SDRAM_128M_64_DDR2_CAS_n	0	
DDR2_RAS_n	SDRAM_128M_64_DDR2_RAS_n	0	
DDR2_ODT	SDRAM_128M_64_DDR2_ODT	0	
DDR2_CS_n	SDRAM_128M_64_DDR2_CS_n	0	
DDR2_CE	SDRAM_128M_64_DDR2_CE	0	
DDR2_Clk_n	SDRAM_128M_64_DDR2_Clk_n	0	CLK
DDR2_Clk	SDRAM_128M_64_DDR2_Clk	0	CLK
MPMC_InitDone	No Connection	0	
MPMC_Idelayctrl_Rdy_0	No Connection	0	
MPMC_Idelayctrl_Rdy_1	No Connection	I	
MPMC_Rst	sys_periph_reset	I	RST
MPMC_Clk_200MHz	clock_200	I	CLK
MPMC_Clk90	clock_200_90	I	CLK
MPMC_Clk0_DIV2	sys_clk_s	I	CLK
MPMC_Clk0	clock_200	I	CLK
FLASH_32M_16			xps_mch_emc
lmb_bram			bram_block
debug_module			mdm
Debug_SYS_Rst	Debug_SYS_Rst	0	
Interrupt	debug_module_Interrupt	0	INTERRUPT

Figura 3-28 : Conexión intermodular 2  
Fuente: Propia (2010)

<b>xps_ethernetlite_0</b>				xps_ethernetlite
IP2INTC_lrp	xps_ethernetlite_0_IP2INTC_lrp	0	INTERRUPT	
PHY_tx_data	xps_ethernetlite_0_PHY_tx_data	0		
PHY_tx_en	xps_ethernetlite_0_PHY_tx_en	0		
PHY_rst_n	xps_ethernetlite_0_PHY_rst_n	0		
PHY_rx_er	xps_ethernetlite_0_PHY_rx_er	1		
PHY_col	xps_ethernetlite_0_PHY_col	1		
PHY_rx_data	xps_ethernetlite_0_PHY_rx_data	1		
PHY_dv	xps_ethernetlite_0_PHY_dv	1		
PHY_crs	xps_ethernetlite_0_PHY_crs	1		
PHY_rx_clk	edk_buf_ether_rx_out_clk	1		
PHY_tx_clk	edk_buf_ether_tx_out_clk	1		
<b>xps_intc_0</b>				xps_intc
Irq	Interrupt	0	INTERRUPT	
Intr	L to H: debug_module_Interrupt&xps_timer_0_Interrupt&xl		INTERRUPT	
<b>xps_timer_0</b>				xps_timer
Freeze	No Connection	1		
Interrupt	xps_timer_0_Interrupt	0	INTERRUPT	
PWM0	No Connection	0		
GenerateOut1	No Connection	0		
GenerateOut0	No Connection	0		
CaptureTrig1	No Connection	1		
CaptureTrig0	No Connection	1		
<b>clock_generator_0</b>				clock_generator
LOCKED	Dcm_all_locked	0		
RST	net_gnd	1	RST	
CLKOUT2	clock_200	0	CLK	
CLKOUT1	clock_200_90	0	CLK	
CLKOUT0	sys_clk_s	0	CLK	
CLKIN	dcm_clk_s	1	CLK	

Figura 3-29 : Conexión intermodular 3  
Fuente: Propia (2010)

<b>edk_buf_ether_rx</b>				
out_clk	edk_buf_ether_rx_out_clk	0		
in_clk	xps_ethernetlite_0_PHY_rx_clk	1		
<b>edk_buf_ether_tx</b>				
out_clk	edk_buf_ether_tx_out_clk	0		
in_clk	xps_ethernetlite_0_PHY_tx_clk	1		
<b>proc_sys_reset_0</b>				
Peripheral_Reset	sys_periph_reset	0	RST	
Bus_Struct_Reset	sys_bus_reset	0	RST	
MB_Reset	mb_reset	0	RST	
Dcm_locked	Dcm_all_locked	1		
MB_Debug_Sys_Rst	Debug_SYS_Rst	1	RST	
Aux_Reset_In	No Connection	1	RST	
Ext_Reset_In	sys_rst_s	1	RST	
Slowest_sync_clk	sys_clk_s	1	CLK	

Figura 3-30 : Conexión intermodular 4  
Fuente: Propia (2010)

### 3.17. Puertos Externos

Los puertos externos tienen la propiedad de convertirse en pines de la FPGA y enlazarse con la tarjeta HAPS. En este caso es necesario controlar la memoria principal DDR2 SDRAM, la memoria Flash, la tarjeta adicional Ethernet a incorporar a la HAPS y la señal de reloj de entrada.

La Figura 3-31 muestra la implementación de puertos externos realizada en el proyecto. En la primera columna se muestra el nombre del futuro pin incorporado, en la segunda el nombre del puerto, en la tercera la dirección de entrada o salida del puerto, la cuarta el tipo de puerto y por último se muestra información del largo en bits del puerto.

<i>sram_clk_pin</i>	sys_clk_s	0		
<i>xps_ethernetlite_0_PHY_rst_n_pin</i>	xps_ethernetlite_0_PHY_rst_n	0		
<i>xps_ethernetlite_0_PHY_dv_pin</i>	xps_ethernetlite_0_PHY_dv	1		
<i>xps_ethernetlite_0_PHY_rx_er_pin</i>	xps_ethernetlite_0_PHY_rx_er	1		
<i>xps_ethernetlite_0_PHY_rx_data_pin</i>	xps_ethernetlite_0_PHY_rx_data	1		[3:0]
<i>xps_ethernetlite_0_PHY_tx_clk_pin</i>	xps_ethernetlite_0_PHY_tx_clk	1		
<i>xps_ethernetlite_0_PHY_rx_clk_pin</i>	xps_ethernetlite_0_PHY_rx_clk	1		
<i>xps_ethernetlite_0_PHY_tx_data_pin</i>	xps_ethernetlite_0_PHY_tx_data	0		[3:0]
<i>xps_ethernetlite_0_PHY_tx_en_pin</i>	xps_ethernetlite_0_PHY_tx_en	0		
<i>xps_ethernetlite_0_PHY_crs_pin</i>	xps_ethernetlite_0_PHY_crs	1		
<i>xps_ethernetlite_0_PHY_col_pin</i>	xps_ethernetlite_0_PHY_col	1		
<i>FLASH_32M_16_SRAM_2M_32_Mem_DQ</i>	FLASH_32M_16_SRAM_2M_32_Mem_DQ	10		[0:31]
<i>SDRAM_128M_64_DDR2_DQ</i>	SDRAM_128M_64_DDR2_DQ	10		[63:0]
<i>SDRAM_128M_64_DDR2_DQS_n</i>	SDRAM_128M_64_DDR2_DQS_n	10		[7:0]
<i>SDRAM_128M_64_DDR2_DQS</i>	SDRAM_128M_64_DDR2_DQS	10		[7:0]
<i>SDRAM_128M_64_DDR2_Addr_pin</i>	SDRAM_128M_64_DDR2_Addr	0		[12:0]
<i>SDRAM_128M_64_DDR2_ODT_pin</i>	SDRAM_128M_64_DDR2_ODT	0		[1:0]
<i>SDRAM_128M_64_DDR2_Clk_pin</i>	SDRAM_128M_64_DDR2_Clk	0	CLK	[1:0]
<i>FLASH_32M_16_SRAM_2M_32_Mem_A_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_A	0		[0:31]
<i>FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_BEN	0		[0:3]
<i>FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN	0		
<i>SDRAM_128M_64_DDR2_BankAddr_pin</i>	SDRAM_128M_64_DDR2_BankAddr	0		[2:0]
<i>SDRAM_128M_64_DDR2_CS_n_pin</i>	SDRAM_128M_64_DDR2_CS_n	0		[1:0]
<i>SDRAM_128M_64_DDR2_CE_pin</i>	SDRAM_128M_64_DDR2_CE	0		[1:0]
<i>SDRAM_128M_64_DDR2_WE_n_pin</i>	SDRAM_128M_64_DDR2_WE_n	0		
<i>SDRAM_128M_64_DDR2_CAS_n_pin</i>	SDRAM_128M_64_DDR2_CAS_n	0		
<i>SDRAM_128M_64_DDR2_RAS_n_pin</i>	SDRAM_128M_64_DDR2_RAS_n	0		
<i>SDRAM_128M_64_DDR2_Clk_n_pin</i>	SDRAM_128M_64_DDR2_Clk_n	0	CLK	[1:0]
<i>SDRAM_128M_64_DDR2_DM_pin</i>	SDRAM_128M_64_DDR2_DM	0		[7:0]
<i>FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_OEN	0		[0:1]
<i>FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_CEN	0		[0:1]
<i>FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_WEN	0		
<i>FLASH_32M_16_SRAM_2M_32_Mem_RPN_pin</i>	FLASH_32M_16_SRAM_2M_32_Mem_RPN	0		
<i>sys_rst_pin</i>	sys_rst_s	1	RST	

Figura 3-31 : Puertos externos  
Fuente: Propia (2010)

El archivo UCF proporciona información que relaciona los puertos externos del proyecto a pines determinados por el usuario.

Además, el archivo UCF especifica estándares para cada pin tal como el tipo de voltaje y la frecuencia del reloj de entrada.

La sintaxis para realizar la asignación de pines es la siguiente:

- NET (Nombre asignado al Pin) LOC = (Identificador del PIN)

La sintaxis para realizar la asignación de voltaje es la siguiente:

- NET (Nombre asignado al Pin) IOSTANDARD = (Estándar de Voltaje)

También se pueden definir ambos de la siguiente forma:

- NET (Nombre asignado al Pin) LOC = (Identificador del PIN) | IOSTANDARD = (Estándar de Voltaje)

La *Figura 3-32* muestra un extracto del archivo UCF.

```

Net sys_clk_pin LOC=AL27 | IOSTANDARD = LVCMOS25; #ok
Net sys_rst_pin LOC=L14 | IOSTANDARD = LVCMOS25; #ok
Net sram_clk_pin LOC=K29 | IOSTANDARD = LVCMOS25; #ok
## Reloj del sistema
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 10000 ps;
Net sys_rst_pin TIG;

## Dispositivos IO

#### Modulo Ethernet_10_100

Net xps_ethernetlite_0_PHY_col_pin LOC="AE33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_crs_pin LOC="AV39" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_en_pin LOC="AU38" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_clk_pin LOC="AE35" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<0> LOC="AP35" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<1> LOC="AN34" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<2> LOC="AM36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<3> LOC="AH36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_er_pin LOC="AU36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_clk_pin LOC="AT32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<0> LOC="AR32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<1> LOC="AN33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<2> LOC="AR33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<3> LOC="AK33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_dv_pin LOC="AU32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rst_n_pin LOC="AV16" | IOSTANDARD = LVCMOS25 ;

Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<23> LOC=AR17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<24> LOC=AP16 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<25> LOC=K18 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<26> LOC=K17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<27> LOC=G18 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<28> LOC=G17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<29> LOC=AP22 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<30> LOC=AR22 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<31> LOC=AT22 | IOSTANDARD = LVCMOS25; #ok

Net FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin LOC=L29; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<1> LOC=AN28; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<1> IOSTANDARD = LVCMOS25;
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<0> LOC=M14; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<0> IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<1> LOC=AM27; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<1> IOSTANDARD = LVCMOS25;
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<0> LOC=AP30; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<0> IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN_pin LOC=K28; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN_pin IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<0> LOC=AM28 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<1> LOC=AM14 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<2> LOC=AM13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<3> LOC=AN29 | IOSTANDARD = LVCMOS25;#ok

Net FLASH_32M_16_SRAM_2M_32_Mem_RPN_pin LOC=AP13 | IOSTANDARD = LVCMOS25; #ok

```

Figura 3-32 : Extracto de archivo .ucf  
Fuente: Propia (2010)

### 3.18. Tarjeta de Red

Es necesario seleccionar un puerto disponible Hapstrack II de la tarjeta HAPS-51 para realizar el acoplamiento entre esta y la tarjeta adicional GEPHY Ethernet.

Los puertos Hapstrack II, están distribuidos alrededor del dispositivo FPGA en la tarjeta HAPS, como se aprecia en la *Figura 3-33*. Estos puertos constan de 119 pines de comunicaciones, los que además pueden ser seleccionados para operar a 3.3, 2.5 y 1.8 volts.

Los puertos A7 y A8 tienen asignado el voltaje de alimentación de salida fijo en 2.5 volts, los cuales se utilizan fundamentalmente para realizar una comunicación inter-dispositivos FPGA. Estos puertos no son requeridos en este proyecto.

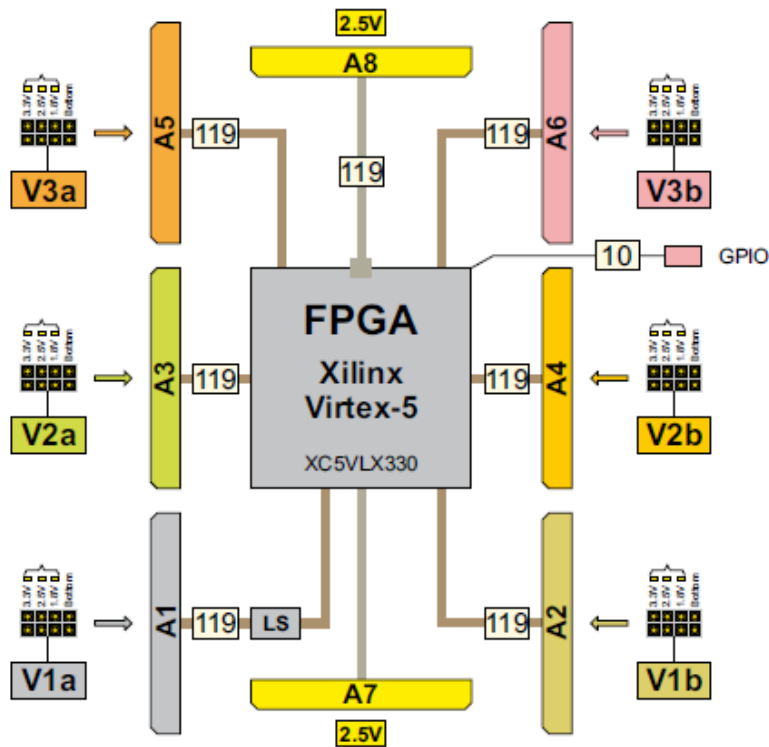


Figura 3-33 : Puertos Haps-51  
Fuente: Synopsys.com (2010)

El puerto seleccionado para incorporar la tarjeta GEPHY será el A2 en base a la comodidad que presenta físicamente una vez integrada a la tarjeta.

En la *Figura 3-34* se muestra una tarjeta GEPHY compatible con la HAPS, debido a que dispone de un puerto Hapstrack II.

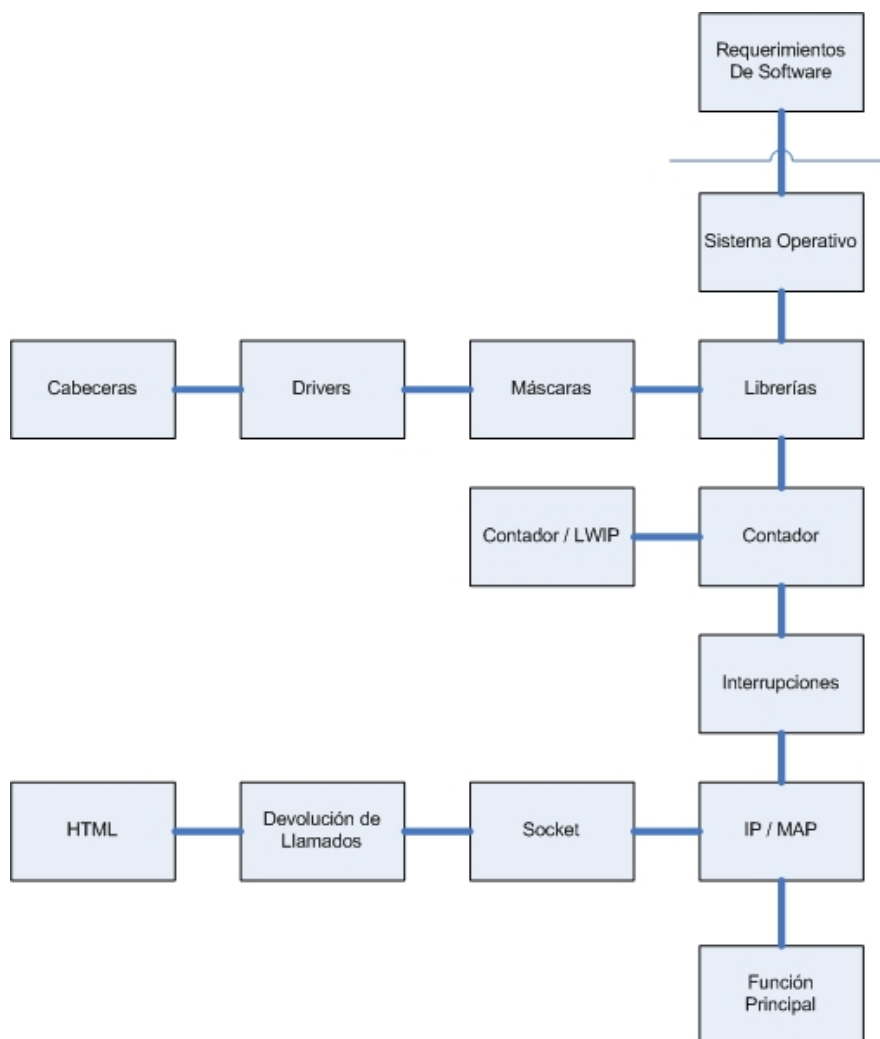


*Figura 3-34 : GEPHY*  
*Fuente: Synopsys.com (2010)*

## 4. Desarrollo Software

### 4.1. Resumen de Actividades

Las tareas y actividades para la implementación del Software se basan en la *Figura 4-1*, la cual indica las diferentes etapas a seguir para completar el servidor HTML. El objetivo de estas actividades es crear el software necesario para el correcto funcionamiento del servidor.



*Figura 4-1 : Diagrama de actividades de software  
Fuente: Propia (2010)*

La descripción de las actividades es la siguiente:

- Sistema Operativo:

Decidir la posible utilización de un sistema operativo para dar soporte al software del servidor HTML.

- Librerías:

Seleccionar las librerías a utilizar para el desarrollo del software, a partir de la lista disponible en la herramienta XPS.

- Máscaras:

Seleccionar e Incorporar las máscaras (de la herramienta XPS) para facilitar la programación del software.

- Drivers:

Generar e inicializar los drivers, con el propósito de controlar los módulos del proyecto.

- Cabeceras:

Incluir los archivos cabeceras al proyecto software.

- Contador:

Configurar el módulo contador para que ejecute instrucciones específicas cada período de tiempo determinado.

- Contador / LWIP:

Asociar la interrupción del contador, con instrucciones de software requeridas por la librería LWIP.

- Interrupciones:

Habilitar las interrupciones por software.

- IP / MAC:

Especificar la dirección IP y la dirección MAC del servidor.

- Socket TCP/IP RAW:

Implementar los eventos de comunicación necesarios para el servidor, por medio del Socket y las primitivas TCP/IP RAW.

- Devolución de Llamados (Callbacks):

Aprovechar el uso de las devoluciones de llamado, para enviar el contenido HTML al cliente desde el servidor creado.

- HTML:

Programar el código HTML que será proporcionado por el servidor a un cliente.

- Función Principal (Main):

Programar el código necesario de la función principal del software.

## **4.2. Sistema Operativo**

El manejo de la aplicación presenta la opción de usar o no sistema operativo. Las opciones son las siguientes:

- Standalone (versión 2.0)
- Xilkernel (versión 4.0)

Standalone es núcleo sin sistema operativo, sin embargo brinda manejo de la entrada y salida de datos por pantalla con terminal como RS-232 por dar un ejemplo. Xilkernel en cambio presenta manejo de multitareas, semáforos, y aplicaciones propias de un kernel.

La implementación se realizará con Standalone ya que brinda una solución simple al sistema y debido a que no requiere manejo de multitareas por tratarse de un servidor prototipo.

Lo fundamental de Standalone es que permitirá utilizar la salida estándar, que puede ser la pantalla de un computador conectado a la Virtex-5, para mostrar los datos o información al ir probando el sistema.

Si se fuera a utilizar un sistema operativo, se deberá modificar la lógica del proyecto. Esto corresponde a agregar un contador adicional al hardware programable y configurar el sistema operativo con propiedades tales como, habilitación o deshabilitación de semáforos, o la cantidad de hebras entre varias opciones. El contador cumplirá la función de administrar el tiempo de conmutación entre cada tarea.

### 4.3. Librerías

Las librerías a utilizar por el software serán seleccionadas a partir del listado proporcionado por la herramienta XPS. La lista de librerías disponibles se muestran en la *Figura 4-2*, nombradas en la columna Library.

Use	Library	Version	Description
<input type="checkbox"/>	xilmfs	1.00.a	Xilinx Memory File System
<input type="checkbox"/>	xilisf	1.00.a	Xilinx In-system and Serial Flash Library
<input type="checkbox"/>	xilflash	1.01.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
<input type="checkbox"/>	xilfatfs	1.00.a	Provides read/write routines to access files stored on a FAT16.
<input checked="" type="checkbox"/>	lwip	3.00.a	LwIP TCP/IP Stack library v3.00.a
<input type="checkbox"/>	lwip130	1.00.a	LwIP TCP/IP Stack library: lwIP v1.3.0, Xilinx adapter v1.00.a

*Figura 4-2 : Librerías disponibles para Microblaze*  
*Fuente: Xilinx EDK Versión 2005*

La librería que será seleccionada es la Light Weight Internet Protocol o LWIP, la cual proporciona control a nivel de sockets TCP/IP.

Las demás librerías proporcionan funciones para manejo de memoria Flash, manejo de Sistemas de Archivos, escritura y lectura a sistema de archivos específicos tipo FAT16 y librería lwip130, la cual proporciona manejo del protocolo de Internet, pero para adaptadores específicos de interfaz Ethernet.

Las librerías restantes no serán utilizadas ya que no son necesarias para alcanzar el objetivo.

#### **4.4. Máscaras**

Existe un archivo importante llamado `xparameters.h` el cual debe ser incorporado como cabecera al momento de desarrollar el software. Este es configurado a través del procedimiento de generación automática del mapa de direcciones para el procesador Microblaze.

Este archivo es un listado de la dirección del primer registro de cada módulo, lo cual brinda comodidad al momento de leer o escribir a los mismos, debido a que se especifica su ubicación a través de una máscara.

Para el caso de algunos módulos, `xparameters.h` proporciona información adicional para el control de registros específicos. Por dar un ejemplo, para el módulo controlador de interrupciones se proporciona una máscara para cada una de las interrupciones agregadas en la tabla de prioridades, lo que será práctico en la inicialización de la misma.

La *Figura 4-3*, *Figura 4-4* y *Figura 4-5* muestran un extracto del archivo `xparameters.h`, para las interrupciones, el contador y el controlador de memoria principal respectivamente.

```

/* Definitions for peripheral XPS_INTC_0 */
#define XPAR_XPS_INTC_0_DEVICE_ID 0
#define XPAR_XPS_INTC_0_BASEADDR 0x81800000
#define XPAR_XPS_INTC_0_HIGHADDR 0x8180FFFF
#define XPAR_XPS_INTC_0_KIND_OF_INTR 0x00000005

/*****/

#define XPAR_INTC_SINGLE_BASEADDR 0x81800000
#define XPAR_INTC_SINGLE_HIGHADDR 0x8180FFFF
#define XPAR_INTC_SINGLE_DEVICE_ID XPAR_XPS_INTC_0_DEVICE_ID
#define XPAR_XPS_ETHERNETLITE_0_IP2INTC_IRPT_MASK 0X000001
#define XPAR_XPS_INTC_0_XPS_ETHERNETLITE_0_IP2INTC_IRPT_INTR 0
#define XPAR_XPS_TIMER_0_INTERRUPT_MASK 0X000002
#define XPAR_XPS_INTC_0_XPS_TIMER_0_INTERRUPT_INTR 1
#define XPAR_DEBUG_MODULE_INTERRUPT_MASK 0X000004
#define XPAR_XPS_INTC_0_DEBUG_MODULE_INTERRUPT_INTR 2

/*****/

```

*Figura 4-3 : Xparameters interrupciones  
Fuente: Propia (2010)*

Extracto del contenido, para el Contador:

```

/* Definitions for peripheral XPS_TIMER_0 */
#define XPAR_XPS_TIMER_0_DEVICE_ID 0
#define XPAR_XPS_TIMER_0_BASEADDR 0x83C00000
#define XPAR_XPS_TIMER_0_HIGHADDR 0x83C0FFFF

```

*Figura 4-4 : Xparameters contador  
Fuente: Propia (2010)*

Extracto del contenido para la memoria SDRAM:

```

/* Definitions for peripheral SDRAM_128M_64 */
#define XPAR_SDRAM_128M_64_DEVICE_ID 0
#define XPAR_SDRAM_128M_64_MPMC_BASEADDR 0xC0000000
#define XPAR_SDRAM_128M_64_MPMC_CTRL_BASEADDR 0xFFFFFFFF
#define XPAR_SDRAM_128M_64_INCLUDE_ECC_SUPPORT 0
#define XPAR_SDRAM_128M_64_USE_STATIC_PHY 0
#define XPAR_SDRAM_128M_64_PM_ENABLE 0
#define XPAR_SDRAM_128M_64_NUM_PORTS 3

```

*Figura 4-5 : Xparameters memoria principal  
Fuente: Propia (2010)*

## 4.5. Drivers

Se debe configurar el tipo de driver a utilizar para cada periférico o módulo que posee comunicación con el procesador Microblaze. La inicialización de los drivers se puede realizar mediante el archivo de extensión \*.MSS.

Esta inicialización radica en especificar en cada uno de los módulos, la siguiente información:

- Nombre del Driver
- Versión del Driver
- Tipo de módulo

Un extracto del archivo .mss generado se muestra en la *Figura 4-6*, en el que se presentan los drivers del módulo Contador, Ethernet y el Controlador de Interrupciones.

```
BEGIN DRIVER
PARAMETER DRIVER_NAME = tmrcetr
PARAMETER DRIVER_VER = 1.10.b
PARAMETER HW_INSTANCE = xps_timer_0
END

BEGIN DRIVER
PARAMETER DRIVER_NAME = emaclite
PARAMETER DRIVER_VER = 1.13.a
PARAMETER HW_INSTANCE = xps_ethernetlite_0
END

BEGIN DRIVER
PARAMETER DRIVER_NAME = intc
PARAMETER DRIVER_VER = 1.11.a
PARAMETER HW_INSTANCE = xps_intc_0
END
```

*Figura 4-6 : .MSS de contador, Ethernet e Interrupciones*  
*Fuente: Propia (2010)*

## 4.6. Cabeceras

Antes de comenzar a implementar el código de la aplicación, es necesario incluir algunos archivos cabeceras en este mismo. Estos archivos pueden incluir bibliotecas, drivers, archivo de macros y librerías estándar de programación en C entre algunas cosas. En la *Figura 4-7* se muestran los archivos cabeceras que fueron incluidos en el proyecto.

```
#include "xparameters.h"
#include "netif/xadapter.h"
#include "xintc.h"
#include "xtrmctr_l.h"
#include "lwip\tcp.h"
#include "lwip\udp.h"
#include <stdio.h>
```

*Figura 4-7 : Archivos cabecera incluidos  
Fuente: Propia (2010)*

Cada uno tiene el siguiente propósito dentro del proyecto:

- Xparameters.h: Máscaras de mapeo de memoria.
- Xintc.h: Driver del controlador de interrupciones
- Xtrmctr\_l.h: Driver del Timer / Contador
- Netif/xadapter.h: Driver de la interfaz ethernet
- Lwip\tcp.h: Biblioteca de funciones TCP de LWIP
- Lwip\udp.h: Biblioteca de funciones UDP de LWIP
- Stdio.h: Biblioteca estándar de programación en C

## 4.7. Contador

La implementación requerida para el Contador es la de configurar su modo de funcionamiento y de especificar el tiempo que tardará en contar.

### COMPRENSIÓN DEL DRIVER

Para lo anterior se utilizará el Driver en donde se utilizaron las siguientes funciones:

- XTmrCtr\_mSetControlStatusReg
- XTmrCtr\_mSetLoadReg

Donde sus parámetros son:

- void XTmrCtr\_mSetLoadReg(u32 BaseAddress, u8TmrCtrNumber,u32 RegisterValue);
- void XTmrCtr\_mSetControlStatusReg(u32 BaseAddress, u8TmrCtrNumber,u32 RegisterValue);

Estos parámetros corresponden a:

- BaseAddress: Es la dirección base del módulo.
- TmrCtrNumber: Corresponde al identificador del Contador a usar. Esto es debido a que para cada módulo Contador, se dispone de un total de dos contadores a utilizar. En este caso es necesario uno solo.
- RegisterValue: Es la configuración que se le quiere dar al hardware del contador. Para asignar estos valores se puede hacer uso de máscaras de control, ubicadas en el mismo driver.

Las máscaras utilizadas son:

La lista de máscaras de control disponibles a ser utilizadas en el parámetro Register Value se muestran en la *Figura 4-8*. Algunas de las configuraciones que proporcionan estas máscaras son las siguientes:

- Habilitar el uso general del módulo Contador
- Habilitar todas las configuraciones del módulo Contador
- Habilitar el uso de la interrupción
- Habilitar la cuenta descendente
- Habilitar la generación de señal externa cada vez que finaliza la cuenta
- Cargar el registro que realiza la cuenta con el número inicial configurado
- Habilitar el modo de autoregenerar la cuenta una vez finalizada

```
#define XTC_CSR_ENABLE_ALL_MASK
#define XTC_CSR_ENABLE_PWM_MASK
#define XTC_CSR_INT_OCCURED_MASK
#define XTC_CSR_ENABLE_TMR_MASK
#define XTC_CSR_ENABLE_INT_MASK
#define XTC_CSR_LOAD_MASK
#define XTC_CSR_AUTO_RELOAD_MASK
#define XTC_CSR_EXT_CAPTURE_MASK
#define XTC_CSR_EXT_GENERATE_MASK
#define XTC_CSR_DOWN_COUNT_MASK
#define XTC_CSR_CAPTURE_MODE_MASK
```

*Figura 4-8 : Máscaras de configuración de contador*  
*Fuente: Xilinx EDK Versión 2005*

## FORMULA DEL INTERVALO DE CUENTA

El intervalo de cuenta, corresponde al tiempo que transcurre desde que se comienza a contar hasta que se alcanza el valor definido como final. Este intervalo se define de acuerdo a los requerimientos que en cada proyecto se tengan.

Para determinar el intervalo de cuenta se necesita conocer el periodo del Reloj conectado al Bus principal. Luego se utiliza la Fórmula descendente o ascendente dependiendo de la configuración.

- Fórmula para cuentas descendentes:

$$\text{TIMING\_INTERVAL} = (\text{TLRx}) \times \text{PLB\_CLOCK\_PERIOD}$$

- Fórmula para cuentas ascendentes:

$$\text{TIMING\_INTERVAL} = (\text{MAX\_COUNT} - \text{TLRx}) \times \text{PLB\_CLOCK\_PERIOD}$$

Donde:

MAX\_count es el máximo valor permitido para el registro. Para el caso de ser un registro de 32 bits el valor máximo permitido sería igual FFFFFFFF hexadecimal

## IMPLEMENTACIÓN

El Contador se implementó con la función `TIMER_HANDLER` mostrada en la *Figura 4-9*.

Esto corresponde a:

- Habilitar el contador para que cuente.
- Modo de Cuenta descendente.
- Habilitar las interrupciones cada vez que se complete la cuenta asociada al número del registro Load.
- Volver a contar automáticamente hasta el mismo número anterior, luego de terminada la cuenta.
- Intervalo de 250 mili segundos
- Auto recarga del valor a contar en el registro Load.

El intervalo de tiempo se producirá cuando el contador comience a contar regresivamente desde el valor 25000000 hasta 0. Recordar para este proceso que la frecuencia del bus PLB es de 100 MHz, siendo la velocidad que aplica en este proceso.

```
void timer_handler(void *p)
{
    static int odd = 1;
    tcp_fasttmr();
    odd = !odd;
    if (odd)
        tcp_slowtmr();

    /*Set Load Registro*/
    XTmrCtr_mSetLoadReg(XPAR_XPS_TIMER_0_BASEADDR, 0, 25000000);

    /* Cargar el registro del timer, resetear el flag de interrupciones */
    XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_0_BASEADDR, 0, XTC_CSR_INT_OCCURED_MASK |
    XTC_CSR_LOAD_MASK);

    /* Habilitar el contador , Habilitar interrupciones, cuenta descendente, modo autorecarga */
    XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_0_BASEADDR, 0, XTC_CSR_ENABLE_TMR_MASK |
    XTC_CSR_ENABLE_INT_MASK
    | XTC_CSR_AUTO_RELOAD_MASK | XTC_CSR_DOWN_COUNT_MASK);

    XIntc_mAckIntr(XPAR_INTC_0_BASEADDR, XPAR_XPS_TIMER_0_INTERRUPT_MASK);
}
```

*Figura 4-9 : Función Timer Handler*  
*Fuente: Propia (2010)*

#### **4.8. Contador/LWIP**

Una vez inicializada la biblioteca TCP de LWIP, por medio del llamado a la función LWIP\_INIT, el programa necesita invocar permanentemente las funciones TCP\_FASTTMR y TCP\_SLOWTMR para limpiar la memoria dinámica del sistema.

TCP\_FASTTMR y TCP\_SLOWTMR deben ser llamadas cada intervalos de tiempo especificados por la librería LWIP. La primera de ellas se invoca cada 250 milisegundos generados directamente por el módulo contador, y la segunda cada 500 milisegundos implementados por software y aumentando en dos veces el intervalo anterior.

La Figura 4-9 muestra como se realiza la implementación del llamado a TCP\_FASTTMR y TCP\_SLOWTMR, al principio de la aplicación.

## 4.9. Interrupciones

La función PLATFORM\_SETUP\_INTERRUPTS es desarrollada en este proyecto para la puesta en marcha del Controlador de Interrupciones. En esta función se realizaron varias tareas dentro de las cuales está la inicialización del driver, la asociación de cada una de las señales de interrupción con una función, la habilitación o deshabilitación de cada interrupción y la habilitación o deshabilitación global del Controlador.

Las actividades de la implementación fueron las siguientes:

- Funciones utilizadas
- Parámetros de las funciones
- Explicación de los parámetros
- Implementación

### FUNCIONES UTILIZADAS

- Xintc\_RegisterHandler: Asignación de tabla de interrupciones
- Xintc\_mEnableIntr: Habilitación o Deshabilita interrupción

### PARAMETROS DE LAS FUNCIONES

- Void Xintc\_RegisterHandler (u32 BaseAddress, int InterruptId, XinterruptHandler Handler, void CallbackRef);
- Void Xintc\_mEnableIntr (u32 BaseAddress, u32 EnableMask);

## EXPLICACIÓN DE PARÁMETROS

- BaseAddress: Dirección base del controlador de interrupciones
- Interrupt ID: Identificador de la interrupción según Prioridad
- XinterruptHandler Handler: Evento asociado a interrupción
- CallbackRef: Sub evento asociado a interrupción
- EnableMask: Máscara que guarda el Identificador de la Interrupción sin Prioridad.

## IMPLEMENTACIÓN

La implementación acometida se aprecia en la *Figura 4-10*. La explicación corresponde a lo siguiente:

- La función es Timer\_Handler, fue ligada a la interrupción que genera el módulo Contador.
- Se habilitó la interrupción del módulo Contador.
- Se habilitó la interrupción Ethernet, que corresponde a la llegada de algún dato al buffer del módulo Controlador de Ethernet. Esta interrupción no necesita ser asociada a algún evento, ya que el evento se manejará por medio de un Socket.

La función se muestra en la *Figura 4-10*.

```
void platform_setup_interrupts()
{
    XIntc *intcp; // ok
    intcp = &intc;

    XIntc_Initialize(intcp, XPAR_XPS_INTC_O_DEVICE_ID);
    XIntc_Start(intcp, XIN_REAL_MODE); // ok

    /* habilitar el controlador de interrupciones */
    XIntc_mMasterEnable(XPAR_XPS_INTC_O_BASEADDR); // ok

    /* Registrar la funcion Timer handler a la tabla de interrupciones*/
    XIntc_RegisterHandler(XPAR_XPS_INTC_O_BASEADDR, XPAR_XPS_INTC_O_XPS_TIMER_O_INTERRUPT_INTR,
        (XInterruptHandler)timer_handler,
        0);

    XIntc_mEnableIntr(XPAR_XPS_INTC_O_BASEADDR, XPAR_XPS_TIMER_O_INTERRUPT_MASK |
        XPAR_XPS_ETHERNETLITE_O_IP2INTC_IRPT_MASK);

    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHandler, intcp);
    microblaze_enable_interrupts();
}
```

*Figura 4-10 : Función platform setup interrupts*  
*Fuente: Propia (2010)*

## 4.10. IP/MAC

Es necesario definir el valor de la dirección IP y de la dirección MAC, asociando estos valores a la interfaz GEPHY. Esto se realiza mediante la función XEMAC\_ADD, la cual requiere de 6 parámetros de configuración, los cuales corresponden a:

- Identificador para la Interfaz a crear
- Dirección IP
- Máscara de Red
- Puerta de Enlace
- Dirección MAC
- Dirección base del módulo controlador Ethernet a utilizar

Con el identificador de esta interfaz creada, se puede levantar la interfaz y dejarla por defecto para ser reconocida al trabajar.

Los requerimientos de implementación implican dejar ejecutando en un ciclo infinito la función XEMACIF\_INPUT, la cual verifica constantemente si ha llegado información al buffer de entrada de la interfaz. La implementación se aprecia en la *Figura 4-11*.

```
int main()
{
    struct netif *netif;
    struct ip_addr ipaddr, netmask, gw;
    unsigned char mac_ethernet_address[] = { 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };

    /* Variables de networking */
    IP4_ADDR(&ipaddr, 192, 168, 1, 20);
    IP4_ADDR(&netmask, 255, 255, 255, 0);
    IP4_ADDR(&gw, 192, 168, 1, 254);

    /* Agregar la interfaz*/
    if (!xemac_add(netif, &ipaddr, &netmask, &gw, mac_ethernet_address, XPAR_XPS_ETHERNETLITE_0_BASEADDR)) {
        xil_printf("Error adding N/W interface\n\r");
        return -1;
    }
    netif_set_default(netif);
    netif_set_up(netif);

    while (1) {
        xemacif_input(netif);
    }
}
```

*Figura 4-11 : IP / MAC  
Fuente: Propia (2010)*

## **4.11. Socket TCP/IP RAW**

Para implementar un Socket se utiliza el concepto de primitivas. Las primitivas son genéricas y dentro de estas tenemos New, Bind, Listen y Accept entre otras.

Las primitivas requeridas deben ser inicializadas, lo que se realiza por medio de funciones específicas proporcionada por la librería. En nuestro caso se emplea LWIP\_TCP.h declarada previamente.

### LISTA DE PRIMITIVAS

- Tcp\_New
- Tcp\_Bind
- Tcp\_Arg
- Tcp\_Listen
- Tcp\_Accept
- Tcp\_write
- Tcp\_close

### TCP NEW

La primitiva TCP\_NEW, devuelve una estructura llamada pcb que inicializa un socket sin ser configurado.

### TCP BIND

La primitiva TCP\_BIND realiza un enlace de la estructura pcb, con un Puerto y una dirección IP determinada. El Puerto seleccionado es el 80, y será aceptada cualquier dirección IP a ese Puerto.

## TCP ARG

La primitiva TCP\_ARG no tiene mayor relevancia en este proyecto, pero es necesario inicializarla con un parámetro NULO.

## TCP LISTEN

La primitiva TCP\_LISTEN configura la estructura PCB para trabajar como Servidor. Eso quiere decir que aceptará peticiones de conexión provenientes de los clientes. Según la configuración previa de la primitiva Bind, los clientes podrán poseer cualquier dirección IP, pero el servicio aceptado será solamente el proporcionado por el Puerto 80.

## TCP ACCEPT

Finalmente la primitiva TCP\_ACCEPT especifica cual función o evento será ejecutado inmediatamente después que se acepte una solicitud de conexión desde un cliente cualquiera.

La función que será llamada luego que sea aceptada una conexión por parte de un cliente será HTTP\_ACCEPT.

## IMPLEMENTACION

Este conjunto de eventos se agrupó por medio de la función SETEO DE SOCKET mostrada en la *Figura 4-12*, la cual muestra la implementación de las primitivas y sus configuraciones asignadas.

```

int seteo_de_soquet()
{
    struct tcp_pcb *pcb;
    struct ip_addr ipaddr2;
    unsigned port = 80;
    err_t err;
    err_t err3;

    /* Declaracion de estructura TCP*/
    pcb = tcp_new();
    if (!pcb) {
        xil_printf("Error creando la estructura TCP. Falta de memoria\n\r");
        return -1;
    }

    /* Bind de estructura TCP con puerto 80*/
    err = tcp_bind(pcb, IP_ADDR_ANY, port);
    if (err != ERR_OK) {
        xil_printf("Error en la asignacion de Puerto",port, err);
        return -2;
    }

    /* No se necesitan argumentos */
    tcp_arg(pcb, NULL);
    /* Setear en estado Listen */
    pcb = tcp_listen(pcb);
    if (!pcb) {
        xil_printf("Error de Listen. Falta de memoria\n\r");
        return -3;
    }

    /* Funcion que se llama cuando se aceptan conexiones externas */
    tcp_accept(pcb, http_accept);
    xil_printf("Tcp/IP inicializado en Puerto %d\n\r", port);

    return 0;
}

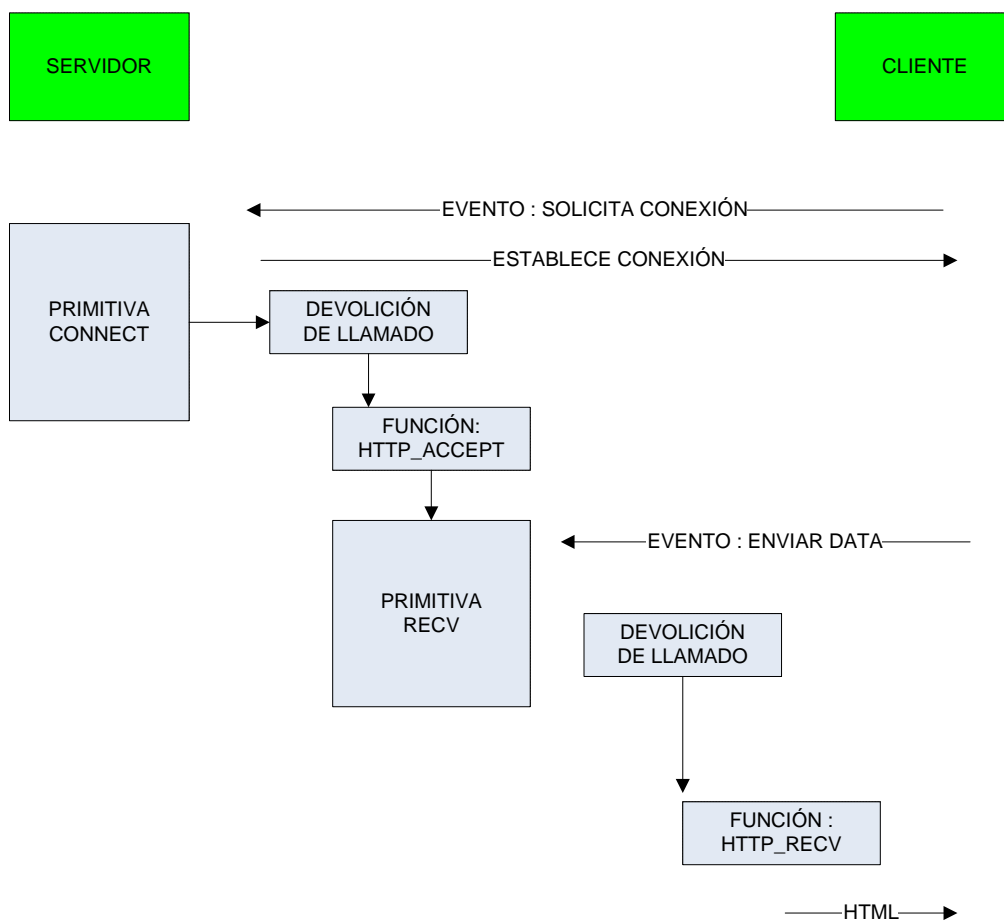
```

*Figura 4-12 : Función seteo del socket  
Fuente: Propia (2010)*

## 4.12. Devolución de Llamados (Callbacks)

La librería LWIP y las propiedades de un Socket Raw, tienen la característica de poseer primitivas que ligan ciertos eventos de comunicación tales como conexión, recepción de datos y desconexión de parte de un cliente, con la ejecución inmediata de ciertas funciones. Esta propiedad se llama Devoluciones de Llamado o Callbacks.

Según los propósitos de este proyecto, las devoluciones de llamados serán utilizadas para eventos de conexión y recepción de datos, en relación a las primitivas Accept y Recv. La *Figura 4-13* modela la implementación de este procedimiento desarrollado en el software.



*Figura 4-13 : Devolución de Llamados y Primitivas*  
Fuente: Propia (2010)

Una vez que sea solicitada una conexión por parte del cliente al servidor y esta es aceptada, la primitiva Connect llamará inmediatamente a la función HTTP\_ACCEPT (Figura 4-14). Esta función inicializará de paso la primitiva TCP\_RECV, la que a su vez establecerá como devolución de llamado a la función HTTP\_RECV (Figura 4-15).

```

/* Esta es la funcion callback que se genera al recibir una conexion */
static void http_accept(void *arg, struct tcp_pcb *pcb)
{
    /* Esta es la declaracion de otra funcion callback */
    tcp_recv(pcb, http_recv);
    xil_printf("Conectado con cliente\r\n");
}

```

*Figura 4-14 : Función HTTP ACCEPT  
Fuente: Propia (2010)*

Cuando se reciba un dato a partir del Cliente conectado con el Servidor, la función HTTP\_RECV será ejecutada inmediatamente por ser parte de la devolución de llamado de TCP\_RECV y en caso de que la data recibida corresponda al código hexadecimal "GETS/", entonces la misma función se encargará de mandar el contenido HTML al Cliente que solicito la información.

En caso de no recibir "GETS/", el Socket será cerrada la conexión."

```

/* Esta funcion se genera automaticamente al recibir datos en el buffer del soquet pcb */
static void http_recv(void *arg, struct tcp_pcb *pcb, struct pbuf *p)
{
    char *rq;
    /* En caso de que se reciva un dato nulo se cierra la conexion */
    if(p != NULL) {
        /* El buffer se iguala a una variable */
        rq = p->payload;
        /* Revisar si es que se revibe los caracteres "GET /\r\n". */
        if(rq[0] == 'G' && rq[1] == 'E' && rq[2] == 'T' && rq[3] == ' ' &&
            rq[4] == '/') {
            /* Una vez obtenido el GET entonces enviar la pagina web */
            xil_printf("GET obtenido\r\n");
            tcp_write(pcb, indexdata, sizeof(indexdata), 1);
        }
        /* Liberar el puntero */
        pbuf_free(p);
    }
    /* Cerrar la conexion */
    tcp_close(pcb);
    xil_printf("Desconectado\r\n");
}

```

*Figura 4-15 : Función HTTP RECV  
Fuente: Propia (2010)*

### 4.13. HTML

La variable `indexdata` es un vector tipo `string`, el cual contiene el código HTML a enviar al Cliente que solicita la información a través del “GETS/”. Como mínimo al inicio del código, se debe especificar que el navegador interprete los datos como código HTML. Posteriormente puede ser agregado el título y la información relevante a mostrar según sea el caso. La *Figura 4-16* muestra el contenido HTML definido en un vector que almacena el contenido HTML a enviar por `TCP_WRITE`.

```
/* Este es el contenido estatico de la pagina HTML. */
static char indexdata[] =
"HTTP/1.0 200 OK\r\n\
Content-type: text/html\r\n\
\r\n\
<html> \
<head><title>PROYECTO DESARROLLO ETHERNET</title></head> \
<body> \
----- \
<br> \
<H1>SERVIDOR HTML DE FPGA VIRTEX-5</H1> \
<br> \
----- \
</body> \
</html>";
```

*Figura 4-16 : Contenido HTML*  
*Fuente: Propia (2010)*

## 4.14. Función principal (Main)

En la función principal se ejecutan las primeras instrucciones de la aplicación. Estas acciones deberán incluir inicialización de variables, llamado a funciones y configuración de hardware. La *Figura 4-17* muestra la función Main.

La lista de instrucciones implementadas son las siguientes:

- Definición del tamaño de la memoria cache para datos
- Definición del tamaño de la memoria cache para instrucciones
- Habilitación del cache de instrucciones
- Inicializar Lwip TCP

Las funciones llamadas en la función Main son las siguientes:

- Llamado de la función de Configuración de Interrupciones
- Llamado de la función Implementación del Socket

```
int main()
{
    struct netif *netif;
    struct ip_addr ipaddr, netmask, gw;
    unsigned char mac_ethernet_address[] = { 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };

    microblaze_init_icache_range(0, XPAR_MICROBLAZE_O_CACHE_BYTE_SIZE);
    microblaze_init_dcachecache_range(0, XPAR_MICROBLAZE_O_DCACHE_BYTE_SIZE);

    XCACHE_ENABLE_ICACHE();
    platform_setup_interrupts();

    /* Variables de networking */
    IP4_ADDR(&ipaddr, 192, 168, 1, 20);
    IP4_ADDR(&netmask, 255, 255, 255, 0);
    IP4_ADDR(&gw, 192, 168, 1, 254);

    lwip_init();

    /* Agregar la interfaz*/
    if (!xemac_add(netif, &ipaddr, &netmask, &gw, mac_ethernet_address, XPAR_XPS_ETHERNETLITE_O_BASEADDR)) {
        xil_printf("Error adding N/W interface\n\r");
        return -1;
    }
    netif_set_default(netif);
    netif_set_up(netif);
    seteo_de_soquet();

    while (1) {
        xemacif_input(netif);
    }
    XCACHE_DISABLE_DCACHE();
    XCACHE_DISABLE_ICACHE();
    return 0;
}
```

*Figura 4-17 : Función Main*  
*Fuente: Propia (2010)*

## 5. Pruebas

Es necesario efectuar la evaluación del sistema mediante pruebas de comunicación para medir el comportamiento del prototipo, su estabilidad y su capacidad de carga máxima para proporcionar un buen servicio. La estabilidad se refiere al correcto envío y recepción de los datos

A continuación se presentan las pruebas consideradas de mayor importancia, ya que involucran los aspectos de comunicación antes mencionados. Se pueden realizar otras pruebas al sistema, como la integración en alguna planta industrial para la supervisión de variables, sin embargo tal actividad de integración esta fuera del alcance de este proyecto.

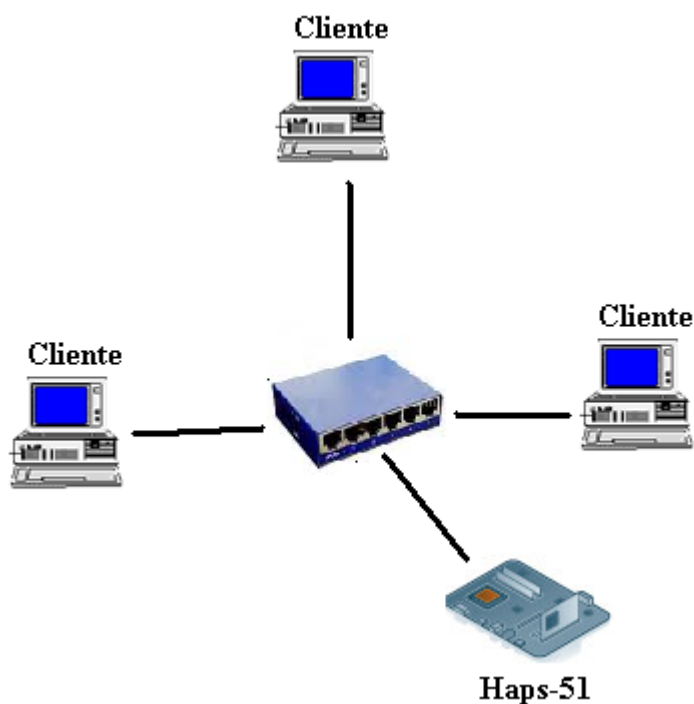
### ELEMENTOS REQUERIDOS

Las herramientas de apoyo utilizadas las siguientes:

- Mozilla FireFox  
*(Interprete de datos HTML)*
- 3 PCs de Prueba:  
*(Clientes)*
- Tarjeta HAPS-51 con Firmware Embebido  
*(Proyecto Firmware cargado en Haps-51)*
- Switch Ethernet 10/100 MBPS  
*(Concentrador)*
- Cables de Red Ethernet

## CONFIGURACION BÁSICA

Los elementos se ordenan como se muestra la *Figura 5-1*. En esta se tienen los 3 computadores conectados al Switch por los cables de red Ethernet, al igual que la HAPS-51.



*Figura 5-1: Configuración de Prueba  
Fuente: Propia (2010)*

Los PC deben configurarse con los siguientes parámetros:

- Cliente 1
  - IP 192.168.1.100
  - Máscara 255.255.255.0
  - Puerta de Enlace 192.168.1.254
  
- Cliente 2
  - IP 192.168.1.101
  - Máscara 255.255.255.0
  - Puerta de Enlace 192.168.1.254

- Cliente 3
  - IP 192.168.1.102
  - Máscara 255.255.255.0
  - Puerta de Enlace 192.168.1.254

## PROCEDIMIENTOS

### PRUEBA 1 (Evaluación de comportamiento y estabilidad)

- El procedimiento seguido fue haber ejecutado la aplicación Mozilla FireFox en los 3 clientes y solicitar el archivo HTML a la dirección 192.168.1.20.  
Posteriormente deberá aparecer el contenido del archivo HTML en cada uno de los clientes.

### PRUEBA 2 (Evaluación de comportamiento, estabilidad y carga máxima)

- Seleccionar un cliente y ejecutar 5 aplicaciones Mozilla FireFox simultáneamente, solicitando el archivo HTML a la dirección 192.168.1.20. Esperar recibir el contenido en las 5 aplicaciones Mozilla abiertas.
- Realizar el procedimiento anterior ejecutando 10 aplicaciones.
- Realizar el procedimiento anterior ejecutando 15 aplicaciones.
- Realizar el procedimiento anterior ejecutando 20 aplicaciones.
- Realizar el procedimiento anterior ejecutando 21 aplicaciones.

## 6. Resultados

El resultado de las pruebas realizadas se muestra a continuación:

### PRUEBA 1

- Se recibió el contenido HTML en el Mozilla del Cliente 1
- Se recibió el contenido HTML en el Mozilla del Cliente 2
- Se recibió el contenido HTML en el Mozilla del Cliente 3

### PRUEBA 2

- Se recibió todo el contenido HTML ejecutando 5 peticiones
- Se recibió todo el contenido HTML ejecutando 10 peticiones
- Se recibió todo el contenido HTML ejecutando 15 peticiones
- Se recibió todo el contenido HTML ejecutando 20 peticiones
- No se recibió todo el contenido HTML ejecutando 21 peticiones. Falló la recepción en una de las aplicaciones Mozilla.

Los resultados han sido satisfactorios para todas las pruebas realizadas. Esto radica en que en las Pruebas 1 se logró establecer una comunicación fiable y un comportamiento estable de envío y recepción de data entre los tres clientes involucrados y el servidor embebido. Los tiempos de respuesta del sistema en este caso son del orden de los milisegundos.

Debido a que la Prueba 2 mide la capacidad de carga máxima del servicio para enviar información a los clientes y evalúa su estabilidad en los procesos de alta demanda, se evidenció un comportamiento estable dentro 20 peticiones. El tráfico a este nivel corresponde a 80 Kbytes lo que establece la carga máxima. Los tiempos de respuesta del sistema en este caso son del orden de los milisegundos.

## 7. Conclusiones

Este proyecto realizó un prototipo de servidor HTML sobre una FPGA Virtex-5 embebida con un Firmware basado en HTTP y TCP/IP. Se analizó el problema, se plantearon requerimientos y se realizaron las actividades de hardware y software mencionadas. Se finalizó realizando las pruebas indicadas para probar el funcionamiento, la estabilidad y capacidad máxima del sistema.

Uno de los aportes de este proyecto está en la experiencia adquirida en el uso de FPGAs de Xilinx, la cual deberá ser considerada para realizar desarrollo de sistemas basados en comunicaciones Ethernet, ya que se comprendió su tendencia en la industria tecnológica.

Además, el aporte es un dispositivo embebido que podrá ser integrado a diversos sistemas electrónicos, dado su alto nivel de adaptabilidad. Un uso útil es adaptar la tarjeta como unidad de control para una planta industrial determinada o para la supervisión de variables.

Con respecto al Firmware desarrollado, este está programado sobre una FPGA Virtex-5 LX330 la cual posee bastante holgura para almacenar mayor programación de lógica, tal como la futura incorporación de procesadores adicionales Microblaze u otros módulos. Debido a esta alta capacidad se recomienda también continuar utilizando una Virtex-5 para mantener el rendimiento.

De esta tecnología se puede rescatar su gran capacidad para crear sistemas de lógica compleja, al quedar demostrado como es posible emular el comportamiento de arquitectura de computadores mediante la incorporación de un procesador Microblaze, memorias y contador entre muchos otros. En comparación con otros dispositivos, los microcontroladores tienen su capacidad de lógica o instrucciones limitadas, lo que genera dificultades al momento de querer realizar una mejora o cambio en los procesos del dispositivo. En las

FPGAs la lógica interna se puede adaptar según se requiera, reprogramando el microchip.

Se ha demostrado en base al resultado de las pruebas efectuadas, que tanto los objetivos generales y específicos del proyecto se cumplieron satisfactoriamente.

Al hablar de estabilidad y buen funcionamiento del servidor embebido se refiere a que el servicio que proporciona de enviar información HTML a los clientes es generado correctamente. También se define estabilidad a la capacidad de establecer conexiones simultáneas sin problemas. Las pruebas efectuadas demostraron que el servidor se comporta estable y presenta un buen funcionamiento cuando se generan peticiones de data HTML con una cantidad igual o menor a tres clientes. Esto a su vez define un número máximo de clientes determinado y garantiza en tal cifra la correcta operación del sistema.

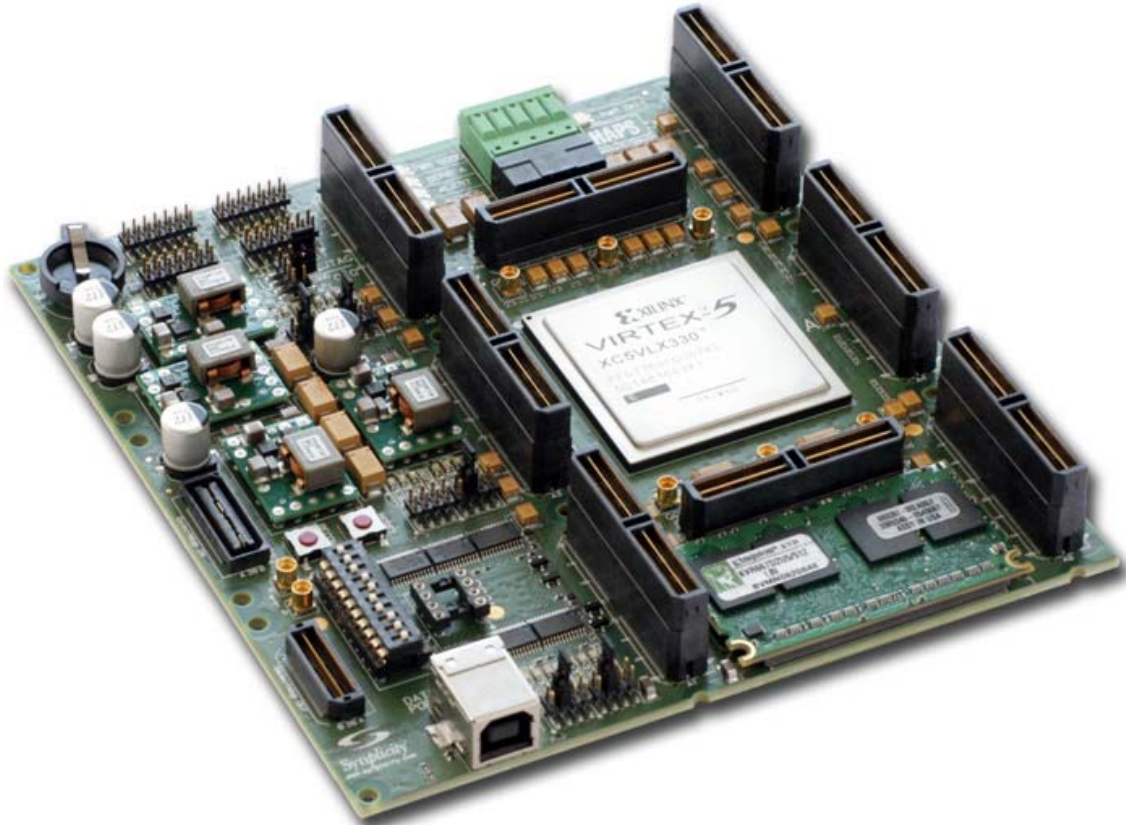
La capacidad de máxima de carga que es capaz de enviar el prototipo a un cliente, es igual o menor a 20 solicitudes de contenido, lo que se traduce en una capacidad máxima de carga igual a 80 KBytes ya que un solo archivo HTML posee 4 KBytes.

## 8. Bibliografía

- [1] Rabaey, Jan M.(1996)“Digital integrated circuits : a design perspective”. Circuits Design, Prencice Hall, 708pp.
- [2] Escarate Moraga, Cristian Andrés (2009) “Diseño y fabricación de hardware y Firmware para sistemas de control múltiple”, CD.
- [3] Jhon Wiley & Sons (2008) “FPGA-based implementation of complex signal processing systems”. Edición XXI, 364 pp.
- [4] Brown, Stephen (2006) “Fundamentos de lógica digital con diseño VHDL”. McGraw-Hill, Edición XX, 939 pp.
- [5] Pardo Carpio, Fernando (1999) “VHDL lenguaje para síntesis y modelamiento de circuitos”. Ra-Ma, CD, Edición XI, 283 pp.
- [6] Rico López, Rafael (1998) “Simulación de arquitectura de computadores”. Edición VIII, 213 pp.
- [7] Ruz Ortiz, José Jaime (1997) “VHDL: De la tecnología a la arquitectura de computadores”. Síntesis, Edición X, 318 pp.
- [8] Nicolás Peña Ralph (2010) “DESARROLLO E IMPLEMENTACION DE SISTEMA DE COMUNICACIÓN RAW SOCKET TCP/IP Y SERVIDOR HTML EMPOTRADOS, BASADOS EN PROGRAMACION DE TECNOLOGIA FPGA MODELO VIRTEX-5 XILINX”, Licenciatura de Titulación. Universidad Mayor. Facultad de Ingeniería. Profesor Guía, Gonzalo Téllez.

## 9. Anexos

### 9.1. Tarjeta Haps-51



*Figura 9-1: Tarjeta Haps-51*  
Fuente: Synopsys.com (2010)

## 9.2. Componentes Haps-51 – Cara superior

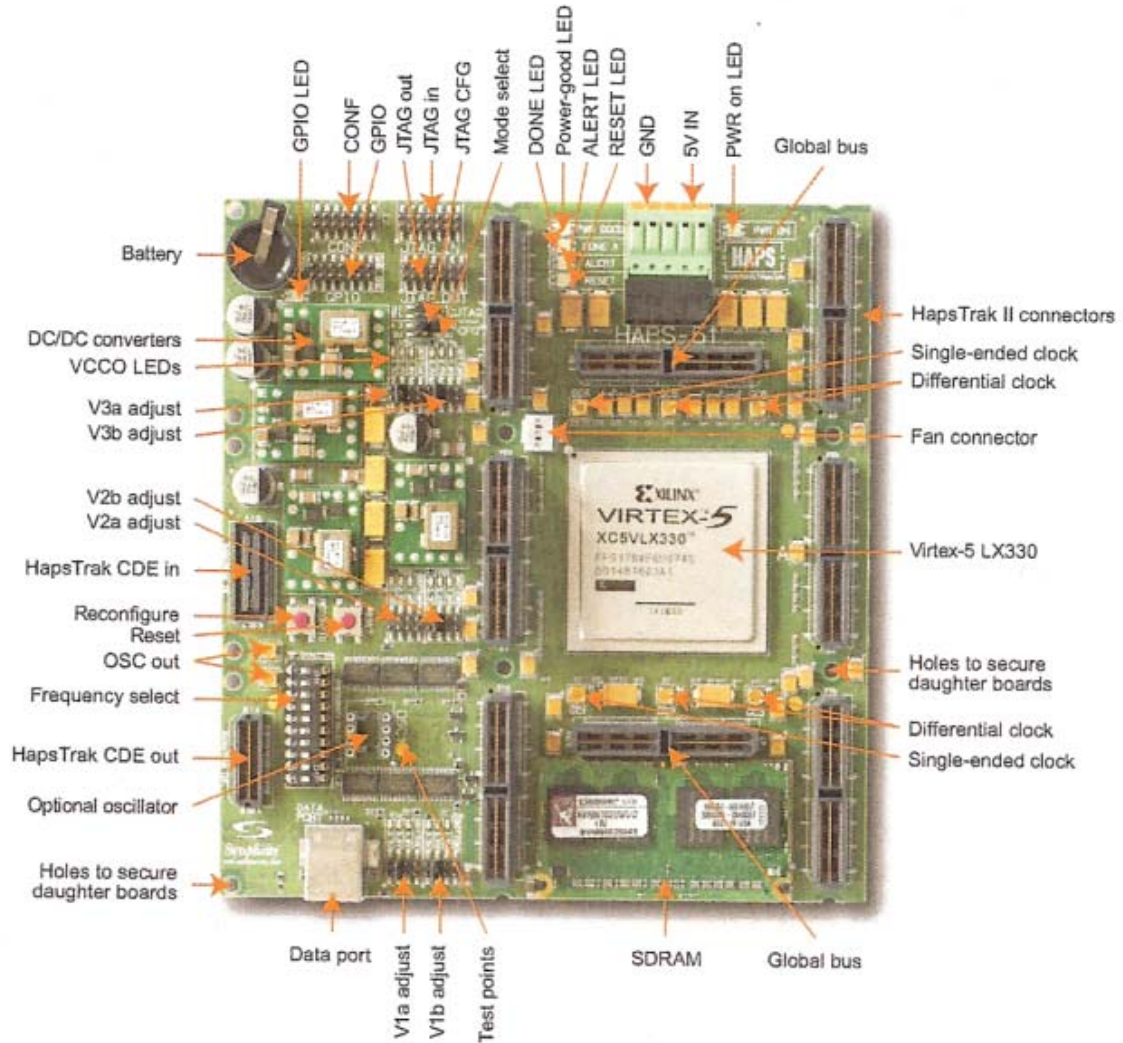


Figura 9-2: Detalles de Haps-51 (cara superior)  
Fuente: Synopsys.com (2010)

### 9.3. Componentes Haps-51 – Cara inferior

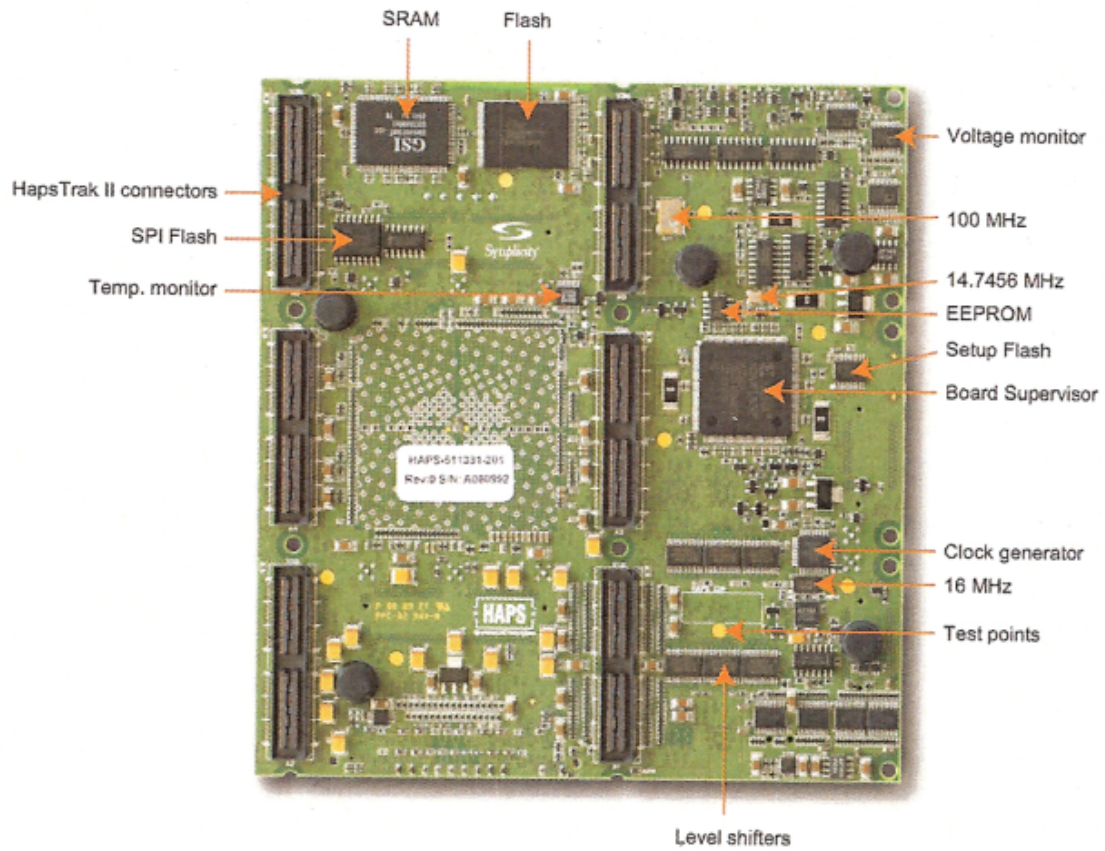


Figura 9-3: Detalles Haps-51 (cara inferior)  
Fuente: Synopsys.com (2010)

#### **9.4. Especificaciones técnicas Haps-51**

- 1 Xilinx Virtex-5 LX330 dispositivo de paquete FF1760
  - 2 Millones de compuertas ASIC.
- Tasas de transferencia de:
  - 1 Gbps LVDS.
  - 600 Mbps para salidas simples.
- 875 señales disponibles de entrada y salidas
  - 714 I/O disponibles en 6 conectores Hapstrack II.
    - I/O Simples o diferenciales.
  - 119 I/O disponibles en 2 conectores Hapstrack II utilizados para expansión con otras tarjetas madres.
  - 10 GPIO.
- Memorias en la tarjeta:
  - DDR2 SDRAM: 512M x 64 bit .
  - SRAM sincrona: 32 M x 36 bit.
  - FLASH PROM: 32 M x 16 bit.
- 2 pares de relojes diferenciales y 2 de salidas simples
- 56 relojes locales diferenciales o simples
- 1 generador de reloj programable
- 6 regiones de VCCO
  - Cada región puede ser configurada individualmente en
  - 3.3, 2.5 o 1.8 V
- Cable de configuración JTAG
- Monitor de temperatura y de voltaje de la tarjeta
- Driver controlador de ventilador de temperatura
- Test de revisión de chequeo de la tarjeta
- Fuente de alimentación de 5 V simple.

## 9.5. Puerto Hapstrack II número A2

bank		A2				bank	
		B	A				
		3.3V	H1	H2	3.3V		
CN	BA37	1	N	1	AD32		
CP	BB37	2	P	2	AC33		
	AY39	3	N	3	AE32		
	AW38	4	P	4	AD33		
	AW37	5	N	5	AE34		
	AY36	6	P	6	AE33		
	BA39	7	N	7	AV38	VR	
	BB36	8	P	8	AV39		
VR	BB39	9	N	9	AU37		
	BA40	10	P	10	AU38		
	BB41	11	N	11	AR36		
	BA42	12	P	12	AT37		
	BA41	13	N	13	AT36		
	AY40	14	P	14	AR37		
	AW40	15	N	15	AF34	CN	21
	AW41	16	P	16	AE35	CP	
CN	AW35	17	N	17	AK34		
CP	AY35	18	P	18	AL34		
	BA34	19	N	19	AL35		
	BB33	20	P	20	AL36		
	AW32	21	N	21	AK35		
	AW33	22	P	22	AJ35		
VR	AY34	23	N	23	AJ36		
	AY33	24	P	24	AH36		
	BA32	25	N	25	AN35	VR	
	AY32	26	P	26	AM36		
	BB31	27	N	27	AM34		
	BB32	28	P	28	AN34		
	BA31	29	N	29	AN36		
	BA30	30	P	30	AP36		
RFU	SCK	H3		H4	A0	RFU	
RFU	SDA	H5		H6	A1	RFU	
	AW31	31	N	31	AG36	CN	21
	AY30	32	P	32	AH35	CP	
CN	AW25	33	N	33	AG33		
CP	AW26	34	P	34	AF32		
	AV23	35	N	35	AG32		
	AV24	36	P	36	AH33		
	AV26	37	N	37	AJ31		
	AW27	38	P	38	AH31		
VR	AU27	39	N	39	AT35		
	AU26	40	P	40	AU36		
	AW22	41	N	41	AV36	VR	
	AW23	42	P	42	AV35		
	AU24	43	N	43	AT34		
	AV25	44	P	44	AU34		
	AW30	45	N	45	AR34		
	AV30	46	P	46	AR35		
	AU26	47	N	47	AU33	CN	25
	AV26	48	P	48	AU32	CP	
CN	AW20	49	N	49	AK32		
CP	AW21	50	P	50	AJ32		
	AV19	51	N	51	AJ33		
	AW18	52	P	52	AK33		
	AV18	53	N	53	AP33	VR	
	AW17	54	P	54	AR33		
VR	AU16	55	N	55	AM33		
	AU17	56	P	56	AN33		
	AW13	57	N	57	AP32		
	AV13	58	P	58	AR32		
	AV16	59	N	59	AT31	CN	
VCCO	V1b	60	A	60	AT32	CP	
VCCO	V1b	H7	A	H8	V1b	VCCO	

Figura 9-4: Pines Hapstrack II (Bus A2)  
Fuente: DS Haps-51(2005)

## 9.6. Virtex-5 LX330 y otros modelos

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices <sup>(2)</sup>	Block RAM Blocks			CMTs <sup>(4)</sup>	PowerPC Processor Blocks	Endpoint Blocks for PCI Express	Ethernet MACs <sup>(5)</sup>	Max RocketIO Transceivers <sup>(6)</sup>		Total I/O Banks <sup>(8)</sup>	Max User I/O <sup>(7)</sup>
	Array (Row x Col)	Virtex-5 Slices <sup>(1)</sup>	Max Distributed RAM (Kb)		18 Kb <sup>(3)</sup>	36 Kb	Max (Kb)					GTP	GTX		
XC5VLX30	80 x 30	4,800	320	32	64	32	1,152	2	N/A	N/A	N/A	N/A	N/A	13	400
XC5VLX50	120 x 30	7,200	480	48	96	48	1,728	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX85	120 x 54	12,960	840	48	192	96	3,456	6	N/A	N/A	N/A	N/A	N/A	17	560
XC5VLX110	160 x 54	17,280	1,120	64	256	128	4,608	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX155	160 x 76	24,320	1,640	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX220	160 x 108	34,560	2,280	128	384	192	6,912	6	N/A	N/A	N/A	N/A	N/A	23	800
XC5VLX330	240 x 108	51,840	3,420	192	576	288	10,368	6	N/A	N/A	N/A	N/A	N/A	33	1,200
XC5VLX20T	60 x 26	3,120	210	24	52	26	936	1	N/A	1	2	4	N/A	7	172
XC5VLX30T	80 x 30	4,800	320	32	72	36	1,296	2	N/A	1	4	8	N/A	12	360
XC5VLX50T	120 x 30	7,200	480	48	120	60	2,160	6	N/A	1	4	12	N/A	15	480
XC5VLX85T	120 x 54	12,960	840	48	216	108	3,888	6	N/A	1	4	12	N/A	15	480
XC5VLX110T	160 x 54	17,280	1,120	64	296	148	5,328	6	N/A	1	4	16	N/A	20	680
XC5VLX155T	160 x 76	24,320	1,640	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX220T	160 x 108	34,560	2,280	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX330T	240 x 108	51,840	3,420	192	648	324	11,664	6	N/A	1	4	24	N/A	27	960
XC5VSX35T	80 x 34	5,440	520	192	168	84	3,024	2	N/A	1	4	8	N/A	12	360
XC5VSX50T	120 x 34	8,160	780	288	264	132	4,752	6	N/A	1	4	12	N/A	15	480
XC5VSX95T	160 x 46	14,720	1,520	640	488	244	8,784	6	N/A	1	4	16	N/A	19	640
XC5VSX240T	240 x 78	37,440	4,200	1,056	1,032	516	18,576	6	N/A	1	4	24	N/A	27	960
XC5VTX150T	200 x 58	23,200	1,500	80	456	228	8,208	6	N/A	1	4	N/A	40	20	680
XC5VTX240T	240 x 78	37,440	2,400	96	648	324	11,664	6	N/A	1	4	N/A	48	20	680
XC5VFX30T	80 x 38	5,120	380	64	136	68	2,448	2	1	1	4	N/A	8	12	360
XC5VFX70T	160 x 38	11,200	820	128	296	148	5,328	6	1	3	4	N/A	16	19	640
XC5VFX100T	160 x 56	16,000	1,240	256	456	228	8,208	6	2	3	4	N/A	16	20	680
XC5VFX130T	200 x 56	20,480	1,580	320	596	298	10,728	6	2	3	6	N/A	20	24	840
XC5VFX200T	240 x 68	30,720	2,280	384	912	456	16,416	6	2	4	8	N/A	24	27	960

Figura 9-5: Familia Virtex-5  
Fuente: Xilinx DS100 v5.0 (2009)

## **9.7. Tarjeta GEPHY y especificaciones técnicas**

- Factor 1x1 quiere decir que se necesita tan solo 1 puerto Hapstrak II para realizar la conexión a una tarjeta madre o a una tarjeta hermana.
- Dos 10/100/1000 links Ethernet de cobre
- Cumple el estándar IEEE 10Base-T, 100Baste-TX y 1000Baste-T.
- Cumple el estándar de 802.3u Auto negociación y de detección paralela.
- Dos relojes maestros de velocidad seleccionadle a 25 MHz y de 125 MHz.
- LEDs indicadores de actividad, link10, link100, link1000 y full/half duplex.

## 9.8. Código de Archivo MHS

PARAMETER VERSION = 2.1.0

```
PORT sys_clk_pin = dcm_clk_s, DIR = I, SIGIS = CLK, CLK_FREQ =
100000000
PORT sys_rst_pin = sys_rst_s, DIR = I, RST_POLARITY = 0, SIGIS = RST
PORT FLASH_32M_16_SRAM_2M_32_Mem_RPN_pin =
FLASH_32M_16_SRAM_2M_32_Mem_RPN, DIR = O
PORT FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin =
FLASH_32M_16_SRAM_2M_32_Mem_WEN, DIR = O
PORT FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin =
FLASH_32M_16_SRAM_2M_32_Mem_CEN, DIR = O, VEC = [0:1]
PORT FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin =
FLASH_32M_16_SRAM_2M_32_Mem_OEN, DIR = O, VEC = [0:1]
PORT SDRAM_128M_64_DDR2_DM_pin = SDRAM_128M_64_DDR2_DM,
DIR = O, VEC = [7:0]
PORT SDRAM_128M_64_DDR2_Clk_n_pin =
SDRAM_128M_64_DDR2_Clk_n, DIR = O, VEC = [1:0], SIGIS = CLK
PORT SDRAM_128M_64_DDR2_RAS_n_pin =
SDRAM_128M_64_DDR2_RAS_n, DIR = O
PORT SDRAM_128M_64_DDR2_CAS_n_pin =
SDRAM_128M_64_DDR2_CAS_n, DIR = O
PORT SDRAM_128M_64_DDR2_WE_n_pin =
SDRAM_128M_64_DDR2_WE_n, DIR = O
PORT SDRAM_128M_64_DDR2_CE_pin = SDRAM_128M_64_DDR2_CE,
DIR = O, VEC = [1:0]
PORT SDRAM_128M_64_DDR2_CS_n_pin =
SDRAM_128M_64_DDR2_CS_n, DIR = O, VEC = [1:0]
PORT SDRAM_128M_64_DDR2_BankAddr_pin =
SDRAM_128M_64_DDR2_BankAddr, DIR = O, VEC = [2:0]
```

PORT FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_ADV\_LDN\_pin =  
 FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_ADV\_LDN, DIR = O  
 PORT FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_BEN\_pin =  
 FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_BEN, DIR = O, VEC = [0:3]  
 PORT FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_A\_pin =  
 FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_A, DIR = O, VEC = [0:31]  
 PORT SDRAM\_128M\_64\_DDR2\_Clk\_pin = SDRAM\_128M\_64\_DDR2\_Clk,  
 DIR = O, VEC = [1:0], SIGIS = CLK  
 PORT SDRAM\_128M\_64\_DDR2\_ODT\_pin = SDRAM\_128M\_64\_DDR2\_ODT,  
 DIR = O, VEC = [1:0]  
 PORT SDRAM\_128M\_64\_DDR2\_Addr\_pin = SDRAM\_128M\_64\_DDR2\_Addr,  
 DIR = O, VEC = [12:0]  
 PORT SDRAM\_128M\_64\_DDR2\_DQS = SDRAM\_128M\_64\_DDR2\_DQS, DIR  
 = IO, VEC = [7:0]  
 PORT SDRAM\_128M\_64\_DDR2\_DQS\_n = SDRAM\_128M\_64\_DDR2\_DQS\_n,  
 DIR = IO, VEC = [7:0]  
 PORT SDRAM\_128M\_64\_DDR2\_DQ = SDRAM\_128M\_64\_DDR2\_DQ, DIR =  
 IO, VEC = [63:0]  
 PORT FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_DQ =  
 FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_DQ, DIR = IO, VEC = [0:31]  
 PORT xps\_ethernetlite\_0\_PHY\_col\_pin = xps\_ethernetlite\_0\_PHY\_col, DIR = I  
 PORT xps\_ethernetlite\_0\_PHY\_crs\_pin = xps\_ethernetlite\_0\_PHY\_crs, DIR = I  
 PORT xps\_ethernetlite\_0\_PHY\_tx\_en\_pin = xps\_ethernetlite\_0\_PHY\_tx\_en,  
 DIR = O  
 PORT xps\_ethernetlite\_0\_PHY\_tx\_data\_pin =  
 xps\_ethernetlite\_0\_PHY\_tx\_data, DIR = O, VEC = [3:0]  
 PORT xps\_ethernetlite\_0\_PHY\_rx\_clk\_pin = xps\_ethernetlite\_0\_PHY\_rx\_clk,  
 DIR = I  
 PORT xps\_ethernetlite\_0\_PHY\_tx\_clk\_pin = xps\_ethernetlite\_0\_PHY\_tx\_clk,  
 DIR = I  
 PORT xps\_ethernetlite\_0\_PHY\_rx\_data\_pin =  
 xps\_ethernetlite\_0\_PHY\_rx\_data, DIR = I, VEC = [3:0]  
 PORT xps\_ethernetlite\_0\_PHY\_rx\_er\_pin = xps\_ethernetlite\_0\_PHY\_rx\_er,  
 DIR = I

```
PORT xps_ethernetlite_0_PHY_dv_pin = xps_ethernetlite_0_PHY_dv, DIR = I
PORT xps_ethernetlite_0_PHY_rst_n_pin = xps_ethernetlite_0_PHY_rst_n,
DIR = O
PORT sram_clk_pin = sys_clk_s, DIR = O
```

```
BEGIN microblaze
```

```
PARAMETER INSTANCE = microblaze_0
PARAMETER HW_VER = 7.10.d
PARAMETER C_CACHE_BYTE_SIZE = 2048
PARAMETER C_FAMILY = virtex5
PARAMETER C_INSTANCE = microblaze_0
PARAMETER C_USE_FPU = 1
PARAMETER C_USE_BARREL = 1
PARAMETER C_DEBUG_ENABLED = 1
PARAMETER C_NUMBER_OF_PC_BRK = 2
PARAMETER C_NUMBER_OF_RD_ADDR_BRK = 1
PARAMETER C_NUMBER_OF_WR_ADDR_BRK = 1
PARAMETER C_USE_ICACHE = 1
PARAMETER C_USE_DCACHE = 1
PARAMETER C_DCACHE_BYTE_SIZE = 2048
PARAMETER C_ICACHE_BASEADDR = 0xc0000000
PARAMETER C_ICACHE_HIGHADDR = 0xffffffff
PARAMETER C_DCACHE_BASEADDR = 0xc0000000
PARAMETER C_DCACHE_HIGHADDR = 0xffffffff
BUS_INTERFACE DPLB = mb_plb
BUS_INTERFACE IPLB = mb_plb
BUS_INTERFACE IXCL = microblaze_0_IXCL
BUS_INTERFACE DXCL = microblaze_0_DXCL
BUS_INTERFACE DEBUG = microblaze_0_MBDEBUG
BUS_INTERFACE DLMB = dlmb
BUS_INTERFACE ILMB = ilmb
PORT MB_RESET = mb_reset
PORT Interrupt = Interrupt
```

END

BEGIN plb\_v46

PARAMETER INSTANCE = mb\_plb

PARAMETER HW\_VER = 1.03.a

PORT PLB\_Clk = sys\_clk\_s

PORT SYS\_Rst = sys\_bus\_reset

END

BEGIN lmb\_v10

PARAMETER INSTANCE = ilmb

PARAMETER HW\_VER = 1.00.a

PORT LMB\_Clk = sys\_clk\_s

PORT SYS\_Rst = sys\_bus\_reset

END

BEGIN lmb\_v10

PARAMETER INSTANCE = dlmb

PARAMETER HW\_VER = 1.00.a

PORT LMB\_Clk = sys\_clk\_s

PORT SYS\_Rst = sys\_bus\_reset

END

BEGIN lmb\_bram\_if\_cntlr

PARAMETER INSTANCE = dlmb\_cntlr

PARAMETER HW\_VER = 2.10.a

PARAMETER C\_BASEADDR = 0x00000000

PARAMETER C\_HIGHADDR = 0x00001fff

BUS\_INTERFACE SLMB = dlmb

BUS\_INTERFACE BRAM\_PORT = dlmb\_port

END

BEGIN lmb\_bram\_if\_cntlr

PARAMETER INSTANCE = ilmb\_cntlr

```
PARAMETER HW_VER = 2.10.a
PARAMETER C_BASEADDR = 0x00000000
PARAMETER C_HIGHADDR = 0x00001fff
BUS_INTERFACE SLMB = ilmb
BUS_INTERFACE BRAM_PORT = ilmb_port
END
```

```
BEGIN bram_block
PARAMETER INSTANCE = lmb_bram
PARAMETER HW_VER = 1.00.a
BUS_INTERFACE PORTA = ilmb_port
BUS_INTERFACE PORTB = dlmb_port
END
```

```
BEGIN clock_generator
PARAMETER INSTANCE = clock_generator_0
PARAMETER HW_VER = 2.01.a
PARAMETER C_EXT_RESET_HIGH = 1
PARAMETER C_CLKIN_FREQ = 100000000
PARAMETER C_CLKOUT0_FREQ = 100000000
PARAMETER C_CLKOUT0_BUF = TRUE
PARAMETER C_CLKOUT0_PHASE = 0
PARAMETER C_CLKOUT0_GROUP = NONE
PARAMETER C_CLKIN_BUF = FALSE
PARAMETER C_CLKOUT1_FREQ = 200000000
PARAMETER C_CLKOUT1_PHASE = 90
PARAMETER C_CLKOUT1_GROUP = PLL0
PARAMETER C_CLKOUT1_BUF = TRUE
PARAMETER C_CLKOUT2_FREQ = 200000000
PARAMETER C_CLKOUT2_PHASE = 0
PARAMETER C_CLKOUT2_GROUP = PLL0
PARAMETER C_CLKOUT2_BUF = TRUE
PARAMETER C_CLK_GEN = 0
PARAMETER C_NUM_DCM = 0
```

PARAMETER C\_CLKFBOUT\_MODULE = NONE  
PARAMETER C\_CLKFBOUT\_PORT = NONE  
PARAMETER C\_CLKOUT0\_MODULE = PLL0  
PARAMETER C\_CLKOUT0\_PORT = CLKOUT2B  
PARAMETER C\_CLKOUT10\_MODULE = NONE  
PARAMETER C\_CLKOUT10\_PORT = NONE  
PARAMETER C\_CLKOUT11\_MODULE = NONE  
PARAMETER C\_CLKOUT11\_PORT = NONE  
PARAMETER C\_CLKOUT12\_MODULE = NONE  
PARAMETER C\_CLKOUT12\_PORT = NONE  
PARAMETER C\_CLKOUT13\_MODULE = NONE  
PARAMETER C\_CLKOUT13\_PORT = NONE  
PARAMETER C\_CLKOUT14\_MODULE = NONE  
PARAMETER C\_CLKOUT14\_PORT = NONE  
PARAMETER C\_CLKOUT15\_MODULE = NONE  
PARAMETER C\_CLKOUT15\_PORT = NONE  
PARAMETER C\_CLKOUT1\_MODULE = PLL0  
PARAMETER C\_CLKOUT1\_PORT = CLKOUT0B  
PARAMETER C\_CLKOUT2\_MODULE = PLL0  
PARAMETER C\_CLKOUT2\_PORT = CLKOUT1B  
PARAMETER C\_CLKOUT3\_MODULE = NONE  
PARAMETER C\_CLKOUT3\_PORT = NONE  
PARAMETER C\_CLKOUT4\_MODULE = NONE  
PARAMETER C\_CLKOUT4\_PORT = NONE  
PARAMETER C\_CLKOUT5\_MODULE = NONE  
PARAMETER C\_CLKOUT5\_PORT = NONE  
PARAMETER C\_CLKOUT6\_MODULE = NONE  
PARAMETER C\_CLKOUT6\_PORT = NONE  
PARAMETER C\_CLKOUT7\_MODULE = NONE  
PARAMETER C\_CLKOUT7\_PORT = NONE  
PARAMETER C\_CLKOUT8\_MODULE = NONE  
PARAMETER C\_CLKOUT8\_PORT = NONE  
PARAMETER C\_CLKOUT9\_MODULE = NONE  
PARAMETER C\_CLKOUT9\_PORT = NONE

PARAMETER C\_PLL0\_BANDWIDTH = OPTIMIZED  
PARAMETER C\_PLL0\_CLKFBIN\_MODULE = PLL0  
PARAMETER C\_PLL0\_CLKFBIN\_PORT = CLKFBOUT  
PARAMETER C\_PLL0\_CLKFBOUT\_BUF = TRUE  
PARAMETER C\_PLL0\_CLKFBOUT\_DESKEW\_ADJUST = PPC  
PARAMETER C\_PLL0\_CLKFBOUT\_MULT = 10  
PARAMETER C\_PLL0\_CLKFBOUT\_PHASE = 0.0  
PARAMETER C\_PLL0\_CLKIN1\_BUF = FALSE  
PARAMETER C\_PLL0\_CLKIN1\_MODULE = CLKGEN  
PARAMETER C\_PLL0\_CLKIN1\_PERIOD = 10.000  
PARAMETER C\_PLL0\_CLKIN1\_PORT = CLKIN  
PARAMETER C\_PLL0\_CLKOUT0\_BUF = TRUE  
PARAMETER C\_PLL0\_CLKOUT0\_DESKEW\_ADJUST = NONE  
PARAMETER C\_PLL0\_CLKOUT0\_DIVIDE = 5  
PARAMETER C\_PLL0\_CLKOUT0\_DUTY\_CYCLE = 0.5  
PARAMETER C\_PLL0\_CLKOUT0\_PHASE = 90.0  
PARAMETER C\_PLL0\_CLKOUT1\_BUF = TRUE  
PARAMETER C\_PLL0\_CLKOUT1\_DESKEW\_ADJUST = NONE  
PARAMETER C\_PLL0\_CLKOUT1\_DIVIDE = 5  
PARAMETER C\_PLL0\_CLKOUT1\_DUTY\_CYCLE = 0.5  
PARAMETER C\_PLL0\_CLKOUT1\_PHASE = 0.0  
PARAMETER C\_PLL0\_CLKOUT2\_BUF = TRUE  
PARAMETER C\_PLL0\_CLKOUT2\_DESKEW\_ADJUST = NONE  
PARAMETER C\_PLL0\_CLKOUT2\_DIVIDE = 10  
PARAMETER C\_PLL0\_CLKOUT2\_DUTY\_CYCLE = 0.5  
PARAMETER C\_PLL0\_CLKOUT2\_PHASE = 0.0  
PARAMETER C\_PLL0\_CLKOUT3\_BUF = FALSE  
PARAMETER C\_PLL0\_CLKOUT3\_DESKEW\_ADJUST = PPC  
PARAMETER C\_PLL0\_CLKOUT3\_DIVIDE = 1  
PARAMETER C\_PLL0\_CLKOUT3\_DUTY\_CYCLE = 0.5  
PARAMETER C\_PLL0\_CLKOUT3\_PHASE = 0.0  
PARAMETER C\_PLL0\_CLKOUT4\_BUF = FALSE  
PARAMETER C\_PLL0\_CLKOUT4\_DESKEW\_ADJUST = PPC  
PARAMETER C\_PLL0\_CLKOUT4\_DIVIDE = 1

```

PARAMETER C_PLL0_CLKOUT4_DUTY_CYCLE = 0.5
PARAMETER C_PLL0_CLKOUT4_PHASE = 0.0
PARAMETER C_PLL0_CLKOUT5_BUF = FALSE
PARAMETER C_PLL0_CLKOUT5_DESKEW_ADJUST = PPC
PARAMETER C_PLL0_CLKOUT5_DIVIDE = 1
PARAMETER C_PLL0_CLKOUT5_DUTY_CYCLE = 0.5
PARAMETER C_PLL0_CLKOUT5_PHASE = 0.0
PARAMETER C_PLL0_COMPENSATION = SYSTEM_SYNCHRONOUS
PARAMETER C_PLL0_DIVCLK_DIVIDE = 1
PARAMETER C_PLL0_EXT_RESET_HIGH = 1
PARAMETER C_PLL0_FAMILY = virtex5
PARAMETER C_PLL0_REF_JITTER = 0.100
PARAMETER C_PLL0_RESET_ON_LOSS_OF_LOCK = FALSE
PARAMETER C_PLL0_RST_DEASSERT_CLK = CLKIN1
PARAMETER C_PLL0_RST_MODULE = CLKGEN
PORT CLKOUT0 = sys_clk_s
PORT CLKIN = dcm_clk_s
PORT LOCKED = Dcm_all_locked
PORT RST = net_gnd
PORT CLKOUT1 = clock_200_90
PORT CLKOUT2 = clock_200
END

```

```

BEGIN proc_sys_reset
PARAMETER INSTANCE = proc_sys_reset_0
PARAMETER HW_VER = 2.00.a
PARAMETER C_EXT_RESET_HIGH = 0
PORT Slowest_sync_clk = sys_clk_s
PORT Dcm_locked = Dcm_all_locked
PORT Ext_Reset_In = sys_rst_s
PORT MB_Reset = mb_reset
PORT Bus_Struct_Reset = sys_bus_reset
PORT MB_Debug_Sys_Rst = Debug_SYS_Rst
PORT Peripheral_Reset = sys_periph_reset

```

END

BEGIN xps\_mch\_emc

PARAMETER INSTANCE = FLASH\_32M\_16

PARAMETER HW\_VER = 2.00.a

PARAMETER C\_NUM\_CHANNELS = 0

PARAMETER C\_MEM0\_WIDTH = 16

PARAMETER C\_INCLUDE\_DATAWIDTH\_MATCHING\_0 = 1

PARAMETER C\_INCLUDE\_DATAWIDTH\_MATCHING\_1 = 1

PARAMETER C\_SYNCH\_MEM\_1 = 1

PARAMETER C\_SPLB\_SMALLEST\_MASTER = 24

PARAMETER C\_MEM0\_BASEADDR = 0xa4000000

PARAMETER C\_MEM0\_HIGHADDR = 0xa7ffffff

BUS\_INTERFACE SPLB = mb\_plb

PORT Mem\_RPN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_RPN

PORT Mem\_WEN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_WEN

PORT Mem\_CEN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_CEN

PORT Mem\_OEN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_OEN

PORT Mem\_DQ = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_DQ

PORT Mem\_A = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_A

PORT Mem\_ADV\_LDN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_ADV\_LDN

PORT RdClk = sys\_clk\_s

PORT Mem\_BEN = FLASH\_32M\_16\_SRAM\_2M\_32\_Mem\_BEN

END

BEGIN mpmc

PARAMETER INSTANCE = SDRAM\_128M\_64

PARAMETER HW\_VER = 4.03.a

PARAMETER C\_MEM\_PARTNO = KVR667D2U5/1G

PARAMETER C\_MPMC\_CLK0\_PERIOD\_PS = 10000

PARAMETER C\_MEM\_DQS\_IO\_COL = 0x00000000000000000000

PARAMETER C\_MEM\_DQ\_IO\_MS = 0x00000000000000000000

PARAMETER C\_PIM1\_BASETTYPE = 1

PARAMETER C\_PIM2\_BASETTYPE = 1

```

PARAMETER C_NUM_PORTS = 3
PARAMETER C_MEM_CE_WIDTH = 2
PARAMETER C_MEM_ODT_WIDTH = 2
PARAMETER C_MEM_CLK_WIDTH = 2
PARAMETER C_MEM_CS_N_WIDTH = 2
PARAMETER C_MEM_NUM_RANKS = 2
PARAMETER C_MEM_REG_DIMM = 0
PARAMETER C_MPMC_BASEADDR = 0xc0000000
PARAMETER C_MPMC_HIGHADDR = 0xffffffff
BUS_INTERFACE SPLB0 = mb_plb
BUS_INTERFACE XCL2 = microblaze_0_IXCL
BUS_INTERFACE XCL1 = microblaze_0_DXCL
PORT DDR2_DQ = SDRAM_128M_64_DDR2_DQ
PORT DDR2_DQS = SDRAM_128M_64_DDR2_DQS
PORT DDR2_DQS_n = SDRAM_128M_64_DDR2_DQS_n
PORT DDR2_DM = SDRAM_128M_64_DDR2_DM
PORT DDR2_Clk = SDRAM_128M_64_DDR2_Clk
PORT DDR2_Clk_n = SDRAM_128M_64_DDR2_Clk_n
PORT DDR2_RAS_n = SDRAM_128M_64_DDR2_RAS_n
PORT DDR2_CAS_n = SDRAM_128M_64_DDR2_CAS_n
PORT DDR2_WE_n = SDRAM_128M_64_DDR2_WE_n
PORT DDR2_ODT = SDRAM_128M_64_DDR2_ODT
PORT DDR2_CE = SDRAM_128M_64_DDR2_CE
PORT DDR2_CS_n = SDRAM_128M_64_DDR2_CS_n
PORT DDR2_BankAddr = SDRAM_128M_64_DDR2_BankAddr
PORT MPMC_Rst = sys_periph_reset
PORT DDR2_Addr = SDRAM_128M_64_DDR2_Addr
PORT MPMC_Clk0 = clock_200
PORT MPMC_Clk_200MHz = clock_200
PORT MPMC_Clk90 = clock_200_90
PORT MPMC_Clk0_DIV2 = sys_clk_s
END

```

```
BEGIN mdm
```

```
PARAMETER INSTANCE = debug_module
PARAMETER HW_VER = 1.00.d
PARAMETER C_MB_DBG_PORTS = 1
PARAMETER C_USE_UART = 1
PARAMETER C_UART_WIDTH = 8
PARAMETER C_BASEADDR = 0x84400000
PARAMETER C_HIGHADDR = 0x8440ffff
BUS_INTERFACE SPLB = mb_plb
BUS_INTERFACE MBDEBUG_0 = microblaze_0_MBDEBUG
PORT Debug_SYS_Rst = Debug_SYS_Rst
PORT Interrupt = debug_module_Interrupt
END
```

```
BEGIN xps_intc
PARAMETER INSTANCE = xps_intc_0
PARAMETER HW_VER = 1.00.a
PARAMETER C_BASEADDR = 0x81800000
PARAMETER C_HIGHADDR = 0x8180ffff
BUS_INTERFACE SPLB = mb_plb
PORT Irq = Interrupt
PORT Intr =
debug_module_Interrupt&xps_timer_0_Interrupt&xps_ethernetlite_0_IP2INTC_I
rpt
END
```

```
BEGIN xps_ethernetlite
PARAMETER INSTANCE = xps_ethernetlite_0
PARAMETER HW_VER = 2.00.b
PARAMETER C_BASEADDR = 0x81000000
PARAMETER C_HIGHADDR = 0x8100ffff
BUS_INTERFACE SPLB = mb_plb
PORT PHY_col = xps_ethernetlite_0_PHY_col
PORT PHY_crs = xps_ethernetlite_0_PHY_crs
PORT PHY_tx_en = xps_ethernetlite_0_PHY_tx_en
```

```
PORT PHY_tx_data = xps_ethernetlite_0_PHY_tx_data
PORT PHY_rx_clk = edk_bufr_ether_rx_out_clk
PORT PHY_tx_clk = edk_bufr_ether_tx_out_clk
PORT PHY_rx_data = xps_ethernetlite_0_PHY_rx_data
PORT PHY_rx_er = xps_ethernetlite_0_PHY_rx_er
PORT PHY_dv = xps_ethernetlite_0_PHY_dv
PORT IP2INTC_Irpt = xps_ethernetlite_0_IP2INTC_Irpt
PORT PHY_rst_n = xps_ethernetlite_0_PHY_rst_n
END
```

```
BEGIN xps_timer
PARAMETER INSTANCE = xps_timer_0
PARAMETER HW_VER = 1.00.a
PARAMETER C_BASEADDR = 0x83c00000
PARAMETER C_HIGHADDR = 0x83c0ffff
BUS_INTERFACE SPLB = mb_plb
PORT Interrupt = xps_timer_0_Interrupt
END
```

```
BEGIN edk_bufr
PARAMETER INSTANCE = edk_bufr_ether_rx
PARAMETER HW_VER = 1.00.a
PORT in_clk = xps_ethernetlite_0_PHY_rx_clk
PORT out_clk = edk_bufr_ether_rx_out_clk
END
```

```
BEGIN edk_bufr
PARAMETER INSTANCE = edk_bufr_ether_tx
PARAMETER HW_VER = 1.00.a
PORT in_clk = xps_ethernetlite_0_PHY_tx_clk
PORT out_clk = edk_bufr_ether_tx_out_clk
END
```

## 9.9. Archivo UCF

```
Net sys_clk_pin LOC=AL27 | IOSTANDARD = LVCMOS25; #ok
Net sys_rst_pin LOC=L14 | IOSTANDARD = LVCMOS25; #ok
Net sram_clk_pin LOC=K29 | IOSTANDARD = LVCMOS25; #ok
## Reloj del sistema
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 10000 ps;
Net sys_rst_pin TIG;

## Dispositivos IO

#### Modulo Ethernet_10_100

Net xps_ethernetlite_0_PHY_col_pin LOC="AE33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_crs_pin LOC="AV39" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_en_pin LOC="AU38" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_clk_pin LOC="AE35" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<0> LOC="AP35" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<1> LOC="AN34" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<2> LOC="AM36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_tx_data_pin<3> LOC="AH36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_er_pin LOC="AU36" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_clk_pin LOC="AT32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<0> LOC="AR32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<1> LOC="AN33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<2> LOC="AR33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rx_data_pin<3> LOC="AK33" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_dv_pin LOC="AU32" | IOSTANDARD = LVCMOS25 ;
Net xps_ethernetlite_0_PHY_rst_n_pin LOC="AV16" | IOSTANDARD = LVCMOS25 ;

#### Modulo DDR_SDRAM_128Mx64

Net SDRAM_128M_64_DDR2_Addr_pin<0> LOC=AJ12 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<1> LOC=AV9 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<2> LOC=AT11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<3> LOC=AT10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<4> LOC=AP11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<5> LOC=AJ10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<6> LOC=AV10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<7> LOC=AN11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<8> LOC=AL11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<9> LOC=AJ11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<10> LOC=AN10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<11> LOC=AM11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Addr_pin<12> LOC=AM12 | IOSTANDARD = SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_DQ<0> LOC= BA29 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<1> LOC= AY28 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<2> LOC= BB28 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<3> LOC= BA25 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<4> LOC= BA27 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<5> LOC= BB29 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<6> LOC= BA24 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<7> LOC= BB24 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<8> LOC= BB18 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<9> LOC= BB27 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<10> LOC= AY18 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<11> LOC= AY15 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<12> LOC= BB23 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<13> LOC= BA16 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<14> LOC= BA17 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DQ<15> LOC= BB17 | IOSTANDARD = SSTL18_II ; #ok
```



```

Net SDRAM_128M_64_DDR2_DQS_n<0> LOC=AY23 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<1> LOC=BB22 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<2> LOC=BA22 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<3> LOC=BA7 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<4> LOC=AL7 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<5> LOC=BA6 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<6> LOC=AK9 | IOSTANDARD = DIFF_SSTL18_II; #ok
Net SDRAM_128M_64_DDR2_DQS_n<7> LOC=AW8 | IOSTANDARD = DIFF_SSTL18_II; #ok

Net SDRAM_128M_64_DDR2_DM_pin<0> LOC=BB26 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<1> LOC=AY27 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<2> LOC=AY14 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<3> LOC=AY10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<4> LOC=AH9 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<5> LOC=BA5 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<6> LOC=AE8 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_DM_pin<7> LOC=AY2 | IOSTANDARD = SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_Clk_pin<0> LOC=AU13 | IOSTANDARD = DIFF_SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Clk_pin<1> LOC=AV6 | IOSTANDARD = DIFF_SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_Clk_n_pin<0> LOC=AU12 | IOSTANDARD = DIFF_SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_Clk_n_pin<1> LOC=AU7 | IOSTANDARD = DIFF_SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_BankAddr_pin<0> LOC=AE10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_BankAddr_pin<1> LOC=AF11 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_BankAddr_pin<2> LOC=AF10 | IOSTANDARD= SSTL18_II ;#ok

Net SDRAM_128M_64_DDR2_CS_n_pin<0> LOC=AU9 | IOSTANDARD= SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_CS_n_pin<1> LOC=AT7 | IOSTANDARD= SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_CE_pin<0> LOC=AH10 | IOSTANDARD= SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_CE_pin<1> LOC=AF12 | IOSTANDARD = SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_ODT_pin<0> LOC=AK10 | IOSTANDARD= SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_ODT_pin<1> LOC=AU6 | IOSTANDARD= SSTL18_II ; #ok

Net SDRAM_128M_64_DDR2_WE_n_pin LOC=AL10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_CAS_n_pin LOC= AR10 | IOSTANDARD = SSTL18_II ; #ok
Net SDRAM_128M_64_DDR2_RAS_n_pin LOC= AG11 | IOSTANDARD = SSTL18_II ; #ok

#### Modulo FLASH

Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<0> LOC=AP21 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<1> LOC=AM21 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<2> LOC=AJ22 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<3> LOC=AJ23 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<4> LOC=AR14 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<5> LOC=AR13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<6> LOC=AP15 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<7> LOC=AR15 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<8> LOC=AM26 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<9> LOC=AL26 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<10> LOC=H30 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<11> LOC=H29 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<12> LOC=P18 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<13> LOC=R18 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<14> LOC=M19 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<15> LOC=M18 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<16> LOC=L19 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<17> LOC=K19 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<18> LOC=N24 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<19> LOC=M24 | IOSTANDARD = LVCMOS25; #ok

```

```

Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<20> LOC=G19 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<21> LOC=H19 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<22> LOC=H20 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<23> LOC=G21 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<24> LOC=N23 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_A_pin<25> LOC=M23 | IOSTANDARD = LVCMOS25; #ok

Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<0> LOC=L15 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<1> LOC=M28 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<2> LOC=L27 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<3> LOC=J13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<4> LOC=K13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<5> LOC=N26 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<6> LOC=M27 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<7> LOC=K14 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<8> LOC=K15 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<9> LOC=AL30 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<10> LOC=AL29 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<11> LOC=AJ30 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<12> LOC=AK30 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<13> LOC=AJ17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<14> LOC=AJ16 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<15> LOC=AH29 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<16> LOC=AH30 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<17> LOC=AJ15 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<18> LOC=AH16 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<19> LOC=N13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<20> LOC=M13 | IOSTANDARD = LVCMOS25; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<21> LOC=R27 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<22> LOC=R28 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<23> LOC=AR17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<24> LOC=AP16 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<25> LOC=K18 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<26> LOC=K17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<27> LOC=G18 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<28> LOC=G17 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<29> LOC=AP22 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<30> LOC=AR22 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_DQ<31> LOC=AT22 | IOSTANDARD = LVCMOS25; #ok

Net FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin LOC=L29; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_WEN_pin IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<1> LOC=AN28; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<1> IOSTANDARD = LVCMOS25;
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<0> LOC=M14; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_CEN_pin<0> IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<1> LOC=AM27; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<1> IOSTANDARD = LVCMOS25;
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<0> LOC=AP30; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_OEN_pin<0> IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN_pin LOC=K28; #ok
Net FLASH_32M_16_SRAM_2M_32_Mem_ADV_LDN_pin IOSTANDARD = LVCMOS25;

Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<0> LOC=AM28 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<1> LOC=AM14 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<2> LOC=AM13 | IOSTANDARD = LVCMOS25;#ok
Net FLASH_32M_16_SRAM_2M_32_Mem_BEN_pin<3> LOC=AN29 | IOSTANDARD = LVCMOS25;#ok

Net FLASH_32M_16_SRAM_2M_32_Mem_RPN_pin LOC=AP13 | IOSTANDARD = LVCMOS25; #ok

```

Figura 9-6: Archivo .UCF  
Fuente: Propia (2010)

## 9.10. Servidor HTML Embebido



*Figura 9-7: Prototipo Servidor HTML Embebido  
Fuente: Propia (2010)*

## 9.11. Tiempos de Acceso módulo MCH EMC

Los tiempos de acceso de la ventana de configuración del módulo XPS son:

- TCEDV (Read cycle chip enable low to data valid duration)
- TAVDV (Read cycle address valid to data valid duration)
- THZCE (Read cycle chip enable high to data bus high impedance duration)
- THZOE (Read cycle output enable high to data bus high impedance duration)
- TWC (Write cycle time)
- TWP (Write enable minimum pulse width duration)
- TLZWE (Write cycle write enable high to data bus low impedance duration)

## 9.12. FPGAs en Sistemas Embebidos

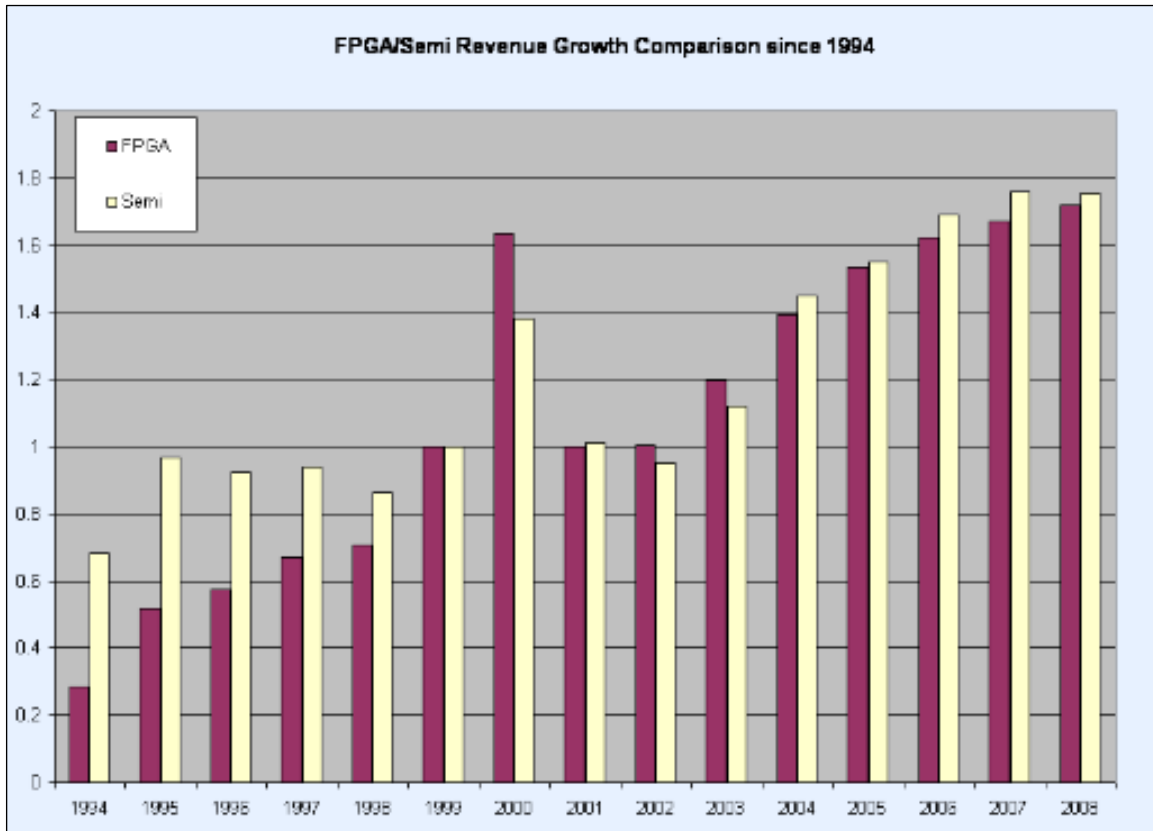


Figura 9-8: FPGA en Sistemas Embebidos

## 9.13. Lista de acrónimos

DDR	Double Data Rate
DS	DataSheet
Firmware	Programa Embebido
FPGA	Field Programmable Gate Array
HTML	HypertText Makeup Language
HTTP	HyperText Transfer Protocol
JTAG	Join Test Accion Group
KBytes	Kilo Bytes
LMB	Local Memory Bus
LWIP	Light Weight Internet Protocol
MCH EMC	Multi Channel External Memory Controller

MDM	Microblaze Debug Module
MHz	Mega Hertz
MPMC	Multi Protocol Memory Controller
PLB	Processor Local Bus
PLD	Programmable Logic Device
SSRAM	Synchronous Static Random Access Memory
TCP	Transmission Transfer Protocol
UDP	User Datagram Protocol
UG	User Guide
VHDL	Very Hardware Description Language
XPS	Xilinx Platform Studio