Universidad de Oriente

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones

TRABAJO DE DIPLOMA

Sistema de almacenamiento de datos para protocolo *IEEE-1394* sobre FPGA.

Autor: Carlos Enrique Bello Vila

Tutores: MSc. Ing. Berta Pallerols Mir

MSc. Ing. Alexander A. Suárez León

Santiago de Cuba

Junio, 2015

Universidad de Oriente

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones

TRABAJO DE DIPLOMA

Sistema de almacenamiento de datos para protocolo *IEEE-1394* sobre FPGA.

Autor: Carlos Enrique Bello Vila

carlos.bello@tle.fie.uo.edu.cu

Tutores: MSc. Ing. Berta Pallerols Mir

MSc. Ing. Alexander A. Suárez León

bertapm@fie.uo.edu.cu

aasl@fie.uo.edu.cu

Santiago de Cuba

Junio, 2015

COMPROMISO DEL AUTOR

Hago constar que el presente trabajo de diploma es de mi autoría exclusivamente, no constituyendo copia de ningún trabajo realizado anteriormente y las fuentes usadas para la realización del trabajo se encuentran referidas en la bibliografía. Doy mi consentimiento a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización del Tutor o Institución.

_____

Firma del Autor

PENSAMIENTO

"Mirada de cerca, la vida parece una tragedia; vista de lejos, parece una comedia. Nunca te olvides de sonreír, porque el día en que no sonrías será un día perdido. La vida es una obra de teatro que no permite ensayos. Por eso, canta, ríe, baila, llora y vive cada momento, antes de que baje el telón y la obra termine sin aplausos. Hay que tener fe en uno mismo... La vida es maravillosa si no se le tiene miedo."

Charles Chaplin

DEDICATORIA

A mis padres, porque sin su apoyo y constancia no habría podido llegar hasta aquí, mami por ser como eres de especial, porque para ti no hay nada imposible; papi, por tus consejos certeros y ser tú, estoy muy feliz de tenerlos conmigo, a ustedes les debo todo, nada será suficiente para retribuirle todo el amor, la dedicación, el esfuerzo y la ternura que han depositado en mí, y porque han sabido ser mi fuente de inspiración y fortaleza en todo momento.

Quiero dedicar este trabajo a una de las personas más importante en mi vida, a la que más admiro y respeto, a la cual le debo mi manera optimista de ver el mundo, la persona que me enseñó que no importa el tamaño de la adversidad, lo que importa es las ganas y la fuerza de voluntad con que las enfrentamos. A esa persona, por darme su amor infinito, por sus consejos, regaños, por enseñarme que la vida hay que disfrutarla pero también hay que luchar y sacrificarse por lo que se quiere y por ser uno de los motores impulsores que me trajeron hasta este punto. A esa persona que aunque no se encuentra hoy entre nosotros, está muy cerca de mi corazón. Esa persona, es mi hermano Jesús Riverón Vila.

A mis innumerables e inigualables familiares y amigos del alma.

AGRADECIMIENTOS

A mis padres, por siempre estar ahí cuando los necesitaba, por confiar en mí en todo momento. Los quiero….

A mi hermano Fabian por poner a prueba mi paciencia, por tantos dolores de cabeza y enseñarme a ver la vida desde distintos puntos de vista.

A toda mi familia, por todo el apoyo incondicional en todo momento y por ayudarme a ser quien soy.

A mis tutores Berta Pallerols y Alexander A. Suárez por su guía, por sus consejos y por poder contar con ellos siempre. Sepan que son un ejemplo a seguir.

A todas las personas que nunca podría dejar de agradecer por todo el apoyo, amor y amistad, en especial a mis tías Amparo, Chichi, Noemí, Graciela, mis abuelos, a Chuchi, a Maité, a Mayi, a Peña, a mis vecinas Adis (La Negra), Yanis y Yailin, a Armando, a Dairon, a Evelin, a mi prima del alma Yania, a Arleen, a Yuraimis, a mi amigo Alean, entre otros más a los que les pido disculpa por no mencionar.

A mis compañeros de aula que tanto me ayudaron en especial a Hernán, Lalé, Eric, Héctor, Albertico, Aguilera, Arnaldo, Rafa, Andrés, Duverger, Hamlet, Chales y a los que se me quedan, los quiero mucho y les agradezco todo su apoyo.

A mis compañeros Ballester, Daniel y Medina por toda la ayuda.

A todos los que formaron parte de mi familia en estos 5 años, mi total agradecimiento.

A la profesora Yanexis, por haberme ayudado tanto sin el mínimo interés. Gracias de corazón!!!

A todos los profesores del Dpto. de Telecomunicaciones por su contribución en mi formación profesional; a los técnicos de laboratorios y demás profesores que aportaron su grano de arena en este proceso.

A todos aquellos que se me puedan haber quedado, y que de una forma u otra signifiquen algo para mí.

A todos, GRACIAS.

RESUMEN

En el presente trabajo se muestra la propuesta de una arquitectura de nodo *IEEE-1394* a partir de un sistema en un *Chip* (SoC) embebido en una FPGA (Arreglos Programables de Campos de Compuertas) para almacenar datos en tiempo real en un dispositivo de almacenamiento masivo de datos IDE (del inglés *Integrated Device Electronics*). En el sistema propuesto se emplea la arquitectura de interconexión *on– chip* Wishbone que es una propuesta de interfaz de uso general que define un estándar de intercambio de datos entre módulos IP, el cual se caracteriza por su simplicidad y flexibilidad. El objetivo es generar una arquitectura de nodo *IEEE-1394* escalable y flexible. Se detallan las características principales del protocolo *IEEE–1394*, la interfaz IDE, la propuesta de arquitectura del SoC Plasma-Wishbone y se describen los aspectos fundamentales del bus a implementar y la tarjeta de desarrollo a emplear.

Palabras clave: *IEEE – 1394*, FPGA, IDE, Wishbone, SoC**.**

ABSTRACT

The present work shows the development of a new IEEE-1394 node architecture based on an embedded FPGA (Field Programmable Gate Array) System on Chip to acquire and store data in real time on a massive storage data device IDE (Integrated Device Electronics). The system has been built using Wishbone SoC interconnection architecture, Wishbone is a general purpose interface that defines a standard for data exchange between IP modules, which it's characterized by simplicity and flexibility. The goal is to generate scalable & flexible IEEE – 1394 node architecture. The document details the main features of IEEE – 1394 protocol, the IDE interface and the proposed SoC Plasma-Wishbone architecture. Finally, a description of both, the bus architecture and the development kit are also included.

***Keywords**: IEEE – 1394*, FPGA, IDE, Wishbone, SoC**.**

ÍNDICE

INTRODUCCIÓN

En el año 1986 *Apple™ Computer*, hoy *Apple™* [1], presentó una especificación de bus de alto rendimiento para microcomputadoras que denominó *FireWire™*. El bus se mostraba como una interfaz de comunicación serie entre periféricos y la computadora con direccionamiento de 64 bits, según el estándar *IEEE Std 1212-1991 Command and Status Register (CSR) Architecture* [2]. En el año 1994, la especificación inicial fue sometida a evaluación por el comité de estandarización del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y posteriormente, en diciembre de 1995 se aprueba como el estándar *IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus* [3].

Desde su surgimiento el estándar ha evolucionado hasta el presente a través de una serie de revisiones y suplementos que han ido ampliando la versión inicial. En el año 2000 se presentó la primera revisión del estándar denominada *IEEE Standard for a High Performance Serial Bus — Amendment 1* más conocida como *IEEE Std 1394a-2000* [4] que incluía una serie de mejoras y correcciones fundamentalmente en la capa física (*phy*). En esta revisión se establecen los elementos fundamentales de administración de energía en el bus y se aclaran algunos aspectos aún oscuros relativos a la parte del protocolo a implementar en *software*. En el año 2001 una nueva revisión del estándar *IEEE 1212* conocida como *IEEE 1212-2001* [5] clarificaba la versión original permitiendo solventar conflictos con *IEEE – 1394* definiendo de forma más precisa la Arquitectura CSR, especialmente en la ROM de configuración. En el año 2002 se publicó otra revisión, esta vez denominada *IEEE 1394b-2002* [6] con una nueva ampliación de las capacidades de la capa física pero manteniendo total compatibilidad con la revisión anterior.

La base de toda la tecnología del bus serie *IEEE – 1394* se continuó ampliando, en el 2004 se publicó *IEEE 1394.1-2004 Bridging* [7] que norma la interconexión (*bridge*) entre dos buses *IEEE – 1394*. En abril del 2006 se aprobó una nueva norma conocida como P1394c [8], que permite la coexistencia en el mismo cable (CAT5e) de *IEEE – 1394* y *Gigabit Ethernet (GigE)*. Esta normativa permite la existencia de dos redes lógicas sobre un mismo medio físico sin la necesidad de un puente (*bridge*) entre ambos protocolos. Las normativas de administración de energía se ampliaron y precisaron cuando la organización *1394 Trade Association* [9] publicó una serie de normas para regular la administración de energía en el bus:

- *1394 TA Power Spec Part 1: Cable Power Distribution* [10], especifica el procedimiento a seguir por los nodos de múltiples puertos que deben distribuir, limitar, transmitir o consumir la energía disponible a través del par de alimentación del cable 1394.

- *1394 TA Power Spec Part 2: Suspend/Resume* [11], especifica la implementación y utilización de los mecanismos de ahorro de energía por parte de los nodos en el bus.
- *1394 TA Power Spec Part 3: Power State Management* [12], describe el modelo a emplear para administrar los estados de energía dentro de un nodo 1394 y en todo el bus.

En febrero de 2010 se publica un documento adicional por *1394 Trade Association* con el nombre *FireWire Design Guide* [13]. Este documento constituye referencia y guía para la implementación del estándar tanto en dispositivos complejos como computadoras personales o en dispositivos de complejidad menor como electrodomésticos.

Conjuntamente a la ampliación y perfeccionamiento del estándar han surgido nuevos campos de aplicación del protocolo, algunos de ellos de rápido crecimiento en los últimos años:

- Electrónica de consumo.
- Industria de computadoras personales.
- Industria automovilística.
- Industria militar y aeroespacial.
- Medios no estándares.
- Instrumentación y control industrial.

Dentro de la electrónica de consumo *IEEE – 1394* ha encontrado aplicación importante en equipos electrodomésticos de AV (Audio/Vídeo) y las tecnologías asociadas. Más de 50 especificaciones diferentes se han publicado relacionadas con esta clase de dispositivos. Las normas fundamentales en este campo de aplicación son:

- *AV/C General Specification V3.0* [14], define los comandos generales para el control de dispositivos electrónicos AV/C.
- *MPEG4 over 1394* [15], es un reporte técnico que incluye las modificaciones necesarias en IEC 61884-4 [16] para el transporte del formato MPEG4 sobre *IEEE – 1394*.
- *IEC 61883-6*[17], es un estándar que describe el protocolo de transporte de datos de audio/música sobre *IEEE – 1394*.
- *Digital Transmission Licensing Authority (DTLA)* [18], autoridad de licencia a terceros creada para licenciar los mecanismos de protección de copia en transmisiones digitales

(léase audio y vídeo) ideados por las compañías Intel, Sony, Matsushita (MEI), Hitachi y Toshiba (5C).

En la industria de la computadora personal se han definido nuevas normas para la aplicación de *IEEE – 1394* tanto en periféricos como en la propia computadora:

- *Open Host Controller Interface (OHCI)* [19]. Esta especificación define un conjunto de registros y servicios comunes para un controlador de *host* genérico (GHC) *IEEE – 1394* en la computadora usando acceso directo a memoria (DMA). La mayoría de las implementaciones utilizan PCI y en la actualidad también *PCI Express* (PCIe).

- *Microsoft 1394 Plug & Play Specification* [20]. Referencias editadas por Microsoft para la asistencia en el diseño robusto e ínter operable de dispositivos compatibles con *IEEE – 1394*.

- *IP over 1394* [21] – [23]. Define los métodos, códigos y estructuras de datos necesarios para el transporte de bloques de datos (datagramas) del protocolo IP sobre *IEEE - 1394*. Estas especificaciones incluyen además los protocolos para DHCP.

- *Serial Bus Protocol 2 (SBP-2) & 3 (SBP-3)* [24], [25]. Es una especificación de protocolo para el transporte de datos y comandos asíncronos (SBP-2) y datos isócronos (sólo SBP-3) que ha ganado popularidad entre los fabricantes de periféricos para computadoras. Entre los periféricos que emplean estos protocolos se encuentran: dispositivos de almacenamiento masivo, impresoras e impresoras multifuncionales, *scanners*, etc.

En la industria automovilística las aplicaciones están orientadas a sistemas de entretenimiento. Las principales especificaciones para la aplicación de *IEEE – 1394* en esta área son:

- *1394 Automotive Specification (IDB-1394)* [26]. Especifica las capas del protocolo *IEEE–1394* que serán implementadas en los automóviles.

- *PMD for Fiber Optic Wake-on-LAN* [27]. Describe la implementación de las funciones dormir y despertar (*sleep* & *wake*) para los transmisores-receptores (*transceivers*) de fibra óptica y especifica las dimensiones de éstos para aplicaciones en la industria automovilística.

En la industria militar y aeroespacial se utiliza principalmente el estándar *IEEE – 1394b*, para lo cual se define la siguiente especificación:

- *1394b for Military Applications* [28] – [30]. Establece los requisitos para el uso de *IEEE - 1394b* como una red de bus de datos en vehículos militares y aeroespaciales. Además define el concepto de las operaciones y la circulación de información en la red. Entre sus aplicaciones en este campo se tienen los aviones *F – 22 Raptor* y *F – 35 Lightning II*. También se emplea en el transbordador espacial de la NASA (*National Aeronautics and Space Administration*) para monitorear restos (espuma, hielo) que pueden chocar con la nave espacial durante el lanzamiento.

IEEE – 1394 se ha expandido a diferentes medios físicos diferentes a los de la especificación inicial. Estos medios son conocidos como medios no estándares, a estas especificaciones adicionales se asocian las redes domésticas y la tecnología inalámbrica (*wireless*), para las cuales se tienen los siguientes documentos:

- *Video Electronics Standards Association Home Network* [31]. Describe la capa física, la capa de enlace de datos, el protocolo entre capas y los servicios relacionados para una red doméstica.

- *User Interface for Home Networks* [32]. Define un método de interfaz usuario – computadora que provee servicios a la red doméstica y permite al usuario el control de dispositivos conectados a ésta a través de interfaces gráficas de usuario accesibles con el navegador de internet.

- *Digital Video Broadcasting (DVB): 1394 Home Network Segment* [33]. Define cómo se transporta el tráfico IP para servicios de DVB sobre la tecnología *IEEE – 1394*, incluyendo la encapsulación de paquetes IP en los paquetes *IEEE – 1394*.

- *Digital Video Broadcasting: Home Local Network* [34]. Normaliza la topología, las interfaces físicas y una pila completa de protocolos para red local doméstica (HLN) basada en *IEEE – 1394*.

- *Protocol Adaptation Layer for IEEE 1394 over IEEE 802.15.3* [35].Especifica los métodos para reproducir la infraestructura *IEEE – 1394* empleando las prestaciones del *Std IEEE 802.15.3 – 2003* e implementar puentes (*bridges)* de *IEEE P1394.1* en este mismo dominio.

En el campo del control y la instrumentación industrial también se ha aplicado el protocolo *IEEE – 1394*. En este último campo se destacan las cámaras destinadas al usuario y las cámaras con fines industriales y de instrumentación. En esta dirección se han definido un conjunto de especificaciones dentro de las que sobresalen:

- *Industrial & Instrumentation Digital Camera (IIDC) V1.31* [36]. Éste estándar constituye una guía de diseño para fabricantes de cámara digitales que usan *IEEE - 1394* como interconexión entre la cámara y la PC; además proporciona interoperabilidad para este tipo de dispositivos. Define los registros de control y estado, y los procedimientos de control para las cámaras digitales basadas en *IEEE – 1394* para uso industrial.

- *Industrial & Instrumentation Control Protocol (IICP)* [37]. Es un protocolo de comunicación similar a AV/C (*Audio Video Control*) que tiene aplicación en la automatización industrial y las comunicaciones en la instrumentación.

- *IEEE 488 over 1394* [38]. Es el protocolo utilizado para el transporte de los comandos del estándar *IEEE – 488* [39] sobre *IEEE – 1394*. *IEEE488* es una especificación de bus paralelo de 8 bits de corto alcance que se utiliza en dispositivos de control automático.

- *1394 Automation Protocol* [40]. Esta especificación permite el control sincronizado y el intercambio de datos en tiempo real para dispositivos industriales como sensores, actuadores, motores, entre otros.

En Cuba, diversas instituciones hospitalarias como el Centro Oftalmológico del Hospital Clínico Quirúrgico Dr. Juan B. Zayas de la provincia de Santiago de Cuba, cuentan con tecnología médica que emplean cámaras compatibles con la norma IIDC, ejemplo de ello es la cámara de retina TRC 50-DX [41] del fabricante de equipos médicos TOPCON [42]. Ésta cámara se utiliza para tomar imágenes o video del segmento posterior del ojo a través de la pupila. Este dispositivo está equipado con dos cámaras, una de las cuales es la *AVT Dolphin F-201B* [43] que es compatible con IIDC versión 1.31.

En laboratorios de microscopía donde es de interés el procesamiento de imágenes digitales fijas y en movimiento para su estudio inmediato o *a posteriori* en tareas de análisis, diagnóstico y tratamiento se emplean cámaras acopladas a microscopios ópticos basadas en IIDC. El laboratorio de microscopía del Departamento de Ingeniería Biomédica de la Universidad de Oriente cuenta con el equipamiento necesario para la adquisición de imágenes y video a través de la cámara digital BST – HDCE en tiempo real. Para el manejo de este tipo de flujo de datos es necesario un protocolo de comunicación que permita o garantice la entrega de estos datos a una alta velocidad, el protocolo que responde a estos requisitos es el *IEEE – 1394*, por tanto es el que dicha cámara utiliza para la conexión y transmisión de la información.

Actualmente en la institución se cuenta con una única computadora que incorpora una interfaz *IEEE – 1394* en la placa base ASUS P5L – VM 1394. El soporte en *hardware* lo proporciona un

chip VLSI de VIA, específicamente VIA *Fire* II VT6308 1394a. Este *chip* proporciona soporte OHCI e interfaz PCI 2.1. La computadora disponible opera con el SO Windows y el *driver* se proporciona en un CD que acompaña la cámara. En estas condiciones la explotación de la cámara está limitada a esta única computadora pues no se disponen de otras computadoras que incorporen esta tecnología. Otra cuestión negativa que impide hacer extensiva la explotación del dispositivo es que para configurar la cámara es necesaria la ejecución de un *software* que se comunica con el *driver* del dispositivo en la computadora; de manera que acciones básicas como encender, apagar y establecer el modo de operación de la cámara requieren el uso de este *software*.

La necesidad de un sistema en FPGA que soporte en tiempo real, el almacenamiento temporal y/o permanente del flujo masivo de datos que requieren los modos de comunicación isócronos del protocolo *IEEE-1394*; es la situación que define el **problema de la investigación** que se toma como punto de partida de este trabajo.

En esta entidad se cuenta con varios *kits* de desarrollo con Arreglos Programables de Campos de Compuertas (FPGA) para el diseño e implementación de dispositivos digitales. Esta disponibilidad constituye el soporte de *hardware* y la estructura electrónica sobre la cual se desarrolla este trabajo.

Siendo el **objeto** de la investigación los controladores de comunicación que utilicen el protocolo *IEEE – 1394*.

Debido a que existen diversos controladores para las diferentes capas que componen este protocolo, en este trabajo se define como **campo de acción**:

Los controladores de capa de enlace (LLC) *IEEE – 1394* en FPGA.

Con el **objetivo** de:

Diseñar una arquitectura de nodo *IEEE – 1394* embebida en una FPGA para el almacenamiento de los flujos de datos requeridos por los modos isócronos de este protocolo en un dispositivo de almacenamiento masivo IDE.

Con los **objetivos específicos** siguientes:

1. Determinar los requerimientos de diseño del sistema de almacenamiento.

2. Proponer una arquitectura robusta basada en un bus de alto rendimiento, escalable y flexible para la transmisión y recepción de datos vía *IEEE – 1394*.

3. Verificar el comportamiento del sistema de almacenamiento.

Si se diseña en FPGA una plataforma con bus de alto rendimiento que incluya un módulo de control de dispositivo de almacenamiento masivo será posible un sistema que permita el almacenamiento en tiempo real del flujo de datos requerido por los modos de comunicación isócronos del protocolo *IEEE-1394*. Lo enunciado anteriormente constituye la **hipótesis**.

Para el desarrollo lógico del proceso de investigación se han planteado las siguientes **tareas**:

1. Caracterizar el protocolo de comunicación *IEEE – 1394*.
2. Estudiar la herramienta de *software* para diseñar el *hardware* en la FPGA a utilizar en el trabajo.
3. Estudiar la interfaz IDE.
4. Evaluar los requisitos que debe cumplir el controlador de capa de enlace *IEEE – 1394* para lograr el almacenamiento temporal y/o permanente del flujo masivo datos que requieren los modos de comunicación isócronos del protocolo *IEEE-1394.*
5. Diseñar un sistema para almacenar los flujos de datos requeridos por los modos isócronos del protocolo *IEEE-1394* en un dispositivo de almacenamiento masivo IDE.
6. Redactar el informe.

La investigación se desarrolla sobre la base de la concepción **dialéctico materialista** partiendo de los principios establecidos para el diseño de estructuras con sistemas de descripción de *hardware*.

En el proceso investigativo se utilizó el método de **análisis y síntesis** para el estudio del contenido referente al diseño digital y sistemas de descripción de *hardware* en la bibliografía especializada, determinar los principales bloques funcionales del diseño y su interrelación, así como elegir las herramientas necesarias para su realización.

Otro método utilizado es el de **inducción-deducción**, ya que se partió de la visión general para determinar los aspectos esenciales del diseño a partir de modelos existentes. También se usó el método de **modelación y simulación** en el proceso creativo, al concebir las unidades como entes independientes conformadoras del diseño final y en la verificación del funcionamiento del mismo.

Se utilizan fundamentalmente los métodos **teórico y práctico**, ya que se ha logrado la concepción de un diseño partiendo de la teoría existente para el diseño de sistemas electrónicos sobre *hardware* programable FPGA y los controladores de capa de enlace *IEEE – 1394*.

La estructura del informe es: Resumen, Introducción, dos Capítulos, Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos.

En el **Capítulo 1** se estudian las principales características del protocolo de comunicación serie *IEEE – 1394*. Se explican brevemente las capas que lo componen; se ilustran los sistemas lógicos programables y dentro de estos las FPGA, discutiéndose sus características, aplicaciones y su utilización en el desarrollo de sistemas embebidos. Se hace referencia a los principales elementos disponibles en la tarjeta Spartan-3 de Xilinx®, que se utilizan en el desarrollo del diseño que se propone. Se explica brevemente la interfaz ATA (*Advanced Technology Attachment*) o PATA (*Parallel Advanced Technology Attachment*), originalmente conocido como IDE (*Integrated Device Electronics*),que no es más que un estándar de interfaz para la conexión de los dispositivos de almacenamiento masivo de datos y las unidades ópticas que utilizan el estándar derivado de ATA y el estándar ATAPI. Además de introducirse el tema de los módulos IP, se determina que se usará el procesador MIPS Lite del sistema en un *chip* Plasma para el desarrollo de la arquitectura de nodo *IEEE -1394*.

En el **Capítulo 2** se analiza uno de los buses *on – chip* más difundidos en la actualidad, el estándar libre Wishbone y se muestran las topologías soportadas por esta especificación. Se presenta el diseño de una arquitectura de nodo *IEEE-1394* para el almacenamiento de datos en tiempo real en un dispositivo de almacenamiento masivo de datos IDE. Para el diseño propuesto se han empleado módulos IP de libre distribución y Wishbone como el estándar de interconexión entre estos, debido a las ventajas que ofrece. También se especifican los bloques funcionales internos e interconexiones.

CAPÍTULO 1: MARCO TEÓRICO.

En el presente capítulo se realiza un estudio del protocolo *IEEE 1394-1995* [3] o bus HPSB (*High Performance Serial Bus*) ya que es el estándar para la transmisión de información de muchos dispositivos comerciales y entre ellos el de la cámara BST - HDCE; describiéndose las generalidades de este protocolo y exponiendo los conceptos fundamentales sobre los cuales está sustentado. Se hace un breve estudio de las metodologías actuales del diseño digital y de los Dispositivos Lógicos Programables, haciendo énfasis en los Arreglos Programables de Campos de Compuertas (FPGA). Se mencionan las características más relevantes y las aplicaciones en el desarrollo de sistemas embebidos. Se incluyen además referencias a los principales elementos disponibles en la tarjeta Spartan-3 de Xilinx®, que constituye el soporte de *hardware* en el desarrollo del diseño que se propone en este trabajo. Se realiza un breve estudio del estándar de interfaz para la conexión de los dispositivos de almacenamiento masivo de datos y las unidades ópticas que utilizan el estándar derivado de ATA y el estándar ATAPI conocido como IDE. Se escoge además el procesador MIPS Lite del sistema en un *chip* Plasma para el desarrollo de la arquitectura de nodo *IEEE -1394* ya que constituye un sistema que demanda menos recursos de área de la tecnología de implementación, factor que abarata los costos.

1.1 Protocolo *IEEE – 1394*.

La cámara BST - HDCE acoplada al microscopio óptico emplea el estándar *FireWire* o *IEEE – 1394*. Este estándar describe un bus serie *Plug&Play* de alta velocidad propuesto inicialmente por Apple™ con el nombre de *FireWire*™ y después por el Comité de Estándares de Microprocesadores y Microcomputadoras de la Sociedad de Computación del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) para su estandarización en 1995. Este estándar está a su vez basado en el estándar ISO/IEC 13213:1994 (ANSI/IEEE 1212), que describe una arquitectura de comunicación entre buses de sistemas de microcomputadoras a través de Registros de Comando y Estado (CSR).

1.1.1 Topología del Bus Serie *IEEE – 1394*.

El estándar *IEEE – 1394* especifica dos medios físicos (entornos) distintos sobre los cuales se implementa la topología del bus. El medio físico cable es el más empleado aunque también se especifica el entorno *backplane*. En el medio cable la interconexión entre los nodos es a través de hilos conductores de cobre; en el medio *backplane* la interconexión es a través de vías en el circuito impreso (PCB). Los nodos pueden interconectarse entre sí en cualquier medio sin

restricción. Para conectar buses con medios físicos distintos se debe utilizar un puente (*bridge*), ver Figura 1.1.

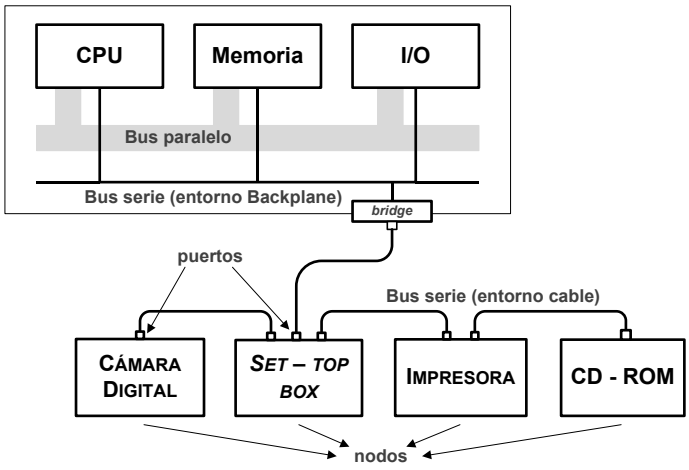


Fig. 1.1 *Topología física del Bus Serie.*

1.1.2 Medio físico cable y medio físico *backplane*.

La topología física para el entorno cable es una red no cíclica de ramas y extensión finitas, donde los lazos cerrados (bucles) no están soportados. El medio consta de dos pares de conductores para señales de datos y un par para la alimentación. Cada puerto consta de terminaciones, transmisores – receptores (*transceivers*) y lógica adicional. Debido a que en esta topología necesariamente existen nodos que no se conectan directamente entre sí, el cable y los puertos actúan como repetidores de bus entre los nodos simulando un bus lógico único.

El par de alimentación – tierra permite a la capa física de cada nodo continuar en funcionamiento incluso si la alimentación local del nodo desaparece. El par puede suministrar energía a un nodo completo si sus requerimientos son moderados proveyendo de 8 - 40 V de DC y hasta 1,5 A. En la especificación inicial del estándar para este medio físico se definen tres velocidades distintas: 100 Mb/s, 200 Mb/s y 400 Mb/s. La topología física del entorno *backplane* es un bus multipunto o tipo *daisy - chain*. El medio consta de dos conductores de terminación simple (*single-ended*) a lo largo del PCB que permiten a los nodos conectarse al bus con un OR cableado (*wired OR*). En el entorno *backplane* se definen dos velocidades 25 y 50 Mb/s, esta variante del estándar se usa menos y no será tratada en lo adelante.

1.1.3 Arquitectura del protocolo *IEEE – 1394*.

El protocolo *IEEE – 1394* describe tres de las capas del modelo *Open System Interface* (OSI) [44] de la *International Standards Organization* (ISO): la capa física (*PHY*), la capa de enlace (*LINK*) y la capa de transacciones (*TRANSACTION*), adicionalmente se define un nivel que se encarga de la gestión del bus (*BUS MANAGEMENT*) y que conecta entre sí todas las capas, ver Figura 1.2.

En cuanto al servicio de transferencia de datos este protocolo soporta dos servicios básicos:

1. Transferencia de datos asíncronos: este tipo de transferencia proporciona un protocolo de entrega de paquete de tamaño variable a una dirección explícita y retorna una confirmación de recepción (*acknowledge*). Esto permite la comprobación de errores y la retransmisión de datos.

2. Transferencia de datos isócronos: provee un protocolo de entrega de paquetes de tamaño variable transferidos a intervalos regulares. Es utilizada en la transferencia de datos tolerantes a error y con un tiempo crítico, como flujo de video o sonido.

El servicio de transferencia de datos asíncronos recibe y entrega datos desde/hacia la capa de transacciones, mientras que el servicio de transferencia de datos isócronos recibe y entrega directamente desde/hacia la capa de aplicación (ver Figura 1.2).

La cámara BST - HDCE se configura empleando los servicios de transferencia de datos asíncronos y siguiendo la arquitectura CSR definida en la especificación IIDC [36]. Por otra parte, emplea el servicio de transferencia isócrono para entregar la información de la imagen o vídeo.

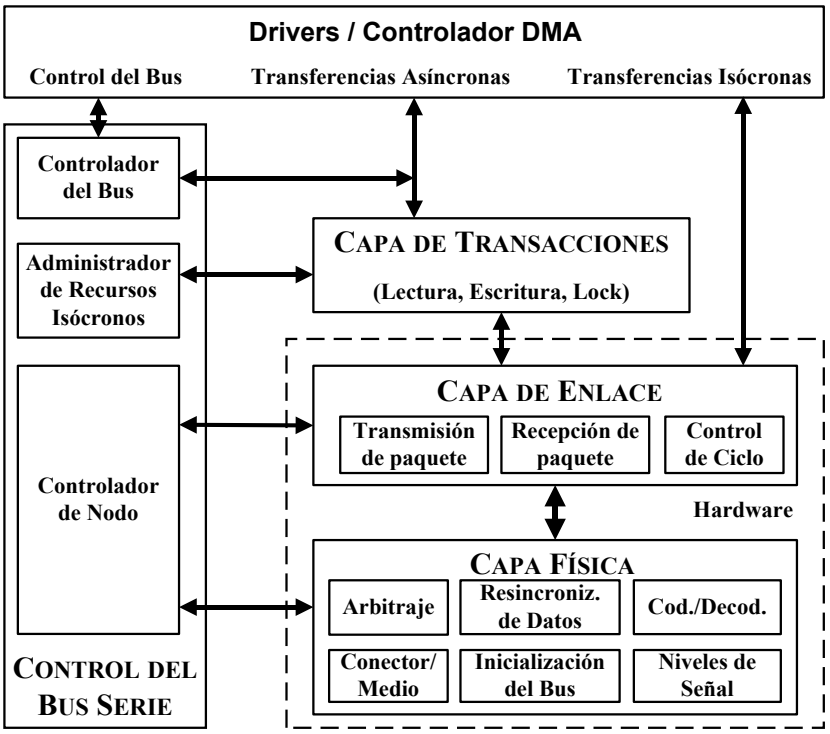


Fig. 1.2 *Estructura en capas del protocolo IEEE – 1394 para el entorno cable.*

La capa física (*PHY*) tiene tres funciones principales:

1. Convierte los símbolos lógicos utilizados por la capa de enlace en señales eléctricas sobre los diferentes medios del bus.

2. Proporciona un servicio de arbitraje, garantizando que solamente un nodo a la vez esté enviando datos.

3. Define la interfaz mecánica para el bus.

Existe una capa física diferente para cada entorno: cable y *backplane*. La capa física del entorno cable además provee servicio de repetición/resincronización de datos e inicialización automática del bus.

La capa de enlace (*LINK*) provee los servicios de transferencia de datos y confirmación de recepción de datagrama a la capa de transacciones. Proporciona direccionamiento, comprobación de datos y estructura de datos (*data framing*) para la transmisión y recepción de paquetes. También provee servicio de transferencia de datos isócronos directamente a la capa de aplicación, incluyendo la generación del paquete de inicio de ciclo (*cycle start*) empleado en la sincronización de los nodos. Una transferencia de capa de enlace es llamada subacción (*subaction*).

La capa de transacciones (*TRANSACTION*) define un protocolo completo de solicitud - respuesta para efectuar las transacciones de bus requeridas como soporte de la Arquitectura CSR. Define las acciones genéricas de lectura, escritura y bloqueo (*lock*). La capa de transacciones no provee ningún servicio para transacciones isócronas, no obstante, proporciona una vía para el control de las operaciones con datos isócronos a través de acciones de lectura y de bloqueo sobre el registro de control isócrono, presente en la Arquitectura CSR.

El nivel de control de bus (*BUS MANAGEMENT*) está también incluido en la especificación del protocolo, este nivel proporciona las funciones básicas de control y los registros del estándar CSR necesarios para controlar los nodos y/o manejar los recursos del bus. La entidad gestora del bus debe estar activa en un único nodo que ejerce la responsabilidad de dirección o control de todo el bus. En los nodos restantes (aquellos que no controlan el bus), este nivel consta únicamente de la entidad controlador de nodo. El controlador de recursos isócronos (IRM) es una entidad adicional que centraliza los servicios necesarios para administrar y monitorear las transacciones isócronas en el bus. El IRM contiene los registros CSR necesarios para la asignación de ancho de banda a canales o recursos isócronos en el bus.

1.1.4 Interfaz eléctrica.

Existen dos tipos de conectores *IEEE – 1394*: de cuatro terminales y de seis terminales (ver Figura 1.3). Los conectores de cuatro terminales se emplean en electrónica de consumo o dispositivos periféricos y los de seis terminales principalmente en concentradores (*hubs*) y computadoras.

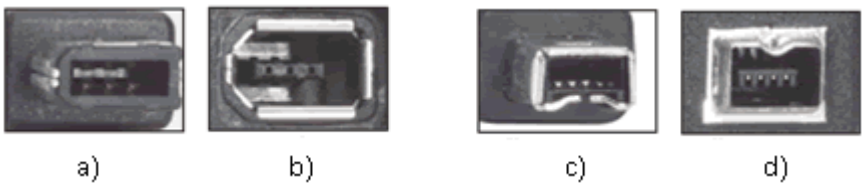


Fig. 1.3 *Conectores IEEE 1394. a) Conector macho de 6 terminales. b) Conector hembra de 6 terminales. c) Conector macho de 4 terminales. d) Conector hembra de 4 terminales.*

Los cables para cada caso tienen la siguiente estructura:

- Cable blindado de seis conductores. Tiene dos pares trenzados y blindados para la transmisión de la señal de datos denominados TPA y TPB, y otro par para la transmisión de energía (VP y VG) (Figura 1.4). Los nodos del bus de relativo bajo consumo pueden alimentarse a través de estos conductores.

- Cable blindado de cuatro conductores. Solamente consta de los dos pares trenzados blindados empleados en la transmisión de la señal de datos. Se utiliza en dispositivos que poseen alimentación local, es decir no obtienen la energía desde el bus.

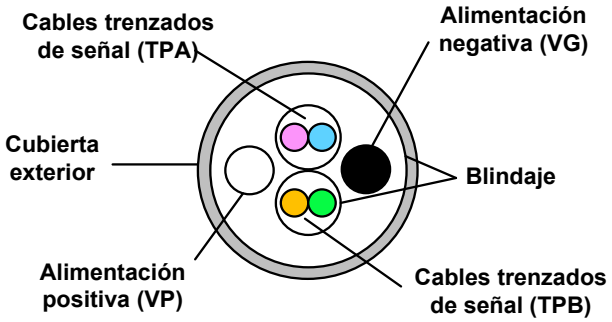


Fig. 1.4 *Sección transversal de un cable IEEE-1394 de seis conductores.*

Este diseño de los cables permite el uso de la codificación *data – strobe*, en los que TPA transmite la señal *strobe* y recibe los datos y TPB recibe la señal de *strobe* y transmite los datos.

TPA y TPB proporcionan un modo de señal común y diferencial que le permite dar soporte a las siguientes funciones:

- Reconocimiento de dispositivos conectados/desconectados.
- Reinicialización.
- Arbitraje.
- Transmisión de paquetes.
- Configuración automática.
- Señalización de velocidad.

La longitud del cable está limitada a 4,5 m debido a que las señales se distorsionan en exceso cuando sobrepasan esa distancia.

1.2 Metodología de diseño de Circuitos Digitales.

La evolución del grado de integración de los circuitos digitales desde la década del 70 se ha comportado según la **Ley de Moore**, enunciada en 1965 y que desde esa fecha predecía un comportamiento exponencial para la razón; número de transistores por milímetro cuadrado

(N/mm^2), más exactamente, la ley plantea que esta razón se duplica cada dos años (actualmente este tiempo ha disminuido).

El aumento de la complejidad de los dispositivos asegurada por las mejoras en la tecnología de fabricación y el grado de integración garantizó un incremento en funcionalidad, produciéndose así una disminución de los costos y un menor ciclo de producción, de otra manera, una mayor productividad. Sin embargo, esto aún constituye el reto de diseño fundamental, dado que por ejemplo, cifras del año 2005 [45], estimaban un 58% de crecimiento anual en complejidad de los circuitos integrados mientras que, el por ciento de crecimiento de la productividad era del 21% anual. Lo que evidencia una brecha significativa en la relación crecimiento en funcionalidad – crecimiento en productividad.

Esta dificultad ha llevado a la búsqueda de metodologías para aumentar la productividad en la realización de circuitos integrados (*IC*).

Para aumentar la productividad es necesario disminuir los costos de diseño. Algunas de las posibles variantes se pueden agrupar con un criterio común: el mejoramiento de las herramientas de diseño, que incluye:

- Automatizar los pasos de diseño.
- Minimizar la posibilidad de errores.
- Acortar el ciclo de diseño.

El mejoramiento de las herramientas de diseño, trajo aparejado el perfeccionamiento de las metodologías de diseño. De manera tal que en la actualidad es posible identificar los factores claves de la metodología de diseño contemporánea de circuitos integrados (*IC*). Estos factores se relacionan a continuación:

- Diseño jerárquico.
- Combinación de estilos de descripción.
- Reusabilidad (*cores o IP*).

El diseño jerárquico se sustenta en la existencia de niveles jerárquicos en el diseño de sistemas digitales:

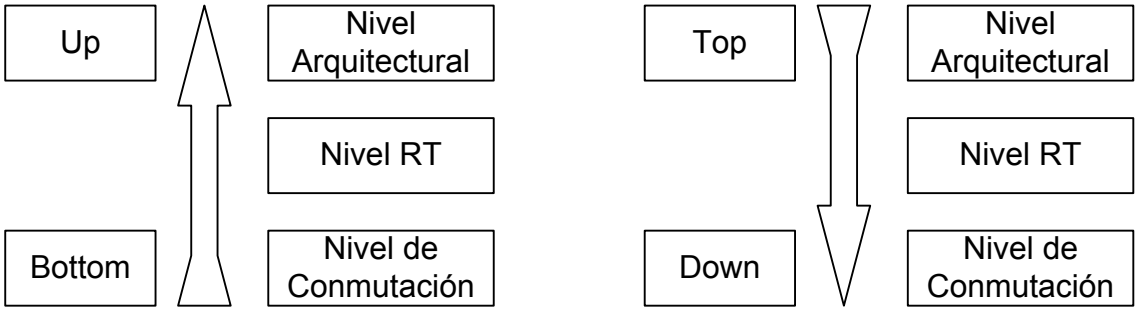
- Nivel de la arquitectura (*Top Level*).
- Nivel RT (Transferencia de registros)

- Nivel de conmutación (puertas y subsistemas) (*Down Level*)

Además agrupa dos metodologías de diseño, que se complementan mutuamente:

1. El término *Bottom – Up* se aplica al método de diseño mediante el cual se realiza la descripción del circuito o sistema que se pretende realizar, empezando por describir los componentes más pequeños del sistema para, más tarde, agruparlos en diferentes módulos, y estos a su vez en otros módulos hasta llegar a uno solo que representa el sistema completo que se pretende realizar.

2. El diseño *Top – Down*, es el proceso de capturar una idea en un alto nivel de abstracción, e implementar esa idea primero en un muy alto nivel, y después ir hacia abajo incrementando el nivel de detalle, según sea necesario. Esta forma de diseñar se puede imaginar como un sistema inicial que se ha dividido en diferentes módulos, cada uno de los cuales se encuentra a su vez subdividido, hasta llegar a los elementos primarios de la descripción. Esta metodología tiene numerosas ventajas, fundamentalmente en los aspectos de productividad del diseño, reusabilidad y detección de errores. En la figura 1.5 se muestra como se relacionan las dos metodologías anteriores con los niveles de abstracción.



***(a)** Metodología Bottom – Up.* ***(b)** Metodología Top – Down.*

***Fig. 1.5** Metodologías de diseño vs niveles jerárquicos.*

1.2.1 Arreglos Programables de Campos de Compuertas (FPGA).

El desarrollo de las metodologías de diseño mencionadas anteriormente no ha estado aislado de la producción de *IC* y consecuentemente con este en la actualidad se han desarrollado tres métodos fundamentales de realización de circuitos digitales:

- Circuitos diseñados a la medida: son específicos para una aplicación dada y requieren la disponibilidad de instalaciones (*facilities*) para su producción.

- Circuitos semidiseñados: son sistemas constituidos por una serie de bloques funcionales de complejidad variable, en los que es posible especificar las interconexiones entre estos bloques para conformar el diseño final.

- Circuitos de lógica programable: son circuitos donde la lógica de interconexión y los bloques funcionales son especificados mediante programas. Es común situarlos como una solución intermedia entre los circuitos digitales diseñados a la medida y los semidiseñados.

Los antecedentes de los dispositivos de lógica programable actuales se encuentran en los conocidos PLA (Arreglo Lógico Programable), PAL™ (Lógica de Arreglo Programable) y GAL™ (Arreglo Lógico Genérico) que son considerados dispositivos lógicos programables simples (SPLD).

Un escalón más arriba, se encuentran los dispositivos lógicos programables complejos (CPLD), que en dependencia de sus prestaciones equivalen a 2 y hasta 64 dispositivos SPLD.

Los Arreglos Programables de Campos de Compuertas (FPGA) surgieron a finales de la década del 90 como un escalón superior en la evolución de la lógica programable; que tenía a los dispositivos lógicos programables complejos (CPLD) como sus predecesores. Desde su surgimiento hasta el presente, estos dispositivos han enarbolado una serie de ventajas que los ha convertido en una solución factible a la hora de diseñar circuitos digitales.

1. Reducción de espacio: estos dispositivos pueden implementar funciones para las cuales serían necesarios numerosos circuitos integrados, implicando una reducción apreciable de la placa de circuito impreso.

2. Flexibilidad en el diseño: sin cambiar las placas del circuito se puede reprogramar y hacer un nuevo diseño.

3. Facilidad de diseño: existen herramientas de diseño electrónico asistido por computadora (*Electronic Design Automation*) que permiten aplicar las metodologías de diseño modernas en cada una de las fases del diseño de un circuito en lógica programable incluyendo en un único paquete herramientas de simulación y depuración.

4. Reducción de costo: el costo de un sistema que utiliza lógica programable se disminuye notablemente teniendo en cuenta la reducción en cuanto a:

- Placa de circuito impreso.

- Número de circuitos integrados.
- Sistemas de alimentación.
- Tiempo de diseño.
- Tiempo de reparación.

5. Fiabilidad: la disminución del número de circuitos integrados mejora la fiabilidad dado que el circuito es menos complejo por lo que la probabilidad de que alguna de sus partes falle es mínima.

6. Inmunidad al ruido: debido a que las líneas de interconexión están dentro del propio circuito, la probabilidad de que se acoplen ruidos en las mismas es también baja.

7. Protección de la propiedad intelectual: en estos dispositivos este punto tiene dos aristas fundamentales, por un lado el propio circuito posee diversos mecanismos para impedir cualquier operación de lectura después de ser programado. Por otro lado, los archivos de cadena de bits (*bitstream*) que se descargan sobre las FPGA no permiten realizar ninguna inferencia acerca del diseño original del sistema embebido.

A nivel mundial existen numerosos fabricantes de FPGA, no obstante, dos compañías, Altera™ [46] y Xilinx® [47] constituyen los líderes mundiales en el mercado de dispositivos FPGA. En lo adelante se tratará sólo con circuitos digitales de este último fabricante pues no se dispone de dispositivos ni *hardware* de desarrollo de Altera™. Seguidamente se exploran de manera muy breve algunos de los elementos fundamentales de la arquitectura de las FPGA de Xilinx®.

1.2.2 Flujo de diseño para dispositivos FPGA de XILINX®.

El proceso de diseño de un módulo *hardware* sobre una FPGA de Xilinx® se divide en cinco fases: descripción del modelo, síntesis, implementación, programación y verificación. La figura 1.6 muestra de forma esquemática el flujo de diseño de un módulo o circuito *hardware* dentro de una FPGA.

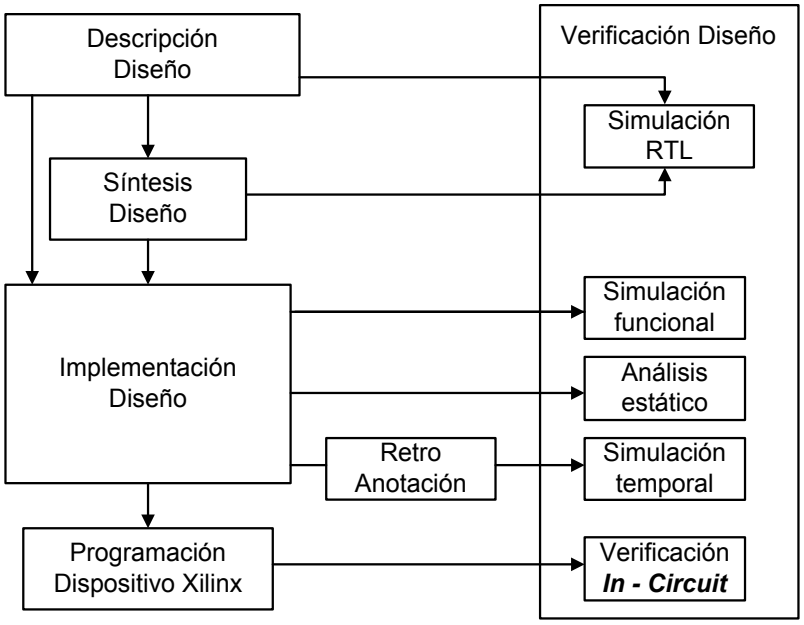


***Fig. 1.6** Flujo de diseño con FPGA de Xilinx®.*

1. Descripción: Proporciona información del sistema en forma de ecuaciones lógicas, tablas de la verdad, grafos de estados, un esquema o un conjunto de instrucciones en un lenguaje de descripción de *hardware* (HDL), utilizándose como entrada en la fase de síntesis.

2. Síntesis: Convierte la descripción inicial en una descripción estructural definida en un formato denominado lista de conexiones (*netlist*) que se usa como entrada de las fases de verificación e implementación.

3. Implementación: Se asignan los recursos de la FPGA seleccionada a los diferentes componentes del sistema incluidos en la lista de conexiones y se obtiene una lista de conexiones actualizada que contiene los retardos inherentes a cada componente. Si todo cumple las especificaciones del sistema el diseño ha concluido, restando sólo crearlo físicamente en el dispositivo, para ello en esta fase se genera un fichero con un formato que permite la programación.

4. Programación: Realiza la descarga del programa de configuración de la FPGA sobre su memoria interna, estableciendo todas las conexiones necesarias y programando cada CLB de manera independiente.

5. Verificación: La verificación es un proceso especial que transcurre de manera paralela a las primeras cuatro tareas, al finalizar cada una de las fases descripción, síntesis e

implementación puede comprobarse la validez del diseño mediante simulación y con herramientas de verificación *on – chip* puede verificarse el diseño ya programado en la FPGA.

Dentro de la verificación se incluye la simulación del diseño en diferentes niveles:

- Simulación funcional RTL, permite validar el modelo del diseño que se ha descrito.
- Simulación funcional post-síntesis a nivel de puertas, permite validar el modelo sintetizado.
- Simulación temporal post-implementación, permite validar la implementación del modelo usando la tecnología de una FPGA concreta y considerar retrasos así como verificar un comportamiento bastante aproximado al real.

Existen herramientas de verificación *on – chip* como *Chip Scope*™ [48], el cual integra dentro del dispositivo programable, junto a un diseño particular, los módulos IP de un analizador lógico y otras herramientas de evaluación para la puesta a punto del diseño. Un grupo de herramientas *software* se encargan de facilitar esta labor así como de proveer comunicación con estos componentes.

Chip Scope Pro 10.1 está integrado por varios módulos de propiedad intelectual (*cores*) y tres aplicaciones.

- *Integrated Controller core (ICON).*
- *Integrated Logic Analyzer core (ILA).*
- *Virtual Input/Output core (VIO).*
- *Integrated Bit Error Ratio core (IBERT).*
- *Xilinx CORE Generator™ tool.*
- *Core Inserter tool.*
- *Analyzer tool.*

1.3 Descripción del *Hardware*. Tarjeta de desarrollo SPARTAN™-3 de XILINX®.

Cuando se va a realizar un prototipo de sistema de procesamiento, ya sea de perfil automático, de comunicaciones u otros, es necesario determinar cuáles serán los elementos *hardware* y *software* para el desarrollo del mismo. Primeramente se debe hacer un estudio de las posibilidades y las limitantes de los elementos de *hardware* en los cuales será implementado el sistema.

La tarjeta Spartan - 3 Starter Kit Board de Digilent Inc. incluye una FPGA XC3S1000-4FT256 de Xilinx™, esta como soporte físico brinda un entorno de desarrollo para diseños que pueden ir desde un circuito digital simple con lógica combinacional hasta uno con sistemas embebidos y módulos de Propiedad Intelectual. La misma posee un circuito integrado FPGA, dispositivos de entrada/salida, visualizadores, interruptores y memorias RAM (Figura 1.7).

Además cuenta con una memoria FLASH programable desde la computadora, con la que se tendría el equivalente a una memoria ROM y con ello la posibilidad de hacer de la tarjeta de desarrollo un entorno perfecto para implementar aplicaciones no volátiles.

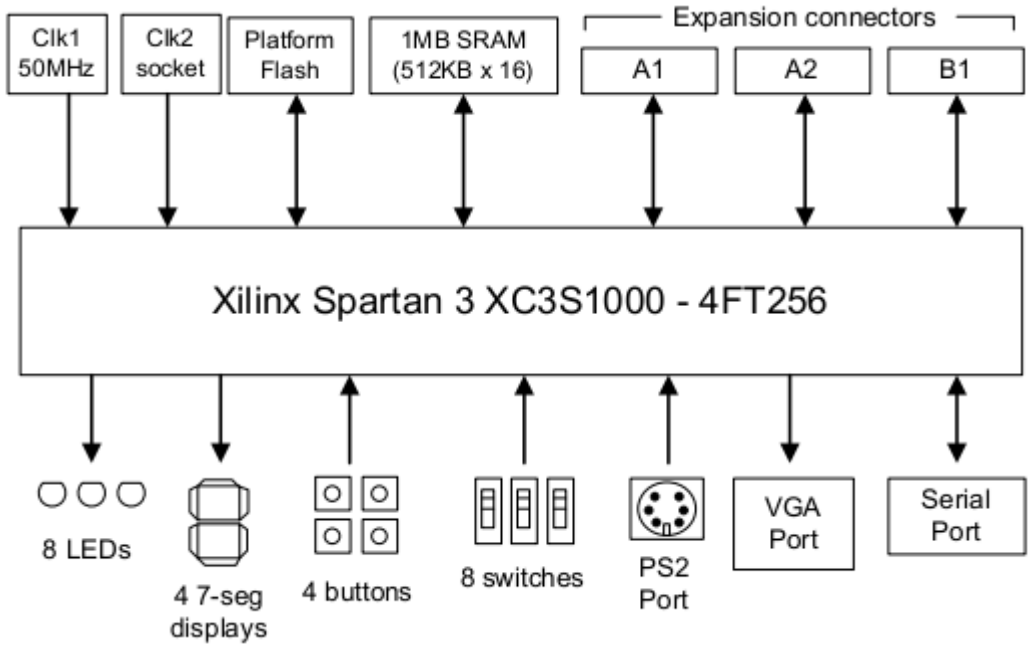


Fig. 1.7 *Diagrama en bloques de la tarjeta de desarrollo. Tomado del manual Spartan™-3 System Board.*

La Spartan - 3 Starter Kit Board incluye los siguientes componentes y características:

- 1 MB SRAM

- 1 Puerto VGA de 8 bit de color

- 1 Puerto Serie RS-232 con conector hembra DB9

- 1 Puerto PS2 para mouse o teclado

- 3 conectores de expansión

- 4 visualizadores 7 segmentos

- 8 leds

- 12 conmutadores: 4 *push button* y 8 *slide*

- Reloj externo 50 MHz

- Socket reloj externo

- Plataforma de Flash 4 MB

1.3.1 Periféricos de la tarjeta de desarrollo.

Con el propósito de controlar un monitor VGA, se tiene un conector DB15 como salida de video. A través de este se generan las señales de color y sincronismo necesarias para la composición de imágenes en la pantalla. La tarjeta asigna a cada píxel 3 bits, con lo que es permisible obtener 8 colores. Con esta posibilidad se pueden lograr aplicaciones para mostrar información o para desarrollar una interfaz gráfica de usuario.

Al igual que el conector para video, se cuenta con un conector para teclado. El conector es del tipo PS/2 basado en el estándar mini-DIN de 6 terminales. Mediante este se comunica el teclado usando un bus de dos líneas: datos y reloj. Con estas dos señales se generan palabras de 11 bits que contienen la información proveniente del teclado. Además la interface de teclado permite la transferencia bidireccional de datos. El teclado PS/2 usa códigos de exploración para comunicar el dato de la tecla presionada.

La comunicación serie se realiza usando el estándar RS-232. Este transmite y recibe las señales a través del conector hembra DB9 denominado como J2. El puerto serie es compatible con los de las PC, conectándose a ellos mediante un cable serie de 9 terminales estándar. Entre la FPGA y el conector DB9 se encuentra un adaptador de niveles del tipo MAX3232, encargado de desplazar los niveles lógicos de voltaje. Además se brinda un canal secundario serie RS-232 de transmisión/recepción para pruebas en la tarjeta, este canal equivale a los terminales del *jumper* J1.

1.4 Lenguaje de Descripción de *Hardware*.

Existe una amplia variedad de lenguajes de descripción de *hardware* (HDL) disponibles para especificar los diseños en los dispositivos lógicos programables como Verilog, System – C, ABEL o

VHDL. A pesar de que algunos diseñadores aún prefieren los diseños en esquemático y otras formas tradicionales de descripción, en la actualidad existe una marcada tendencia a la distribución e incluso comercialización de diseños electrónicos a través de descripciones en lenguajes HDL.

Tradicionalmente, las herramientas basadas en esquemático proporcionan a los diseñadores avanzados un mayor control de la localización y las conexiones físicas de la lógica en el dispositivo; pero se necesita un tiempo mayor para lograr un diseño determinado. Sin embargo, al emplearse las herramientas basadas en lenguajes, se logran diseños más rápidos pero con la posibilidad de no aprovechar eficientemente los recursos del dispositivo.

La síntesis de diseños basados en lenguajes ha tomado un gran auge en los últimos años, específicamente para los diseños en FPGA. En cualquier caso, conocer la arquitectura de los dispositivos y las herramientas de diseño y síntesis ayuda a realizar un mejor diseño.

El aumento de las prestaciones de las herramientas de diseño asistido por computadora (*Computer Aided Design*, *CAD*) para el diseño de sistemas electrónicos facilitan el desarrollo de circuitos cada vez más sofisticados, simplificando y acelerando el trabajo de diseño electrónico. La descripción de los sistemas electrónicos se realiza de forma simple o mixta a través de:

- Lenguajes de Descripción de *Hardware* (HDL): describen mediante un texto la estructura o el comportamiento del circuito, es independiente de la tecnología y tipo de FPGA. Con estos se aprovechan mejor los recursos de las FPGA y facilitan realizar grandes diseños con varios niveles jerárquicos. Son lenguajes mediante los cuales es posible describir un circuito digital o electrónico. La descripción puede ser de bloques donde se muestra la arquitectura del diseño, o de comportamiento, donde se describe el comportamiento del circuito en vez de los elementos de los que está compuesto.
- Esquemas electrónicos: la descripción está basada en un diagrama donde se muestran los diferentes componentes de un circuito. Se realiza a través de editores de esquemas electrónicos suministrados por los propios fabricantes de FPGA. Poseen bibliotecas de componentes de uso común y equivalentes a componentes convencionales no configurables.
- Grafos de estado: es un sistema gráfico con símbolos, que indican la secuencia del funcionamiento de determinados tipos de circuitos.

1.4.1 Lenguaje de Descripción de *Hardware* VHDL.

Con el desarrollo de herramientas de diseño más sofisticadas, fue apareciendo la necesidad de describir los circuitos con un mayor grado de abstracción, no desde el punto de vista estructural sino desde el punto de vista funcional.

Alrededor de 1981 el Departamento de defensa de los Estados Unidos desarrolla un proyecto llamado *Very High Speed Integrated Circuit*, VHSIC, cuyo objetivo era rentabilizar las inversiones en *hardware* haciendo más sencillo su mantenimiento. Se pretendía con ello resolver el problema de modificar el *hardware* diseñado en un proyecto para utilizarlo en otro, lo que no era posible porque no existía una herramienta adecuada para normar esta tarea.

En 1983 IBM, Intermetrics y Texas Instruments comenzaron el desarrollo de un lenguaje de diseño que permitiera la estandarización, facilitando con ello, el mantenimiento de los diseños y la depuración de los algoritmos.

Tras varias versiones llevadas a cabo con la colaboración de la industria y de las universidades, que *a posteriori* constituyeron etapas intermedias en el desarrollo del lenguaje, el IEEE publicó en diciembre de 1987 el estándar IEEE STD 1076-1987 [49] que constituyó el punto de partida de lo que después de cinco años sería ratificado como VHDL ("*Very High Speed Integrated Circuit / Hardware Description Language*").

La independencia en la metodología de diseño, su capacidad descriptiva en múltiples dominios y niveles de abstracción, la versatilidad para la descripción de sistemas complejos, la posibilidad de reutilización y la independencia de que goza con respecto a los fabricantes han hecho que VHDL se convierta en estándar como lenguaje de descripción de *hardware*.

En este trabajo se realiza el diseño a través de descripciones en VHDL, las herramientas empleadas para el diseño sobre FPGA fueron el entorno de desarrollo ISE 10.1 de Xilinx® y como simuladores el Xilinx ISE Simulator incorporado en el ISE 10.1 y el *ModelSim* SE 6.2b de *Mentor Graphics* [50]. El ISE 10.1 es una herramienta gratuita para generar diseños sobre FPGA de Xilinx®. Desde el punto de vista informático es un paquete de *software* que agrupa las herramientas de síntesis, el programa para generar los esquemáticos (XCS); una herramienta para la descarga a los dispositivos (*iMPACT*®).

Todas se agrupan en el navegador de proyecto (*Proyect Navigator*) que permite interactuar con estas herramientas mediante las barras de menú y diferentes iconos.

El simulador *ModelSim* SE 6.2b, es capaz de reconocer dispositivos como memorias RAM y ROM; agrupa los elementos que permiten simular de manera funcional los bloques lógicos configurables (CLB) básicos de Xilinx®. Esto permite especificar retrasos en la simulación. Admite además archivos de retrasos SDF (*Standard Delay Format*) lo que permite independizar los parámetros y agrupar las simulaciones para valores máximos, mínimos y típicos de los retrasos del dispositivo.

1.5 Capa de enlace *IEEE-1394* en FPGA.

En el año 2011 en Rodriguez M. [51] se presenta el diseño e implementación de un controlador de capa de enlace *IEEE - 1394* en lenguaje de descripción de *hardware* VHDL para FPGA. El diseño ha sido inspirado en una capa de enlace de *Texas Instruments*: TSB12LV31C [52]. La capa de enlace presentada brinda soporte para efectuar transacciones bidireccionales asíncronas (desde y hacia una capa física) y de recepción isócronas. La transmisión isócrona no está soportada.

La capa de enlace *IEEE - 1394* permite la interfaz con un microprocesador de 32 bits e incluye señales para la comunicación con la capa física que se utilice. En este trabajo se ha elegido el microprocesador MIPS Lite del sistema on - *chip* (SoC) Plasma de Steve Rhoads [53]. La interfaz con la capa física se basa en algunos *chips* comerciales como: FW803 [54], PDI1394P11A [55] y SBPH400-3 [56].

1.5.1 Arquitectura interna.

Internamente la capa de enlace está diseñada a partir de bloques funcionales interrelacionados. Los bloques que componen la capa de enlace son: Interfaz con microprocesador, Motor DMA (DMA ENG), Transmisor, Temporizador de ciclo, Controlador de ciclo, Receptor, Cálculo de CRC e Interfaz con capa física (*PHY - LINK*), ver Figura 1.8.

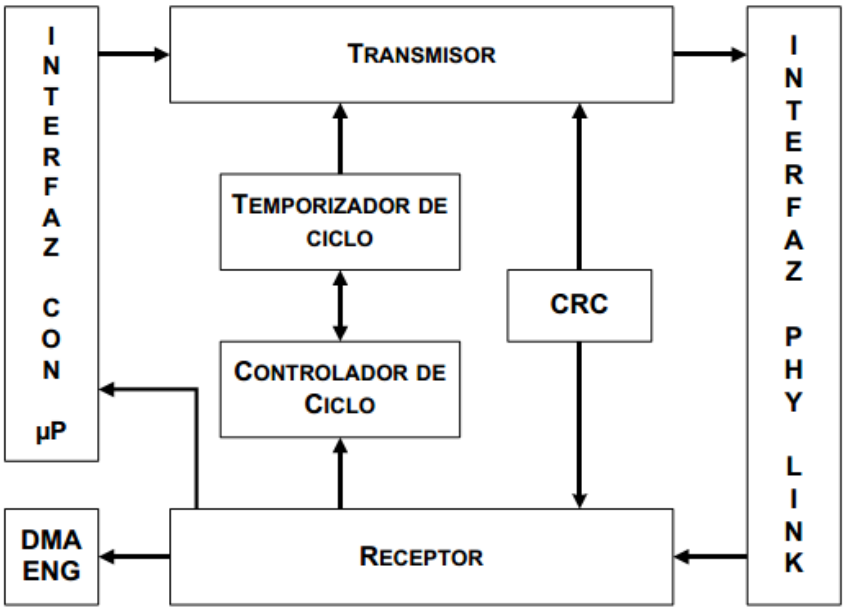


Fig. 1.8 *Diagrama en bloques de la capa de enlace.*

La interfaz física (*PHY - LINK*) provee los servicios del nivel físico (*PHY*) al transmisor y al receptor. Esto incluye el acceso al Bus Serie, envío y recepción de paquetes primarios, paquetes de confirmación y paquetes de capa física.

El transmisor tiene la función de enviar paquetes asíncronos recibidos desde la capa de transacciones hacia la interfaz *PHY - LINK*. El receptor de paquetes realiza la función de recepción de tramas tomando los datos proporcionados por la interfaz *PHY – LINK*.

El bloque CRC es un conjunto circuitos digitales para el cálculo del CRC de: (1) los datos a transmitir, (2) los datos recibidos y (3) del paquete de inicio de ciclo (*cycle start packet*).

El bloque Temporizador de ciclo se emplea en nodos que soportan transferencias de datos isócronos. Es un registro de 32 bits denominado CYCLE_TIME que está ubicado en la dirección 200h del espacio de registro inicial IEEE - 1212. Los 12 bits de menor orden de este registro contienen el valor de un contador módulo 3072, que se incrementa una vez cada período de reloj de 24.576 MHz (40.69 ns). Los siguientes 13 bits de orden más alto cuentan ciclos de 8 kHz (ó 125 µs). Finalmente los 7 bits más significativos o de mayor orden cuentan los segundos. El bloque Controlador de ciclo tiene la función de comenzar el ciclo isócrono cuando el nodo actual es el nodo raíz y además el maestro de ciclo.

El motor DMA es una interfaz para el acceso directo a memoria. Permite que los paquetes isócronos se guarden directamente en una zona de memoria incrementando un contador externo que apunta a la próxima dirección. La organización asumida para la memoria es en palabras de 32 bits, es decir, la transferencia de un *quadlet* desde la FIFO de recepción general (GRF). La utilidad fundamental de este bloque es la gestión de la transferencia de los datos desde la capa de enlace a la capa de aplicación. Esta unidad está enteramente concebida para Plasma SoC.

La interfaz con microprocesador permite la configuración de la capa de enlace. El bus de datos de la interfaz es de 32 bits y el bus de direcciones de 8 bits. Esta interfaz permite que las capas superiores (capa de transacción y nivel de control de bus) puedan acceder a los servicios de la capa de enlace y a los servicios de la capa física conectada. El banco de registros y sus direcciones se muestran en la Figura 1.9.

00h	Version
04h	MISC
08h	Control
0Ch	Interrupt
10h	Int_Mask
14h	CycleTime
18h	IsoPortN
1Ch – 20h	Reserved
24h	PhyAccess
28h – 2Ch	Reserved
30h	ATF_status
34h	BusRst
38h	Reserved
3Ch	GRF_status
40h – 4Ch	Reserved
54h	IsoCtrl
58h	IsoMode
5Ch	IsoHeader

Fig. 1.9 *Mapa de direcciones.*

Para el desarrollo de un nodo IEEE – 1394 capaz de conectarse a dispositivos que utilicen dicho protocolo y poder almacenar datos en un dispositivo de almacenamiento, es necesaria además de la implementación de la capa de enlace, la construcción de un sistema *hardware/software* que soporte el ancho de banda necesario y que sea compatible en la medida de lo posible con la

arquitectura de la capa de enlace. El nodo debe ser capaz además de implementar el resto de las funciones de la arquitectura vistas en el epígrafe 1.1, delimitadas en la Figura 1.10 por la línea continua de color rojo.

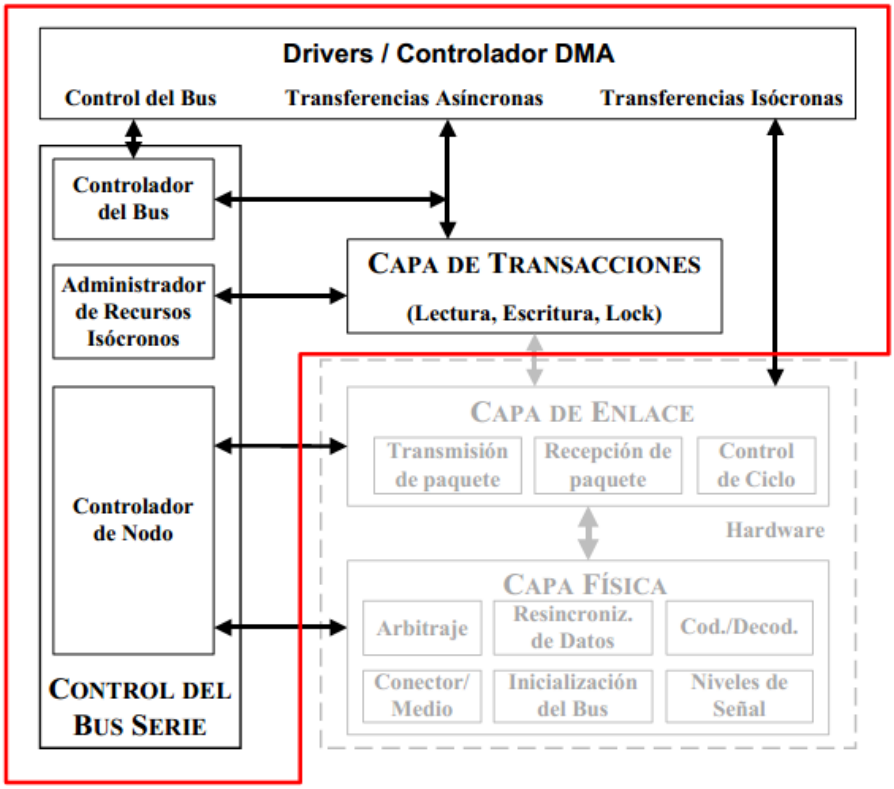


Fig. 1.10 *Algunas de las funciones de una arquitectura de nodo IEEE – 1394.*

1.6 Interfaz IDE.

La interfaz ATA o PATA, originalmente conocida como IDE, es un estándar de interfaz para la conexión de los dispositivos de almacenamiento masivo de datos y las unidades ópticas que utilizan el estándar derivado de ATA y el estándar ATAPI.

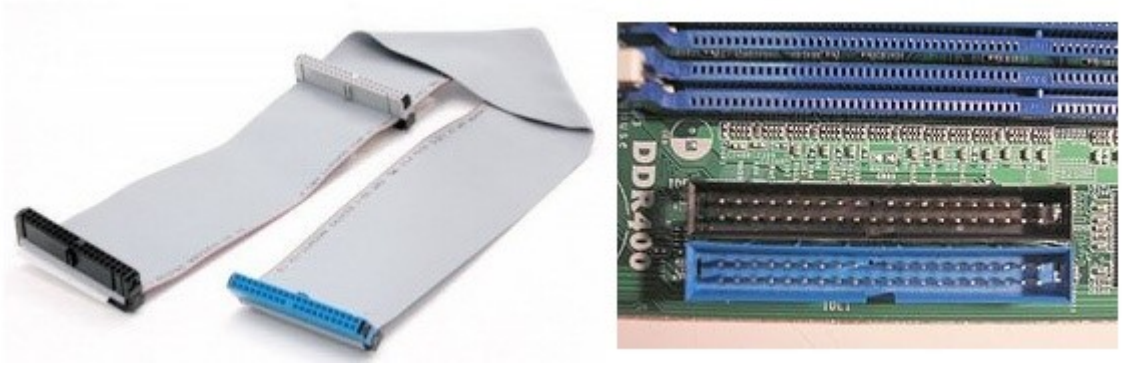


Fig. 1.11 *Conector PATA ♀ en un cable a la izquierda, dos conectores ♂ PATA en placa base a la derecha.*

Fig. 1.12 *Conector PATA ♀ de la interfaz IDE.*

Por lo general, el conector IDE/ATA de la placa base es un conector de 40 terminales al que se abrocha un cable plano, que va desde la placa base a la unidad de disco (se recomienda que el cable plano no exceda de 18 pulgadas (unos 45 centímetros), con objeto de que no se vea afectado por las interferencias). Estos terminales son un subconjunto de los 98 contactos de las ranuras ISA de 16 bits. La razón es que un controlador de disco nunca necesita más de 40 señales del bus ISA.

Las unidades muy pequeñas, principalmente de equipos portátiles, no disponen de espacio para un conector de alimentación independiente, por lo que utilizan un conector con 44 terminales, en el que los 4 terminales adicionales se utilizan para alimentación.

Tabla 1.1. Distribución de Terminales de la interfaz IDE.

Terminales	Distribución	Terminales	Distribución	Terminales	Distribución
1	*RESET-*	16	*Data 14*	31	*INTRQ*
2	*Ground*	17	*Data 0*	32	*No connect*
3	*Data 7*	18	*Data 15*	33	*addr 1*
4	*Data 8*	19	*Ground*	34	*GPIO_DMA66_Detect*
5	*Data 6*	20	*Key*	35	*addr 0*
6	*Data 9*	21	*DMARQ*	36	*addr 2*
7	*Data 5*	22	*Ground*	37	*CS0-*
8	*Data 10*	23	*DIOW-*	38	*CS1-*
9	*Data 4*	24	*Ground*	39	*DASP-*
10	*Data 11*	25	*DIOR-*	40	*Ground*
11	*Data 3*	26	*Ground*	41	*+ 5V (logic)*
12	*Data 12*	27	*IORDY*	42	*+ 5V (motor)*
13	*Data 2*	28	*Cable select*	43	*Ground*
14	*Data 13*	29	*DMACK-*	44	*TYPE (0-ATA)*
15	*Data 1*	30	*Ground*		

1.6.1 Surgimiento y evolución del estándar IDE.

"*Integrated Drive Electronics*", o más conocida como IDE, fue creada por la firma Western Digital [57] que en conjunto con *Conner*, definieron la interfaz EIDE (*Enhanced* IDE o IDE mejorado), al tiempo que *Seagate Technology* [58] y Quantum definieron el FAST ATA. Esto fue un encargo de *Compaq Computer* [59] para una nueva gama de computadoras personales, su característica más representativa era la implementación de un controlador en el propio disco duro, de ahí su denominación. Desde ese momento, únicamente se necesita una conexión entre el cable IDE y el bus del sistema, siendo posible implementarla en la placa base (como de hecho ya se hace desde los 486 DX4 PCI) o en tarjeta (equipos 486 VLB e inferiores). Igualmente se eliminó la necesidad de disponer de dos cables separados para control y datos, bastando con un cable de 40 hilos desde el bus al disco duro. Se estableció también el término ATA que define una serie de normas a las que deben acogerse los fabricantes de unidades de este tipo.

IDE permite transferencias de 4 Mb/s, aunque dispone de varios métodos para realizar estos movimientos de datos. La interfaz IDE supuso la simplificación en el proceso de instalación y configuración de discos duros, y estuvo durante un tiempo a la altura de las exigencias del

mercado. No obstante, no tardaron en ponerse en manifiesto ciertas modificaciones en su diseño, unas muy importantes eran de capacidad de almacenamiento, de conexión y de razones de transferencia; en efecto, la tasa de transferencia se iba quedando atrás ante la demanda cada vez mayor de prestaciones por parte del *software*. La interfaz EIDE o IDE mejorada, propuesta con el fin de sustituir a los antiguos controladores del PC XT, aumentando las prestaciones, dotó a la unidad de disco de la lógica necesaria para codificar-decodificar los datos, de manera que el controlador solo tuviese que efectuar peticiones y recibir respuestas. La especificación original permitía fabricar discos de buenas prestaciones y precio asequible, a costa de ciertas limitaciones que se hicieron evidentes con el paso del tiempo, como la imposibilidad para soportar discos de más de 528 MB o la limitación a dos dispositivos por controlador, este logra una mejora de flexibilidad y prestaciones, aumenta la capacidad, hasta 8,4 GB, y la tasa de transferencia empieza a subir a partir de los 10 Mb/s, según el modo de transferencia usado. Además, se implementaron dos sistemas de traducción de los parámetros físicos de la unidad, de forma que se pudiera acceder a superiores capacidades. Estos sistemas, denominados CHS y LBA aportaron ventajas innegables, ya que con mínimas modificaciones (aunque LBA exigía también cambios en la BIOS del PC) se podían acceder a las máximas capacidades permitidas.

Otra mejora de la interfaz EIDE se reflejó en el número de unidades que podían ser instaladas al mismo tiempo, que se aumentó a cuatro. Para ello se obligó a fabricantes de sistemas y de BIOS a soportar los controladores secundarios (dirección 170h, IRQ 15) siempre presentes en el diseño del PC pero nunca usados hasta el momento, de forma que se pudiera montar una unidad y otra esclava, configurada como secundaria. Más aún, se habilitó la posibilidad de instalar unidades CD-ROM y de cinta, coexistiendo pacíficamente en el sistema. A nivel externo, no existen prácticamente diferencias con el anterior IDE, en todo caso un menor tamaño o más bien una superior integración de un mayor número de componentes en el mismo espacio.

1.6.2 Transferencia y velocidad de la interfaz IDE.

Los dispositivos IDE pueden transferir información principalmente empleando dos métodos: Protocolo PIO (*Programmed I/O, en español* Entrada/Salida Programada. Modo de transferencia de datos) y DMA (Acceso Directo a Memoria. Modo de transferencia de datos.); el modo PIO es más lento que DMA porque requiere el uso del procesador para efectuar el traslado de datos, pues permite que los periféricos puedan intercambiar datos con la RAM con la ayuda de comandos administrados directamente por el procesador.

El otro método es el DMA (este acelera las velocidades de transferencia porque no accede al procesador de la computadora) así la CPU se desentiende de la transferencia, teniendo ésta lugar por mediación de un *chip* DMA dedicado. Con el IDE original se usaban los modos PIO 1 y 2, que podían llegar a unos 4 Mb/s de transferencia; el modo DMA del IDE original no superaba precisamente esa tasa, quedándose en unos 2 ó 3 Mb/s.

Existen dos tipos de modos de DMA:

- El DMA de "palabra única", que permite la transferencia de una sola palabra (2 bytes ó 16 bits) durante cada sesión de transferencia.
- El DMA de "palabras múltiples", que permite la transferencia sucesiva de varias palabras en cada sesión de transferencia.

El estándar ATA se basa originalmente en un modo de transferencia asincrónico, es decir, que el envío de comandos y de datos se ajusta al ancho de banda del bus y se realiza en cada flanco ascendente de la señal del reloj. Sin embargo, el envío de comandos y el envío de datos no ocurren de manera simultánea, es decir, un comando no puede ser enviado en tanto los datos no hayan sido recibidos y viceversa.

Para aumentar el rendimiento de los datos, se podría aumentar la frecuencia de señal del reloj, sin embargo, en una interfaz donde los datos se envían en paralelo, el aumento de la frecuencia ocasiona problemas de interferencia electromagnética.

De este modo, Ultra DMA (UDMA) fue diseñado con el fin de optimizar al máximo la interfaz ATA. El primer concepto de Ultra DMA consiste en utilizar los flancos ascendentes y descendentes de la señal para realizar las transferencias de datos, lo que significa un aumento de la velocidad en un 100% (con un aumento del rendimiento de 16,6 Mb/s a 33,3 Mb/s). Además, Ultra DMA incorpora el uso de códigos CRC que permiten la detección de errores de transmisión. Por lo tanto, los diferentes modos Ultra DMA definen la frecuencia de la transferencia de datos. Al producirse un error (cuando el CRC recibido no corresponde a los datos), la transferencia se produce en un modo Ultra DMA más bajo o incluso sin Ultra DMA.

Con la incorporación del modo Ultra DMA, se introdujo un nuevo tipo de cable de cinta que permite limitar la interferencia. Este tipo de cable de cinta añade 40 alambres (en un total de 80) entrelazados con los alambres de datos para poder aislarlos y tener los mismos conectores que el cable de cinta de 40 alambres.

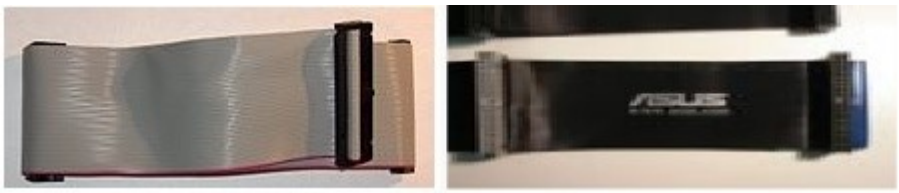


Fig. 1.13 *Cable IDE clásico de 40 alambres a la izquierda, cable IDE de 80 alambres a la derecha.*

Es necesario también tener presente que las velocidades de transferencia no son velocidades netas de transferencia de archivos, ya que estas quedan bastante por bajo. Por ejemplo, los 33 Mb/s de ultra ATA/33 quedan reducidos a unos 6 Mb/s reales. La siguiente tabla muestra un resumen del máximo rendimiento según el modo de transferencia de la interfaz IDE.

Tabla 1.2. Velocidad definida según el Modo de Transferencia.

Modo	Versión	Máxima Tasa de Transferencia(Mb/s)	Ciclos de Tiempo
PIO	0	3,3	600 ns
	1	5,2	383 ns
	2	8,3	240 ns
	3	11,1	180 ns
	4	16,7	120 ns
Single-word DMA	0	2,1	960 ns
	1	4,2	480 ns
	2	8,3	240 ns
Multi-word DMA	0	4,2	480 ns
	1	13,3	150 ns
	2	16,7	120 ns
	3	20	100 ns
	4	25	80 ns
Ultra DMA	0	16,7	240 ns ÷ 2
	1	25	160 ns ÷ 2
	2 (Ultra ATA/33)	33,3	120 ns ÷ 2
	3	44,4	90 ns ÷ 2
	4 (Ultra ATA/66)	66,7	60 ns ÷ 2
	5 (Ultra ATA/100)	100	40 ns ÷ 2
	6 (Ultra ATA/133)	133,3	30 ns ÷ 2
	7 (Ultra ATA/167)	166,6	24 ns ÷ 2

1.6.3 Ventajas y desventajas de la interfaz IDE.

La principal ventaja de los discos duros IDE es su precio, mientras que las prestaciones de los discos IDE se van acercando poco a poco a los SCSI, su precio se mantiene mucho más bajo, generalmente entre un 30% y un 50% en unidades de igual capacidad.

Todas las placas madre del mercado llevan integradas un controlador IDE, esto elimina la necesidad de un controlador PCI y su costo adicional. Además, prácticamente todas las placas madre del mercado tienen soporte para UDMA (*Ultra Disk Matching Architecture*), la cual permite transferencias de hasta 100 Mb/s. Este diseño tiene además la ventaja de que se reduce el número total de componentes, las trayectoria de las señales es más corta y por tanto menos vulnerable a las interferencias electromagnéticas.

El diseño original de ATA (dos dispositivos a un bus) tiene el inconveniente de que mientras se accede a un dispositivo, el otro dispositivo del mismo conector ATA no se puede usar. En algunos *chipset* (por ejemplo, *Intel FX triton*) no se podría usar siquiera el otro ATA a la vez. Este inconveniente está resuelto en SATA (*Serial Advanced Technology Attachment*) y en SCSI (*Small Computers System Interface*), ya que se utiliza un dispositivo en cada puerto. Por otra parte algunas áreas de las normas ATA dejaron cierta libertad a los diseñadores, lo que motivó comandos específicos de los proveedores. Esto hace que sea difícil dar formato de bajo nivel a estas unidades y redefinir sus tablas de sectores defectuosos, ya que los comandos cambian de un proveedor a otro. La mayoría de fabricantes de unidades ATA ofrecen el *software* de formateo a bajo nivel en sus sitios web.

Los discos ATA se extendieron más que los SCSI debido a su costo mucho menor, aunque su rendimiento también era más bajo. Debido a las limitaciones que tenía ATA, desarrollaron su sucesor, que se dio a conocer como Serial ATA, el cual mejoró considerablemente en rendimiento. Por tanto, hoy en día se está reduciendo progresivamente el uso de PATA, ya que ha sido sustituida por SATA.

1.7 OpenCores.

Dentro del marco del *hardware* libre, surge a finales de los 90 la iniciativa *OpenCores* [60], un amplio grupo de usuarios y empresas interesados en el desarrollo de *hardware* libre. El modelo de desarrollo se basa en la generación de componentes reutilizables que puedan inter - operar bajo ciertas circunstancias, y unirse para formar un sistema digital complejo. Estos componentes, también llamados *cores* o módulos IP, pueden ser considerados bloques básicos de construcción, para una posterior implementación del sistema sobre un ASIC (Circuito Integrado para

Aplicaciones Específicas) o FPGA. Esta metodología fomenta principalmente el abaratamiento de costos y la reutilización.

Estos componentes son desarrollados en distintos lenguajes de descripción de *hardware* (HDL) y se encuentran con licencias diversas (todas libres). La comunidad *OpenCores* se declara agnóstica con respecto al lenguaje y licencia de sus *cores*, quedando esta decisión a la elección de los usuarios. Este hecho provoca que al utilizar diferentes *cores* se deba tener en cuenta la posible incompatibilidad entre las licencias, pues no todas ellas son compatibles entre sí, más aún cuando se combinen estos *cores* con otros no libres. Esto es considerado como una manifestación más del carácter libre, pues favorece que los nuevos diseños desarrollados para ser interoperables, sean liberados también.

En el plano funcional, en *OpenCores* se potencia la interoperabilidad entre módulos IP fomentando el uso del bus estándar llamado Wishbone [61]. Aunque su uso no es obligatorio, más de la mitad de los *cores* lo usan como interfaz con el exterior, permitiendo así poder combinarlos en sistemas de gran complejidad a un costo mínimo. En *OpenCores* se pueden encontrar desde procesadores (CPU) hasta módulos criptográficos pasando por gran variedad de coprocesadores, controladores de comunicación, controladores de memoria, DSP, etc.

1.7.1 Plasma y Mips Lite.

Plasma es un sistema en un *chip*, ver Figura 1.14, desarrollado por Steve Rhoads, en cuyo núcleo se encuentra el procesador MIPS Lite de 32 bits que implementa la mayoría de las instrucciones del set de instrucciones MIPS I. Plasma es libre y está disponible en el sitio web de *OpenCores*; ha sido utilizado con éxito en un pequeño servidor web [62], y como controlador de cuatro robots de comunicación utilizando Virtex™ [63]. El procesador MIPS Lite tiene una arquitectura *Von Neumann* y es capaz de direccionar hasta 4 GB de memoria RAM distribuidos en 4 bancos de 1 GB cada uno. Es configurable en 2 y 3 etapas de *pipeline* y soporta los tipos de datos enteros: *byte* (8 bits), *word* (16 bits) y *double word* (32 bits).

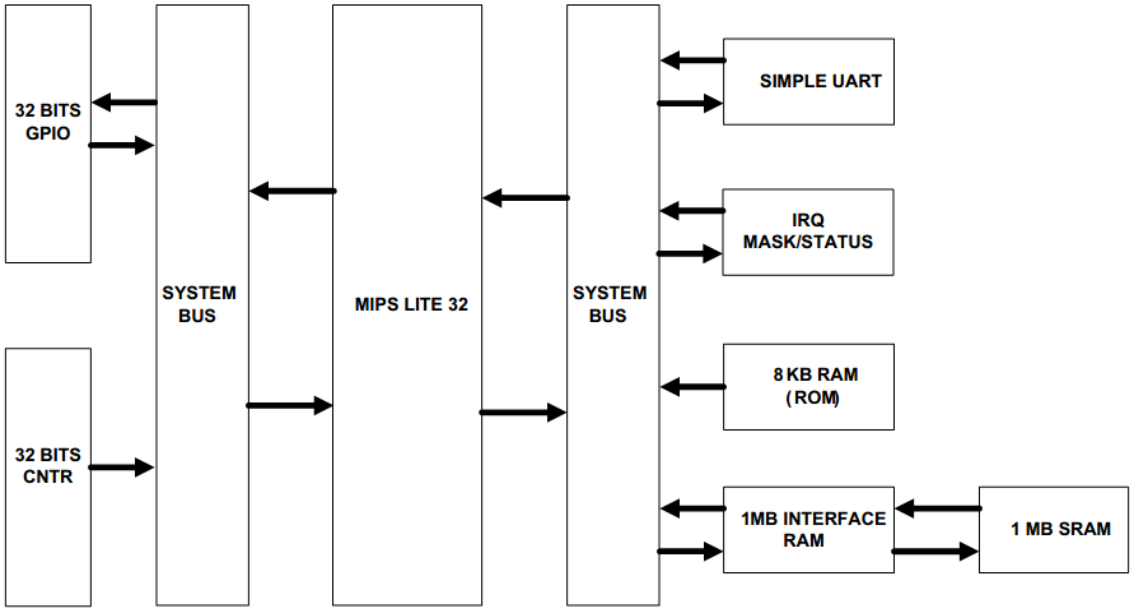


Fig. 1.14 *Plasma SoC (configuración mínima).*

Junto al procesador, en Plasma se incluye una interfaz serie a 57600 Baud, RAM de 8 kB, puertos de E/S de propósito general (GPIO), un contador, y opcionalmente un controlador de caché, una interfaz DMA – ETH y un controlador DRAM DDR. Una implementación típica alcanza de 33 a 36 MHz.

Para el desarrollo de aplicaciones está disponible el compilador ANSI C de GNU GCC para la arquitectura MIPS. El desarrollador ha proporcionado además herramientas de utilería, un simulador y un pequeño sistema operativo de tiempo real.

A pesar de ser una plataforma mucho más modesta en comparación con las plataformas LEON (microprocesador de 32 bits que incorpora un núcleo de arquitectura RISC (del inglés *Reduced Instruction Set Computer*, en español Computadora con Conjunto de Instrucciones Reducidas)) y *OpenRISC* (arquitectura de procesador en la que se basa la familia de procesadores libres *OpenRISC* 1xxx), precisamente por esta razón, Plasma y dentro de ella MIPS Lite constituyen un sistema que demanda menos recursos de área de la tecnología de implementación, factor que abarata los costos. Por otra parte, comparte con las plataformas más extensas el disponer de una cadena de herramientas GNU. Por estas razones, de Plasma se ha tomado el procesador MIPS Lite para el desarrollo de la arquitectura de nodo *IEEE -1394* que se propone en este trabajo. Seguidamente se profundiza en la arquitectura de MIPS Lite.

El CPU MIPS Lite es un procesador de 32-bit sintetizable, es capaz de ejecutar todas las instrucciones MIPS, excepto las operaciones de carga/almacenamiento no alineadas. Está implementado en lenguaje VHDL.

En la Figura 1.15 se muestra la arquitectura externa del procesador MIPS Lite.

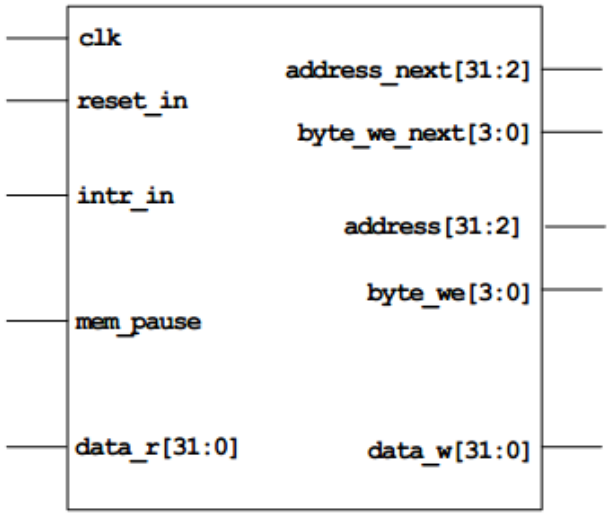


Fig. 1.15 *Arquitectura externa del procesador MIPS Lite.*

El bus del sistema está compuesto por 4 señales, dos de datos: ***data_r[31:0]***, como entrada y ***data_w[31:0]*** como salida, una señal de direcciones, ***address[31:2]*** y otra de control, ***byte_we[3:0]***. La señal de control (***byte_we***) es de 4 bits activos en nivel alto, cada bit representa la habilitación en escritura de un banco de memoria. Las señales de dirección y control se encuentran en dos versiones: valor actual y valor después del próximo flanco del reloj (añaden el sufijo ***next***) con el objetivo de acceder en un solo ciclo a la RAM sincrónica. Se incluyen además la señal de reloj (***clk***), reinicio (***reset_in***), una señal de entrada de interrupción (***intr_in***) y una señal para introducir estados de espera en accesos a memoria o periféricos (***mem_pause***).

La arquitectura interna está compuesta por 8 bloques funcionales, Figura 1.16:

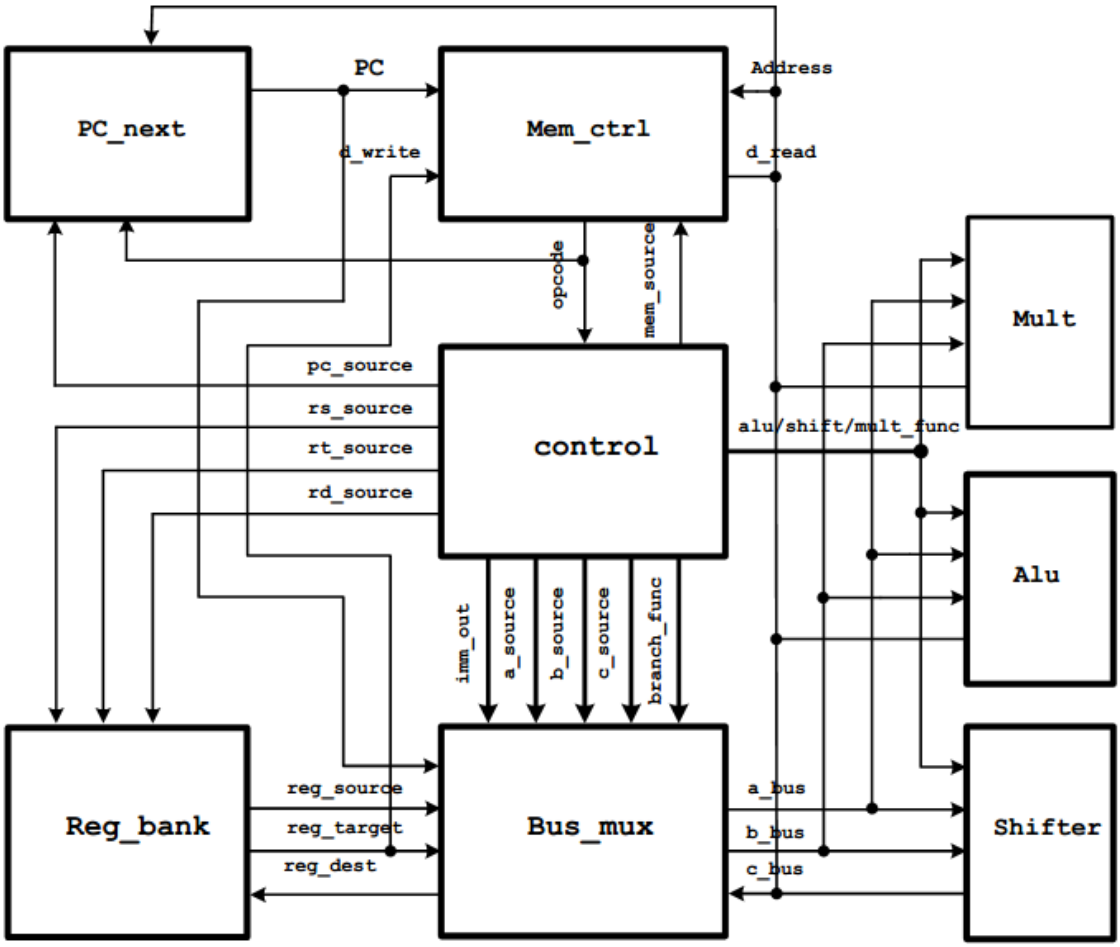


Fig. 1.16 *Arquitectura interna del procesador MIPS Lite (tomado del datasheet).*

Las funciones de cada unidad son las que se describen a continuación:

- Puntero de instrucción (PC_next): Determina las funciones lógicas del puntero de instrucción (IP) o contador de programa (PC) que apunta a la próxima instrucción a leer de la memoria.
- Controlador de memoria (Mem_ctrl): Determina el comportamiento del bus del sistema para las diferentes transacciones que soporta el procesador. Soporta los modos de organización de la memoria *Big* y *Little Endian*.
- Banco de registros (Reg_bank): Es un banco de 32 registros de propósito general similar al disponible en las arquitecturas MIPS convencionales. Incorpora dos canales de lectura y un canal de escritura.
- Unidad de Multiplexación de Buses (Bus_mux): Es el principal bloque de interconexión interna de los módulos que componen el procesador, en este sentido, se comporta como

un bus interno. En dependencia de las señales de la unidad de control dirige los operandos provenientes de múltiples fuentes a la unidad del camino de datos que corresponda.

- Multiplicador (Mult): Implementa las operaciones aritméticas de multiplicación y división. Esta unidad es opcional y para cada una de las operaciones toma 32 ciclos de reloj.
- Unidad Aritmética – Lógica (ALU): Implementa las operaciones aritméticas suma y resta, comparaciones y las operaciones lógicas habituales: AND, OR, XOR y NOR.
- Desplazador (Shifter): Es un desplazador combinacional de 32 bits. Permite las instrucciones de desplazamiento lógico.
- Unidad de control (Control): Es la unidad de control y el decodificador de instrucción al unísono. El bloque decodifica el *set* de instrucciones MIPS (32 bits) y lo convierte al formato *Very Long Instruction Word* (VLIW) (60 bits). Estas señales decodificadas son enviadas como señales de control hacia el resto de las unidades del procesador.

Las principales limitaciones del procesador MIPS Lite a juicio del autor se encuentran en las estrechas posibilidades de la interfaz del sistema (el bus del sistema). Para fundamentar este planteamiento se detallan a continuación estas limitaciones:

- Implementa un bus sencillo pero no estándar e incompatible con los buses más extendidos de la actualidad. Esto implica que los periféricos que se adicionen deban ser diseñados a la medida de este bus.
- El bus no incorpora protocolo de *handshake*. Lo que es una fuerte limitante para la adición de periféricos que incorporen y/o requieran esta funcionalidad.
- La topología en Plasma es de un bus compartido de único maestro, en esta estructura el maestro (microprocesador MIPS Lite) se conecta al resto de los periféricos (esclavos), de manera que toda actividad en el bus es iniciada por el maestro. Otros sistemas que incorporen uno o más maestros son difíciles de implementar manteniendo el rendimiento del conjunto.

Estas restricciones de escalabilidad, ya que la composición del bus no favorece la ampliación de periféricos y en algunas situaciones limita el rendimiento del sistema globalmente, conducen a la conclusión de que es necesaria la transformación del sistema de bus de MIPS Lite. Esta transformación debe estar dirigida a portar la CPU a un sistema de bus extendido universalmente y que elimine o mitigue las dificultades ya examinadas.

Como parte de las conclusiones de este capítulo se ha determinado que sea el procesador MIPS Lite de la plataforma SoC Plasma el elegido para implementar las funciones de las capas superiores del bus *IEEE – 1394*. Sin embargo, las limitaciones del SoC Plasma y del propio sistema de bus del procesador, han planteado la necesidad de migrar este procesador a una arquitectura de interconexión más extendida y con más posibilidades. La arquitectura de bus Wishbone ha sido la elegida por su carácter libre, abundante soporte y difusión para llevar a cabo dicha migración.

En el capítulo siguiente se analizan los buses en un *chip*, estudiándose con mayor profundidad la arquitectura de bus Wishbone y las posibilidades que brinda para la interconexión de módulos IP. Además de emplearse las capacidades de Wishbone, se emplean los elementos generales revisados en el Capítulo 1 para proponer una arquitectura de nodo *IEEE – 1394* que permita almacenar datos en tiempo real en un dispositivo de almacenamiento masivo de datos IDE empleando como estructura alternativa una FPGA posibilitando la comunicación entre dispositivos que utilicen el protocolo *IEEE – 1394* y un dispositivo IDE.

CAPÍTULO 2: ARQUITECTURA DE NODO *IEEE-1394* SOBRE FPGA.

En este capítulo se analizan los buses *on – chip*, haciendo énfasis en uno de los más difundidos en la actualidad, el estándar libre Wishbone. Se presenta el diseño de una arquitectura de nodo *IEEE-1394* para el almacenamiento de datos en un dispositivo IDE. Para el diseño se han empleado módulos IP de libre distribución y el estándar de interconexión entre estos es Wishbone. En el capítulo se detalla la propuesta de diseño punto por punto.

2.1 Buses.

Cada uno de los elementos de un sistema con microprocesador está unido a los demás mediante un conjunto de señales necesarias para la transferencia de información, a este conjunto de señales se le llama BUS. El bus está formado por un conjunto de señales con funciones afines que conectan las distintas partes del sistema. Al conjunto de buses de un sistema se denomina genéricamente bus del sistema.

Toda transferencia de información entre elementos de un sistema requiere de dos elementos, uno es el elemento origen de la transferencia, el que inicia la secuencia, y el otro, el que responde a esta secuencia que se denomina destino de la transferencia.

Los elementos que pueden iniciar una transferencia por el bus son los que se denominan elementos origen de la transferencia, estos elementos son la CPU y las unidades que dispongan de acceso directo a memoria. Las unidades destino de una transferencia son unidades de memoria o dispositivos de entrada/salida.

2.1.1 Buses en un *chip* (*on-chip*).

El proceso de miniaturización de la tecnología y el aumento del tamaño de los diseños ha llevado a circuitos integrados (*IC*) de miles de millones de transistores. En consecuencia, los fabricantes están integrando números crecientes de componentes en un *chip*. Un sistema en un *chip* (SoC) heterogéneo puede incluir uno o más componentes programables, tales como procesadores de propósito general, DSP, o módulos de propiedad intelectual (IP *cores*), como memoria, dispositivos de entrada/salida y otros circuitos de aplicaciones específicas. En este sentido un SoC es un *IC* que cumple la mayor parte o todas las funciones de un sistema electrónico completo.

En general, el rendimiento de un SoC depende fuertemente de la eficiencia de la estructura de su bus. El equilibrio de computación y comunicación en una aplicación o tarea depende de la

eficiencia en la transferencia de los datos por el bus. En ciertos casos los módulos IP, como elementos del SoC, se diseñan para diferentes tipos de interfaz y protocolos de comunicación. Integrar tales núcleos en un SoC a menudo requiere la inserción de lógica de interconexión adicional. Precisamente, los estándares para buses *on - chip* fueron desarrollados para evitar esta dificultad. Actualmente, existe una variedad de arquitecturas de bus disponibles internacionalmente: *Core-Connect* de IBM [64], AMBA de *ARM* [65], *Avalon* de Altera [66] y otros. Estas arquitecturas de bus están normalmente ligadas a un procesador, que puede ser *PowerPC, LEON, ARM, NIOS* u otro. De esta forma, los fabricantes proporcionan *cores* optimizados para trabajar con estas arquitecturas, requiriendo así una lógica extra mínima.

2.1.2 Topologías de buses.

Con respeto a la topología, una arquitectura de comunicación (bus) *on-chip* puede ser clasificada en los siguientes tipos:

Bus Compartido: este sistema de bus es el tipo más simple de topología de arquitectura de comunicación y el más común en varios SoC comerciales [67]. En este bus, varios maestros y esclavos pueden ser interconectados. El árbitro del bus, a partir de las solicitudes de las interfaces maestro, otorga el acceso a un único maestro. El mecanismo para otorgar dicho acceso es variable y puede o no ser especificado por el protocolo del bus. El incremento de la cantidad global de líneas en el bus limita su ancho de banda. Las ventajas de una arquitectura de bus compartido son: incluye una topología simple, extensible y de bajo costo en área; es una topología fácil de construir y en general es factible de implementar eficientemente. Las desventajas de esta arquitectura son: larga carga de datos por líneas del bus, mayor retraso en la transferencia de datos, alto consumo de energía, y limitado ancho de banda.

Bus Jerárquico: esta arquitectura consiste en varios buses compartidos interconectados por puentes formando una jerarquía. Los componentes del SoC son situados al nivel apropiado en la jerarquía según el nivel de rendimiento que estos requieran. Los componentes de bajo rendimiento son situados en buses de menor ancho de banda, la conexión de este bus al de alto rendimiento se concreta con un puente, de esta manera no se produce efecto de carga en los buses de mayor rendimiento. Ejemplos comerciales de tales arquitecturas son el bus AMBA y *CoreConnect*. Las transacciones a través del puente requieren que durante la transferencia ambos buses permanezcan inaccesibles para el resto de los componentes. Los buses jerárquicos ofrecen grandes mejoras de rendimiento con respecto a los buses compartidos debido a que: disminuyen la carga por bus; admiten un número potencial de transacciones a efectuarse en paralelo en

diferentes buses y múltiples transacciones independientes pueden efectuarse a través del puente de forma canalizada [68].

Anillo: son ampliamente usadas en numerosas aplicaciones, basadas en aplicaciones de anillo tales como procesadores de red, conmutadores ATM [68]. En un anillo cada nodo (maestro/esclavo) se comunica con un protocolo de traspaso de consigna.

2.2 Selección de la especificación de bus a utilizar.

Se ha optado por el bus Wishbone como estándar de interoperabilidad entre *cores*. El motivo fundamental de esta elección es que se trata de una especificación libre de patentes y derecho de autor, lo que hace que su uso sea gratuito. No obstante, la elección de Wishbone como arquitectura estándar tiene fuertes argumentos tecnológicos a favor y no sólo de índole comercial. Wishbone presenta dos objetivos fundamentales: simplicidad y flexibilidad.

La simplicidad se establece como la capacidad de conseguir elevadas tasas de datos con una complejidad de *hardware* mínima. A diferencia de otros estándares donde se define una jerarquía de varios protocolos, en Wishbone sólo existe uno y es de alta velocidad. La conexión de dispositivos de alta y baja velocidad queda determinada empleando dos buses Wishbone, uno para dispositivos de alta velocidad y otro para dispositivos de baja velocidad, de esta manera se evita dificultar la gestión del bus.

Desde el punto de vista de la flexibilidad, el estándar no define una estructura para el árbitro de manera que queda a elección del diseñador elegir la mejor opción según sea el caso. Por otro lado, el estándar permite utilizar diversas topologías de bus, punto a punto, flujo de datos, bus compartido y bus conmutador de barra. Adicionalmente admite configurar otros parámetros como: ancho del bus, etiquetas de datos personalizadas, direcciones, entre otros.

2.2.1 Bus Wishbone.

La arquitectura de interconexión para módulos IP WISHBONE (SoC) [60] es una metodología de diseño flexible, propuesta por *Wade Peterson*, director de la compañía *Silicore Corporation* y mantenida por la comunidad *OpenCores*. Esta arquitectura tiene como objetivo fundamental fomentar la reusabilidad de módulos IP, disminuyendo los problemas de integración de SoC y mejorando la portabilidad y fiabilidad del sistema.

La interconexión Wishbone es una propuesta de interfaz de uso general y define un estándar de intercambio de datos entre módulos IP, no especifica las características internas y funciones del

módulo que la implementa. Tres factores fundamentales sustentan esta arquitectura: (1) la necesidad de una solución satisfactoria y confiable de integración SoC; (2) la necesidad de una especificación de interfaz común para facilitar metodologías de diseño estructuradas en equipos de grandes proyectos y (3) la influencia de los sistemas de integración tradicionales como PCI y VMEbus.

Wishbone es análogo a un bus de microcomputadora en tres aspectos fundamentales, (1) proporciona una solución de integración flexible que puede ser fácilmente diseñada a la medida para una aplicación específica, (2) ofrece variedad de ciclos de bus y tamaños de palabra de datos y (3) permite el empleo de módulos IP de diferentes fuentes.

La arquitectura de interconexión Wishbone facilita el diseño de módulos IP y de Sistemas en un *Chip*. Para ello se apoya en un protocolo estándar para el intercambio de datos.

Algunas características generales de esta tecnología son:

- Soporta metodologías de diseño estructurado usadas en grandes proyectos.
- Conjunto completo de protocolos de transferencia de datos por el bus:
  - Ciclo de lectura/escritura
  - Ciclo de trasferencia de bloques de datos
  - Ciclo de lectura/escritura unificado (RMW)
- Soporta ordenamientos *Big Endian* y *Little Endian*.
- Protocolo de *handshaking* que permite al módulo IP regular la velocidad de transferencia de datos.
- Soporta diversos tipos de terminación de ciclo: normal, reintento y error.
- Esquema de decodificación parcial.
- Etiquetas definidas por el usuario.
- Arquitectura Maestro/Esclavo.
- Capacidad de multiprocesamiento (soporta varios maestros).
- Soporta varias topologías de interconexión:
  - Punto a Punto
  - Bus compartido
  - *Switch* de interconexiones
  - Flujo de datos
- Diseño sincrónico que asegura portabilidad, simplicidad y facilidad de uso.

- Independiente de la tecnología del *hardware* (FPGA, ASIC, etc.)

2.2.2 Especificación de la interfaz Wishbone.

La especificación Wishbone define cuatro módulos funcionales (o macros) de la arquitectura del bus. El conjunto de estos módulos y las reglas de interacción entre ellos define el bus Wishbone. Los módulos funcionales son:

***SYSCON*:** Define el comportamiento de dos señales fundamentales en Wishbone, la señal de reinicio (RST) y la señal de reloj (CLK). Wishbone es un bus sincrónico y la señal de re-inicialización es una señal sincrónica, es por ello que se define explícitamente este módulo y se le presta especial atención a la señal de reloj. El módulo define dos salidas RST_O y CLK_O que son las entradas al resto del sistema.

***MASTER*:** Define uno de los componentes de la arquitectura Maestro/Esclavo de Wishbone. Toda interfaz *MASTER* es aquella capaz de iniciar por ella misma un ciclo de bus. Define un conjunto de señales para esta interfaz y especifica su operación.

***SLAVE*:** Define el otro componente de la arquitectura Maestro/Esclavo de Wishbone. Toda interfaz *SLAVE* es aquella capaz de aceptar un ciclo de bus iniciado por un *MASTER*. Define el conjunto de señales para este tipo de interfaz y especifica su operación.

***INTERCONN*:** Es la lógica asociada a la interconexión de las tres interfaces anteriores para conformar el bus Wishbone. Aquí se define la topología del bus y los mecanismos de arbitraje.

2.2.3 Módulo *INTERCONN* del bus Wishbone.

Como ya se definió más arriba el módulo *INTERCONN* especifica la interconexión entre las diferentes interfaces *MASTER*, *SLAVE* y el módulo *SYSCON*. De esta manera, la topología del bus y los tipos de arbitraje quedan definidos por este módulo. Las topologías soportadas por el estándar son las siguientes.

- Punto a Punto
- Bus compartido
- Bus conmutador de barra (bus conmutado)
- Flujo de datos

La topología punto a punto es el tipo de conexión más sencillo y conecta directamente una interfaz maestro con una interfaz esclavo, en la Figura 2.1 se muestra un ejemplo de este tipo de conexión.

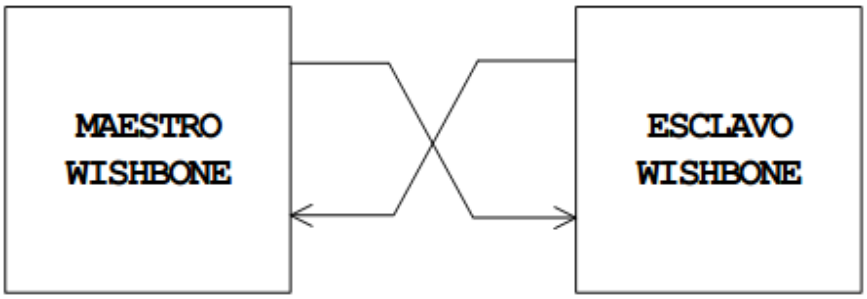


Fig. 2.1 *Topología de interconexión punto a punto.*

En la topología de bus compartido uno o varios *MASTER* se conectan a uno o varios *SLAVE* mediante un canal o bus compartido, ver Figura 2.2.

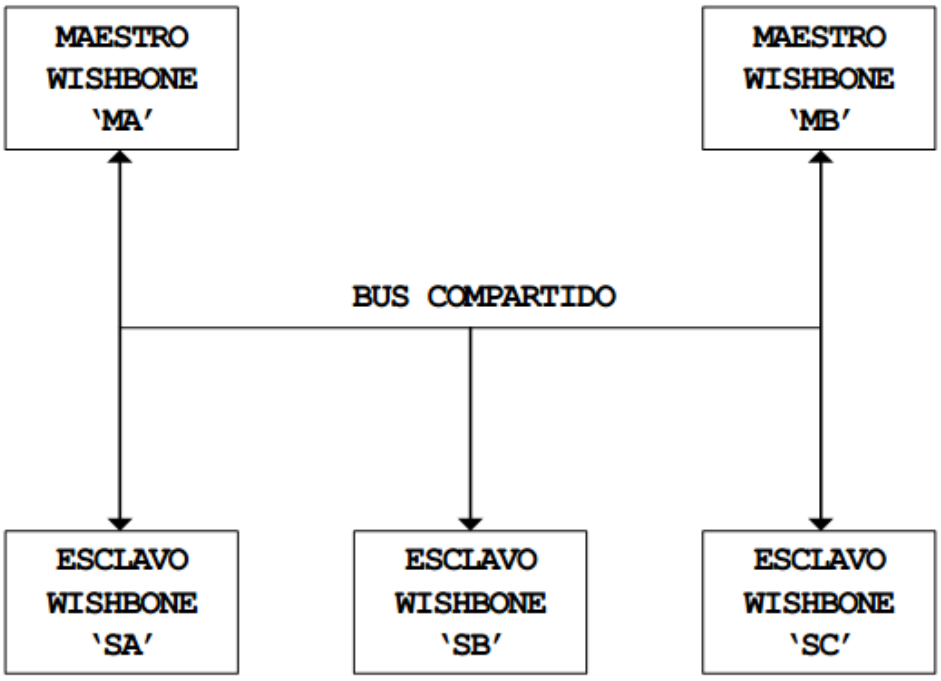


Fig. 2.2 *Topología de interconexión de bus compartido.*

El bus conmutador de barra o bus conmutado es una topología donde el bus se divide en canales. Cada canal puede ser un bus compartido con uno o varios *SLAVE* conectados. Cada interfaz *MASTER* arbitra por el control de un canal y el árbitro entrega el control de los canales en base a algún algoritmo. Este tipo de topología aunque requiere de mayor cantidad de recursos lógicos que

las anteriores, también multiplica el ancho de banda equivalente del sistema de bus. Con lo que se alcanzan rendimientos superiores al resto de las topologías. En la Figura 2.3 se muestra un bus con estas características.

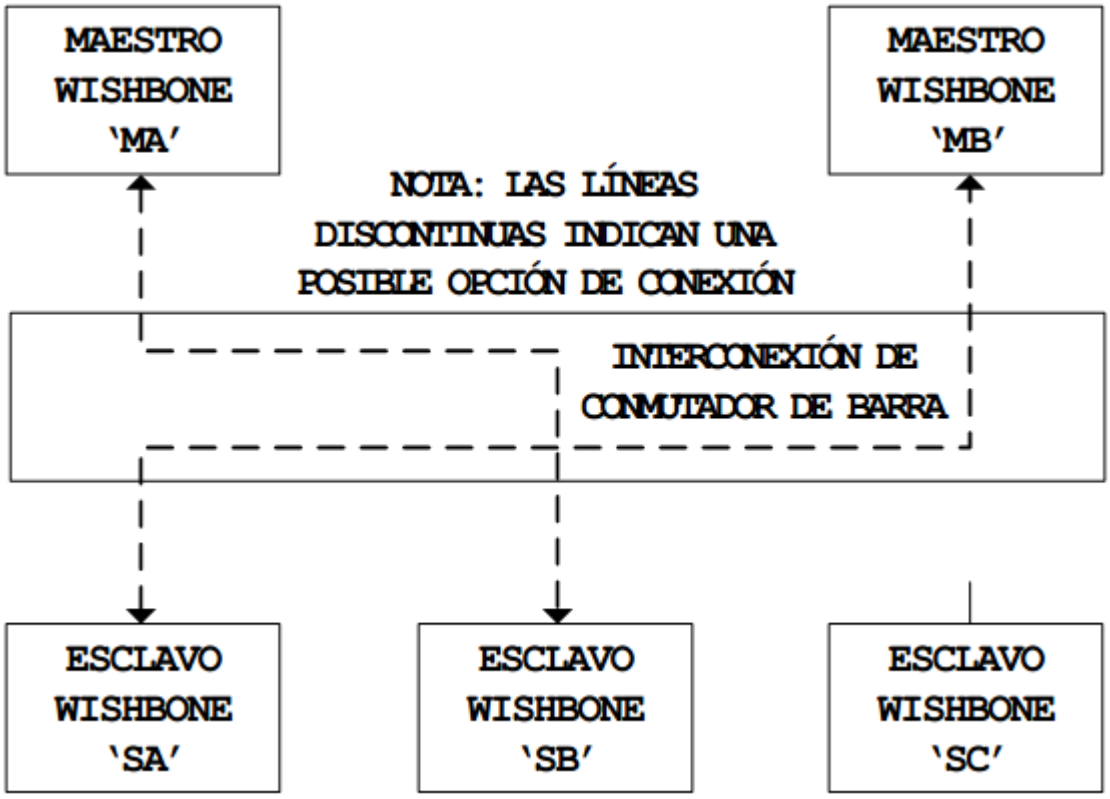


Fig. 2.3 *Topología de interconexión de bus conmutador de barra o bus conmutado*

Por último la topología de bus flujo de datos requiere que cada módulo en el interior del flujo de datos incorpore una interfaz maestro y una interfaz esclavo. De esta manera se interconectan de igual forma que en una conexión punto a punto encadenada. Esta topología es empleada a menudo en sistemas de procesamiento de señales. En la Figura 2.4 se muestra un ejemplo de esta interconexión.

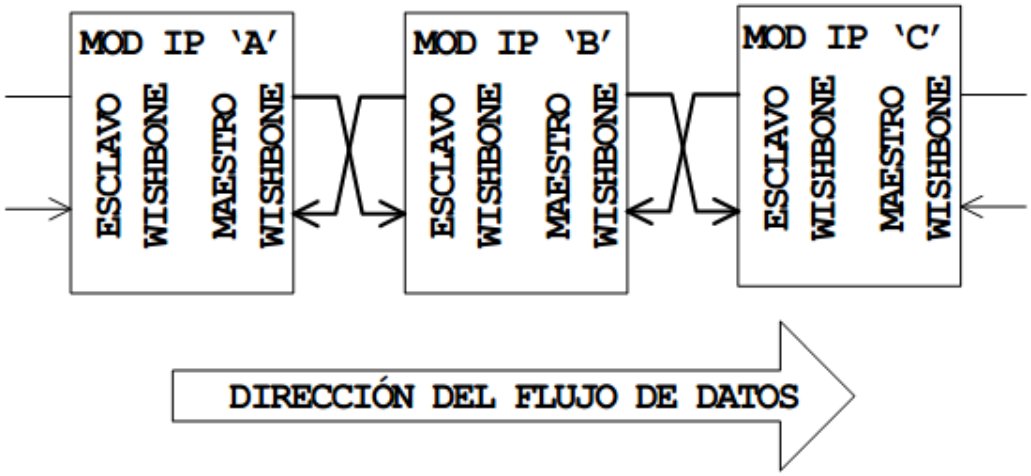


Fig. 2.4 *Topología de interconexión de flujo de datos.*

2.3 Arquitectura propuesta para el Sistema en un *Chip*.

La arquitectura define dos buses Wishbone de diferente topología, Figura 2.5. El bus superior (A) es un bus compartido de un único maestro y es donde se conectarán los periféricos esclavos de baja velocidad (UART, VGA TXT, PIC, etc.). Para este bus se ha empleado Terminación de Ciclo Sincrónica (TCS), pero, puede utilizarse igualmente Terminación de Ciclo Asincrónica (TCA).

El bus inferior (B) es multi-maestro y de alta velocidad pues a él se conectarán los dispositivos maestros y esclavos que requieran altas tasas de transferencia (DMA, Memoria, Capa de Enlace *IEEE-1394*, interfaz IDE, etc.). La topología empleada para el bus B es de bus conmutador de barra e implementa la Terminación de Ciclo Sincrónica Avanzada (TCSA). Para ambos buses se ha empleado el generador Wishbone disponible en *OpenCores* [69].

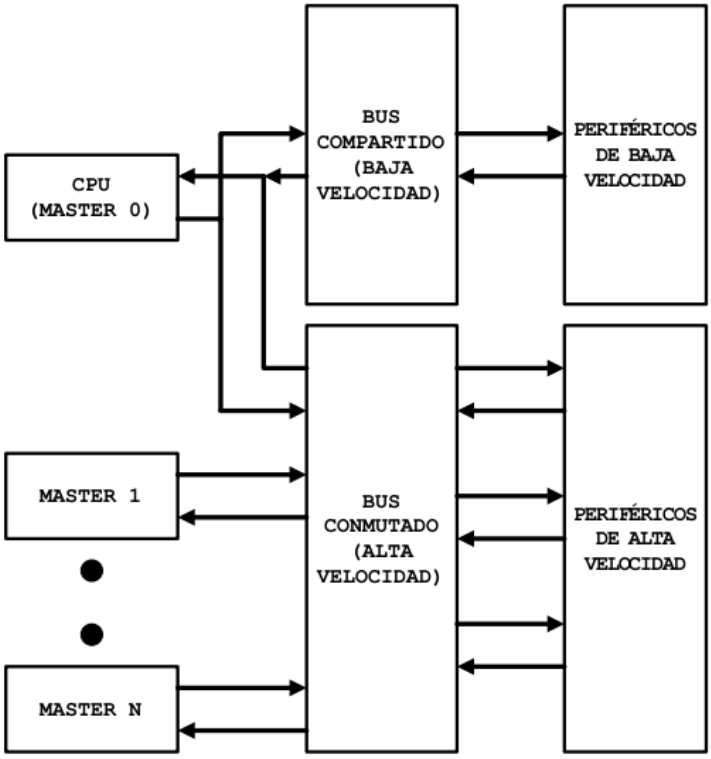


Fig. 2.5 *Diagrama simplificado de la arquitectura del SoC.*

Hay que decir que este generador solo devuelve código VHDL, y es por ello que se emplea este lenguaje de descripción de *hardware* para integrar todo el sistema. Otro elemento a destacar es que el programa genera para la resolución de los buses sólo lógica cableada y es necesario cambiar todo a multiplexores. Este cambio se realiza sin ningún inconveniente.

En la Figura 2.5 se muestra cómo el procesador accede directamente a ambos buses y se comporta como maestro principal. El resto de los dispositivos maestros sólo pueden acceder directamente a los esclavos conectados en el bus B. Esto implica que para la comunicación entre los dos buses es necesaria la inclusión de un puente (*bridge*) que soporte señales Wishbone.

2.4 Realización del puente entre buses.

Para permitir la comunicación de dispositivos en buses distintos se ha incluido en el diseño un módulo IP especial disponible en *OpenCores* denominado Wishbone DMA/*Bridge* IP *Core* [70], Figura 2.6. Este módulo es de libre distribución, ha sido desarrollado por Rudolf Usselmann, está descrito en Verilog y disponible en *OpenCores*.

2.4.1 Wishbone DMA/*Bridge* IP *Core*.

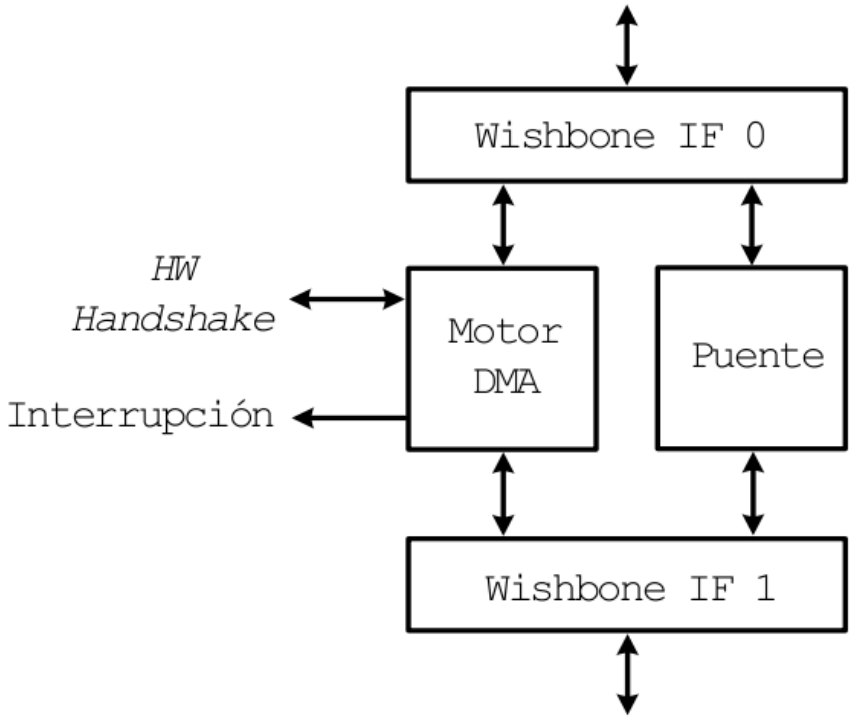


Fig. 2.6 *Arquitectura del módulo DMA.*

El módulo Wishbone DMA/*Bridge* IP *Core* provee transferencias DMA entre dos interfaces Wishbone diferentes o desde una misma interfaz. Puede además funcionar como puente (*bridge*). Soporta hasta 31 canales DMA, admite 2, 4 u 8 niveles de prioridad y permite los modos: normal, lista de descriptores, *buffers* circulares y *buffers* FIFO. Realiza transferencias iniciadas por *software* o por *hardware*.

2.4.1.1 Operación en modo puente.

En este modo el módulo DMA actúa como puente aunque no se adiciona ninguna funcionalidad al tráfico en el bus. La conexión se realiza de manera combinacional, las puertas de enlace (entrada) hacia las otras interfaces Wishbone son sus interfaces esclavos y la salida es por las interfaces maestros opuestas. De manera que un dispositivo maestro de una interfaz pude iniciar ciclos de bus en otra interfaz, si está conectada al DMA/*Bridge*, Figura 2.7. De esta forma, un ciclo iniciado por algún maestro conectado a la interfaz esclavo 0 que direccione el puente se ve repetido en la interfaz maestro 1. Lo mismo ocurre si se inicia un ciclo de bus por la interfaz esclavo 1, este se verá reflejado en la interfaz maestro 0. Esta funcionalidad es la que se aprovecha para diseñar el puente en la arquitectura propuesta.

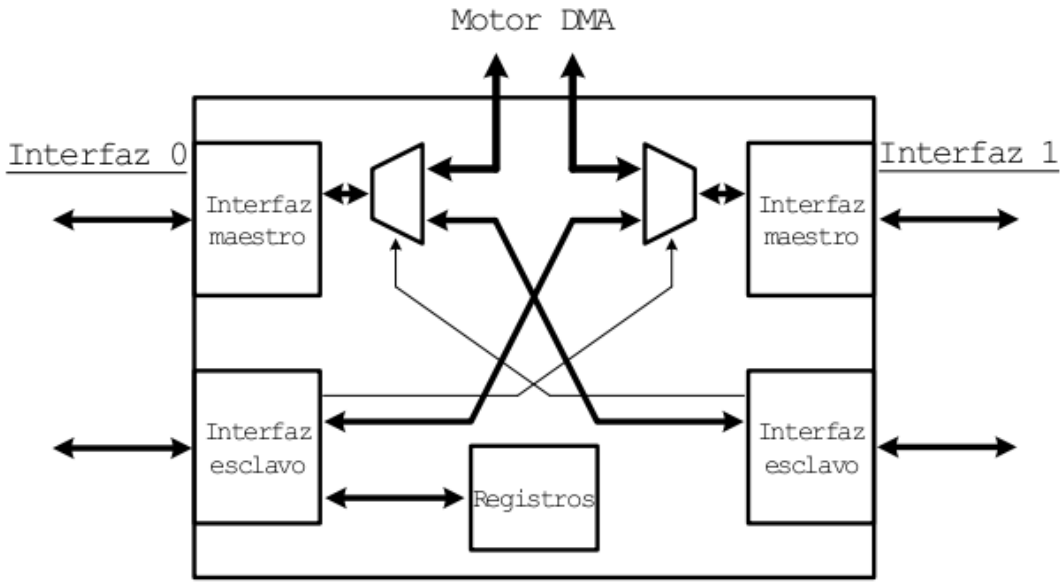


Fig. 2.7 *Lógica de realización del puente.*

2.4.2 Comunicación entre buses.

La comunicación entre buses está soportada por el módulo DMA y el puente arbitra por el control del bus en ambas interfaces dependiendo del periférico al cual se debe acceder. El acceso del procesador a los periféricos de baja velocidad se realiza a través del puente establecido por el módulo DMA entre las interfaces DMA esclavo 0 y DMA maestro 1. Figura 2.8. De esta forma un acceso del procesador o bien es al bus de alta velocidad (Wishbone B) o se ve reflejado en el bus de baja velocidad por medio de la interfaz maestro1. Como se ve de la Figura 2.8 es necesario que la CPU sea compatible con el bus, por lo que se requiere modificar el procesador MIPS Lite para que genere ciclos Wishbone.

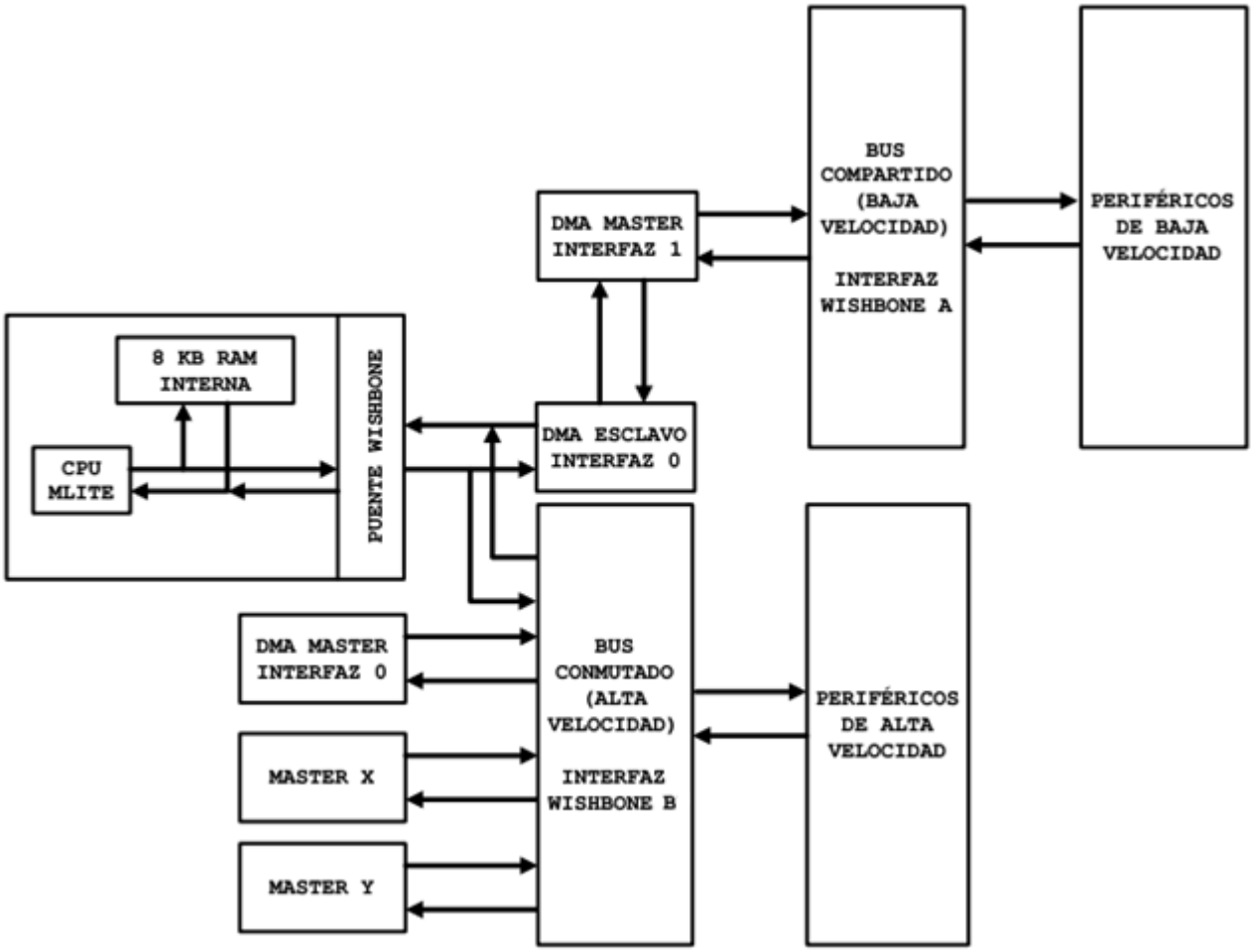


Fig. 2.8 *SoC empleando el módulo Wishbone DMA/Bridge.*

2.5 Procesador MIPS Lite.

Para convertir el procesador MIPS Lite en una arquitectura Wishbone, se definen varios pasos. Primero es necesario garantizar máxima frecuencia, para ello se ha configurado el procesador MIPS Lite en tres (3) etapas de *pipeline* (se puede configurar en 2 y 3 etapas de *pipeline*). Para garantizar un mayor rendimiento se ha establecido este parámetro en tres (3) a pesar de que también implica un mayor consumo en recursos lógicos (área). Otro aspecto fundamental que el bus del procesador no es compatible con Wishbone, para compatibilizar se ha construido un *wrapper- bridge* (puente) que convierte las señales del bus de MIPS Lite a equivalentes Wishbone. Con el fin de aprovechar las señales de acceso a RAM sincrónica disponibles en Plasma se han conservado los 8 kB de memoria del sistema original. De esta manera al núcleo del procesador se han añadido los siguientes elementos: CPU MIPS Lite; 8 kB de memoria embebida SRAM y *Wrapper- bridge* Wishbone.

El puente está diseñado de forma que el acceso a la memoria embebida no produce ciclo de bus al exterior del módulo. De esta manera, direcciones menores en valor absoluto a 0x10000000 no generarán ciclo de bus fuera del módulo del procesador.

Para generar las señales Wishbone no existe ningún problema excepto por la señal SEL_O de Wishbone que no tiene equivalente en MIPS Lite. Con el fin de generar estas señales directamente se ha modificado el código del controlador de memoria (Mem_ctrl) de MIPS Lite, para generar SEL_O desde el interior del procesador. Esta señal tiene la misma temporización que byte_we de MIPS Lite pero valores distintos. Para garantizar una buena sincronización se introducen estados de espera con la señal mem_pause. La señal ACK_I de Wishbone y lógica adicional definen el valor de mem_pause. Las señales CYC_O y STB_O son generadas en el puente.

2.6 Dispositivos Periféricos.

Para valorar la efectividad de este esquema se ha creado un SoC de prueba que incluye:

- Dispositivos maestros de alta velocidad
- Dispositivos esclavos de alta velocidad
- Dispositivos esclavos de baja velocidad

De todos los elementos disponibles en la tarjeta de desarrollo se emplearán: la memoria Micron® SRAM, el puerto VGA, el puerto RS232 y los dispositivos de E/S.

2.6.1 Módulo controlador de memoria SRAM.

Para dar soporte a la memoria disponible en la tarjeta FPGA Spartan-3, se diseñó un controlador que traduce las señales del bus Wishbone a señales de entrada de la memoria SRAM.

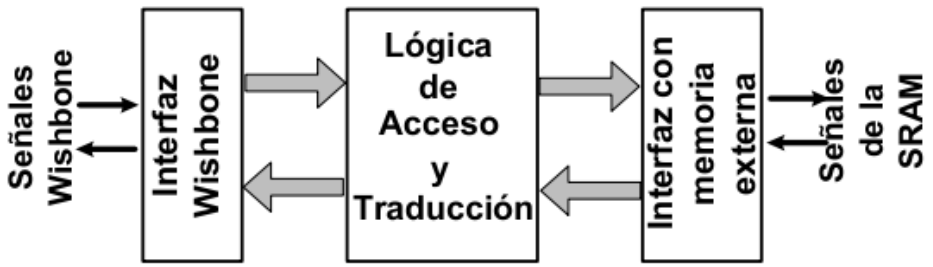


Fig. 2.9 *Controlador de memoria SRAM.*

Dado que el bus de la memoria es de 32 bits al igual que el bus Wishbone el mecanismo de traducción de las lecturas y escrituras se vuelve simple tal como se muestra en la Figura 2.9.

El controlador está implementado en lenguaje VHDL y está disponible en el sitio web de *OpenCores*. Su arquitectura externa se muestra en la Figura 2.10.

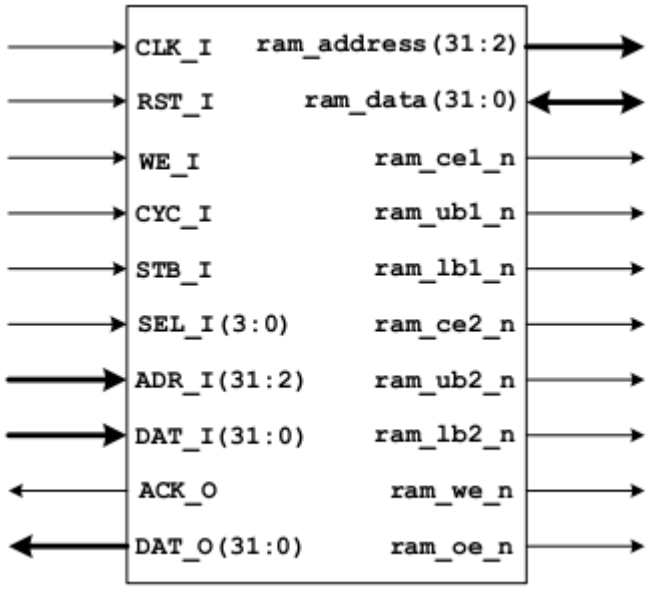


Fig. 2.10 *Arquitectura externa del controlador de memoria SRAM.*

El bus Wishbone se compone de las señales de entrada ***ADR_I(31:2)*** para el direccionamiento de los datos, la señal de datos ***DAT_I(31:0)***; a través de la señal ***CYC_I*** se solicita el canal y ***STB_I*** indica el inicio de ciclo, ***WE_I*** en nivel alto activa la escritura de datos y en nivel bajo la lectura, la señal ***SEL_I(3:0)*** es de 4 bits activos en nivel alto y cada bit representa la habilitación en escritura de un banco de memoria.

La señal de reinicio ***(RST_I)*** es sincrónica; cuando el reinicio está activo en nivel alto las salidas de dirección y datos ***ram_address(31:2)***, ***ram_data(31:0)*** y ***DAT_O(31:0)*** están en '0'. Si el reinicio está desactivado el primer flanco de reloj carga en ***ram_address(31:2)*** el valor de ***ADR_I(31:2)*** y el valor de ***DAT_I(31:0)*** se carga en el buffer de datos interno. Cuando las señales ***STB_I*** y ***CYC_I*** están activas se habilitan los bloques de memoria a través de las señales activas en nivel bajo ***ram_ce1_n*** y ***ram_ce2_n***, quedando los bloques de memoria listos para las operaciones de lectura y escritura de datos. La señal ***WE_I*** en nivel bajo activa la lectura de datos mediante la activación de la señal de salida ***ram_oe_n*** (nivel bajo) y en nivel alto la escritura de datos

mediante la activación de la señal de salida ***ram_we_n*** (nivel bajo). La selección de los bancos de memoria para escritura se logra negando los bits de la señal ***SEL_I(3:0)***, los dos bits menos significativos activan las señales ***ram_ub2_n*** y ***ram_lb2_n*** y los más significativos las señales ***ram_ub1_n*** y ***ram_lb1_n***. La señal de salida ***ACK_O*** se activa (nivel alto) en el segundo ciclo de reloj luego de que se ha realizado una operación indicando que se ha concluido satisfactoriamente.

2.6.2 Módulo UART IP *Core*.

Después de analizar las posibilidades de la UART que acompaña al SoC Plasma con motivo de la adaptación del sistema para una aplicación de comunicaciones concreta, se observan algunas limitaciones importantes. La UART sintetizable incluida en el SoC Plasma solamente dispone de las líneas de transmisión y recepción, además de fuertes limitaciones de configuración ya que solo puede operar a 57600 baudios. Para dotar al sistema de una UART capaz de proporcionar una funcionalidad completa, incluyendo, por ejemplo, las líneas de control de modem, se decide utilizar un modelo sintetizable, disponible en *OpenCores*: el módulo **UART IP *Core*** [71] compatible con el *chip* NS 16550, frecuentemente utilizado en la industria y particularmente en las computadoras personales. Este módulo, ver Figura 2.11, está descrito en Verilog y es de la autoría de Jacob Gorban. Las características principales de este dispositivo son:

- Interfaz Wishbone en modo 32 bits u 8 bits de datos, configurable
- Operación en modo FIFO
- Compatible con los registros y niveles del *chip* NS 16550
- Interfaz de depuración en el modo de 32 bits de datos
- Provee señales de control para Modem

Este módulo UART es muy similar en operación al UART estándar 16550 con la excepción de que este solo soporta el modo FIFO. Puede operar en los modos bus de datos 8 bits o 32 bits.

El modo 32 bits es completamente compatible con Wishbone y la interfaz del módulo emplea la señal de SEL_I para recibir y entregar adecuadamente los 8 bits de datos en el bus de 32 bits, con un bus de direcciones ADR_I de 5 bits. En el modo de 8 bits se usan solo 3 bits de direcciones.

Adicionalmente en el modo 32 bits está disponible en el módulo una interfaz de depuración. Esta interfaz tiene 2 registros de 32 bits que contienen los valores de algunas de las señales internas del UART, estas señales pueden ser leídas sin afectar el funcionamiento del módulo.

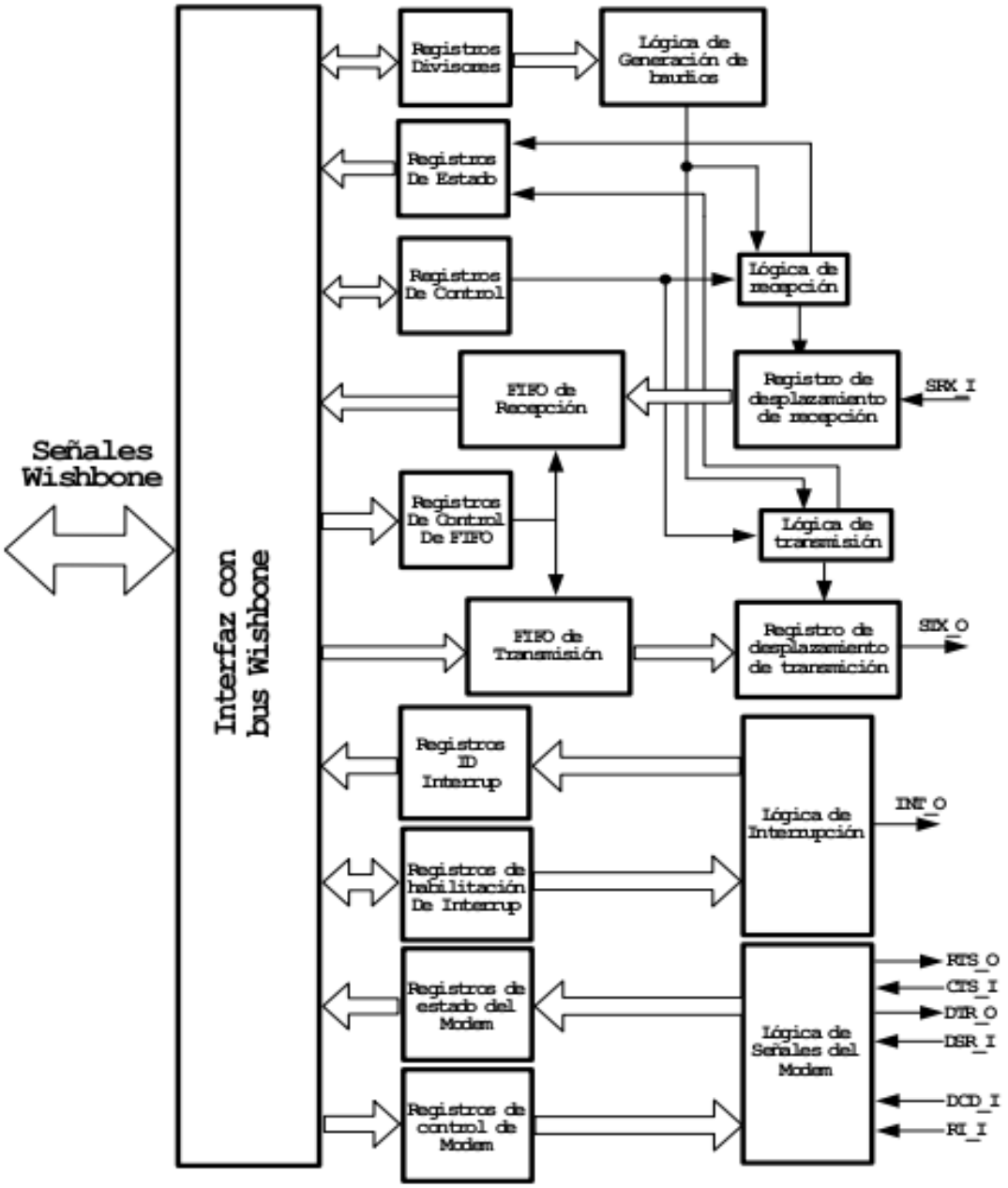


Fig. 2.11 *Diagrama en bloques del UART.*

Para un funcionamiento correcto se deben cumplir los siguientes pasos de configuración:

- Establecer un '1' en el bit 7 del Registro de Control de Línea (LCR) para tener acceso al divisor.
- Escribir el valor en el divisor. Primero el MSB y luego el LSB.
- Establecer un '0' en el bit 7 del Registro de Control de Línea (LCR) para deshabilitar el acceso al divisor.

- Configurar el Registro de Control de FIFO (FCR) con la cantidad de bytes recibidos antes de generar una interrupción para el sistema.
- Habilitar las interrupciones deseadas en el Registro de Habilitación de Interrupciones (IER).

2.6.3 Módulo Controlador Programable de Interrupciones (PIC).

Este dispositivo es esencial para todo el sistema, dado que las tareas como la recepción de datos vía *IEEE-1394* (requieren tiempo crítico fijo), la ocurrencia de eventos en la capa de enlace y otras tareas no pueden ser tratadas por encuesta. Es así que el controlador de interrupciones permite la gestión de numerosas fuentes de interrupción (hasta 8). Determinado por las características del tipo de dispositivo y el nivel de prioridad que tengan estos para el sistema. De tal manera que deben ser atendidas mediantes interrupciones, Figura 2.12. El dispositivo ha sido diseñado en Verilog por Richard Herveille [72], está disponible en *OpenCores*.

Características del controlador:

- El número de interrupciones depende del tamaño del bus, 8 fuentes de interrupción corresponden a un bus de datos de 8 bits
- Todos los dispositivos conectados a él comparten el mismo nivel de prioridad para la CPU.
- Provee una interfaz Wishbone para configuración y control

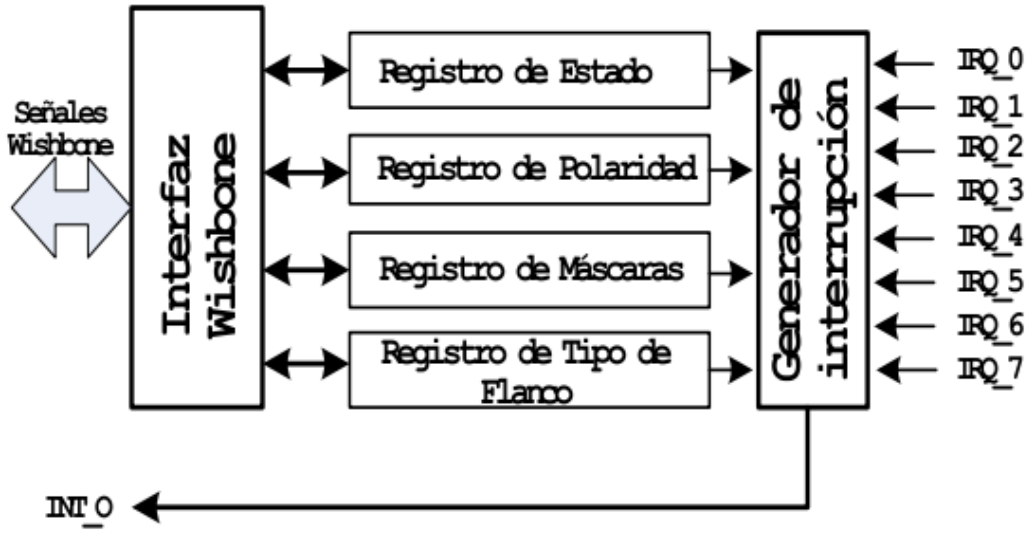


Fig. 2.12 *Controlador Programable de Interrupciones.*

2.6.4 Módulo controlador de visualizadores 7 segmentos.

El sistema incorpora un módulo controlador para 4 visualizadores 7 segmentos (Figura 2.13). La visualización se realiza de forma multiplexada lo que permite aprovechar los visualizadores disponibles en la tarjeta de desarrollo. Internamente el módulo está diseñado con un circuito de temporización (compuesto por dos contadores) y un camino de datos que incluye: un registro, un multiplexor y un decodificador BCD – 7 segmentos. La interfaz Wishbone permite la conexión directa con un bus que cumpla con esta especificación. Para visualizar un dato se envía a la dirección del dispositivo un código de 16 bits en formato BCD.

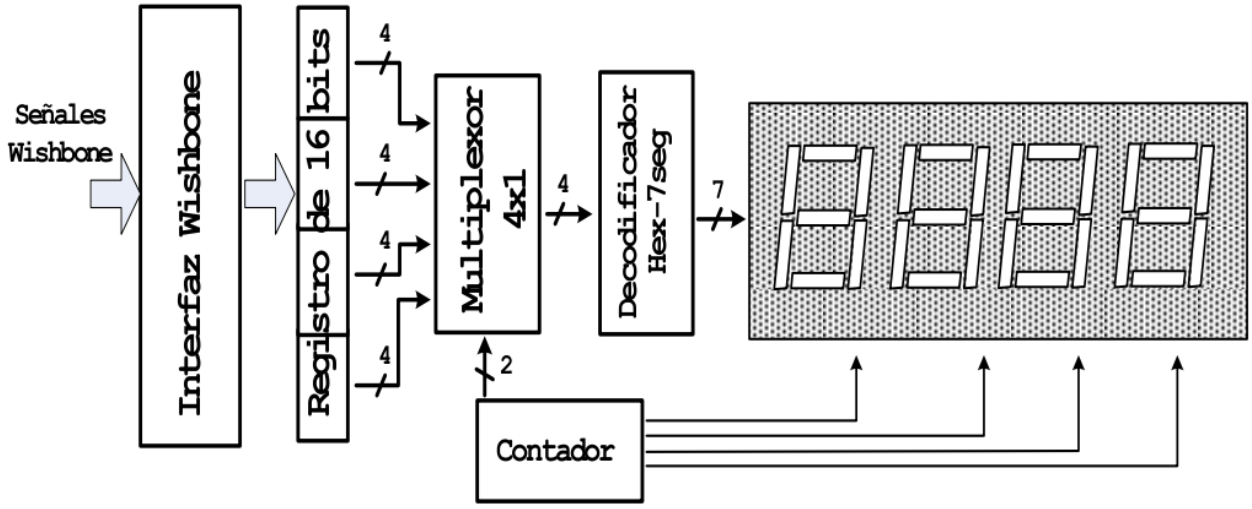


Fig. 2.13 *Controlador de visualizadores 7 segmentos.*

Desde el punto de vista de la interfaz con el bus el dispositivo es de propósito general de sólo escritura. Se puede utilizar con propósitos de depuración de *software*, por ejemplo, para comprobar la ejecución de un número de instrucciones o para determinar el punto en el cual se encuentra el programa que se está ejecutando dentro del SoC. Sin embargo, siempre queda a disposición del diseñador del *software* la función y uso del módulo.

2.6.5 Módulo Adaptador de *Display* Monocromático.

El módulo es un Adaptador de *Display* VGA Monocromático en modo texto y está disponible en *OpenCores* [73]. Las características principales del módulo adaptador son:

- Resolución de 80x40 caracteres y 640x480 píxeles a 60 Hz por lo que el núcleo necesita una señal de reloj de 25MHz.
- Monocromático, el color de salida es seleccionable entre 8 colores diferentes.

- Códigos ASCII de 8 bits.
- El búfer de texto de vídeo es una memoria externa de 80 × 40 = 3200 bytes.
- La memoria ROM que almacena la fuente del mapa de bits de ancho fijo también es externa.
- Cursor de *hardware*, con dos formas diferentes y control de activar / desactivar.
- Interfaz simple de I/O. Posee tres registros para posición y control del cursor de *hardware*.
- El diseño es independiente del proveedor.

La arquitectura externa del adaptador de *display* monocromático, Figura 2.14, está compuesta de cuatro memorias RAM doble puerto de 4 kB donde se almacenan los *buffers* de texto y de fuente, tres registros para los cursores de *hardware*, la señal de control de periférico (***octl [7:0]***) y el módulo adaptador el cual fue programado en lenguaje VHDL.

El módulo original de *OpenCores* fue adaptado al sistema Plasma-Wishbone y se diseñó para utilizar como *buffers* de texto y fuente las memorias doble puerto de 2 kB presentes en la tarjeta FPGA Spartan - 3. Estas memorias cuentan con dos puertos independientes lo que permite realizar dos operaciones simultáneas ya sean de lectura, escritura o la combinación de ambas. El puerto B de estas memorias es accedido por el procesador MIPS Lite a través de las señales Wishbone de entrada y salida de datos ***DAT_I [31:0]*** y ***DAT_O [31:0]***.

Las señales de doce bits ***Text_A [11:0]*** y ***Font_A [11:0]*** se usan para acceder a las direcciones de los bloques de memoria doble puerto donde están ubicados los datos de texto y fuente que son entregados al módulo adaptador a través de las señales ***Text_D [7:0]*** y ***Font_D [7:0]***, de esta manera el adaptador al recibir los datos decide a través de los bits (2:0) de la señal de control ***octl [7:0]*** la información de color en las salidas ***R***, ***G*** y ***B***.

Las señales ***ocrx [7:0]*** y ***ocry [7:0]*** provienen de dos registros y brindan la posición en columna y fila del cursor de *hardware*, los pulsos de borrado horizontal y vertical, así como la información de sincronismo horizontal y vertical mediante las señales de salida ***hsync*** y ***vsync***.

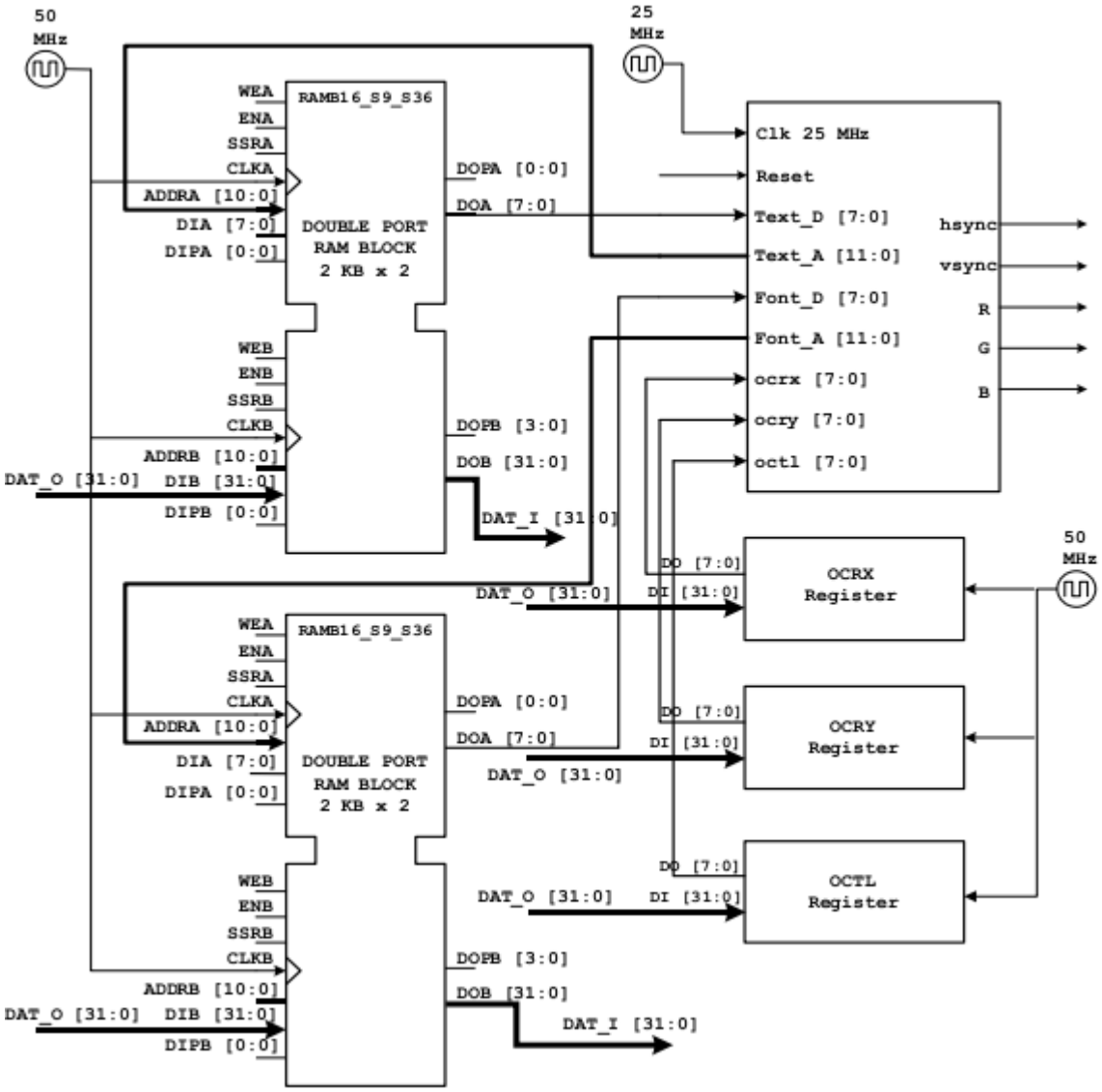


Fig. 2.14 *Arquitectura externa del Adaptador de Display Monocromático.*

2.6.6 Módulo controlador de la interfaz IDE.

Dado que se manejará una interfaz IDE, para dotar al sistema de alta confiabilidad y eficiencia se decide utilizar un modelo sintetizable, disponible en *OpenCores*: el módulo **ATA/ATAPI-5 *Core*** [74] compatible con Wishbone, cuyo autor es Richard Herveille. En otras palabras OCIDEC (*OpenCores* IDE Controller) es una implementación de interfaz ATA/ATAPI-5 compatible con WISHBONE rev.B2.

Tres versiones diferentes del *core* están actualmente disponibles. Todas las versiones del *software* y sus funciones son compatibles hacia atrás. El *software* puede detectar qué versión está

implementada leyendo el identificador del dispositivo y el número de revisión del registro de estado, haciendo así posible manejar todos los *cores* con un solo controlador de dispositivo. Esto le permite a un **diseñador** / **sistema** la capacidad de buscar un compromiso ente la **complejidad** / **uso de recurso** para buscar mejoras en **rendimiento** /**características**. La siguiente tabla muestra las características específicas de cada *core*.

Tabla 2.1. Características específicas de cada versión de controlador OCIDEC.

	OCIDEC-1	**OCIDEC-2**	**OCIDEC-3**
Características	*Core* más pequeño Sólo sincronización compatible PIO	*Core* Pequeño Soporte de transferencia PIO solamente Puerto de datos PIO de sincronización rápida por dispositivo	Soporte de transferencia PIO y DMA Configuración PIO por dispositivo Configuración DMA por dispositivo
Uso previsto	Sistemas individuales CompactFlash / PCCard Sistema que requiere capacidades ATA simples	Sistemas Duales CompactFlash / PCCard Sistema que requiere capacidades ATA rápidas	Interfaz de Disco duro / CDROM Sistema que requiere capacidades ATA
Número de puertas	Aprox. 4kgates	Aprox.4.6kgates	Aprox.14kgates

Características generales:

- *Core* compatible con ATA / ATAPI Rev.5.
- *Cores* de retrocompatibilidad de *software* y funciones.
- PIO común compatible, con ajustes de sincronización para todos los dispositivos conectados.
- Puerto de datos *Fast* PIO con ajustes de sincronización para dispositivo conectado.
- Ajustes de sincronización *Singleword / Multiword* para dispositivo conectado.
- Escritura PIO *PingPong* mejorada.

- *Big Endian* automático frente a la conversión *Little Endian.*
- *Buffers* DMA lectura/ escritura.
- *OpenCores* WISHBONE motor DMA compatible.
- Compatible con WISHBONE rev.B2.
- Interfaz con *host* de 32 bits, el reloj de entrada opera en una amplia gama de frecuencias, diseño síncrono estático y totalmente sintetizable.

Las siguientes tablas muestran el conjunto de señales empleadas por la interfaz.

Tabla 2.2. Señales de interconexión WISHBONE.

Puerto	**Tamaño (bits)**	**Dirección**	**Descripción**	**Versión**
wb_clk_i	1	Entrada	Entrada reloj maestro	Todos
wb_rst_i	1	Entrada	*Reset* sincrónico activo en alto	Todos
wb_adr_i	5	Entrada	*Bits* bajos de dirección	Todos
wb_dat_i	32	Entrada	Datos hacia el c*ore*	Todos
wb_dat_o	32	Salida	Datos desde el *core*	Todos
wb_sel_i	4	Entrada	*Byte* de selección de señales	Todos
wb_we_i	1	Entrada	Entrada de habilitación de escritura	Todos
wb_stb_i	1	Entrada	Señal *Strobe* / entrada de selección del *core*	Todos
wb_cyc_i	1	Entrada	Entrada ciclo de bus válida	Todos
wb_ack_o	1	Salida	Salida de ciclo de bus *acknowledge*	Todos
wb_rty_o	1	Salida	Salida de ciclo de bus de reintento	Todos
wb_err_o	1	Salida	Salida ciclo de bus de error	Todos
wb_inta_o	1	Salida	Salida de señal de petición de interrupción	Todos

Tabla 2.3. Señales de la interfaz ATA.

Puerto	**Tamaño (bits)**	**Dirección**	**Descripción**	**Versión**
resetn_pad_o	1	Salida	Reinicio de *hardware* IDE	Todos
dd_pad_i	16	Entrada	Datos del dispositivo (desde el dispositivo ATA)	Todos
dd_pad_o	16	Salida	Datos del dispositivo (hacia el dispositivo ATA)	Todos
dd_padoe_o	1	Salida	DD habilitación de la salida	Todos
da_pad_o	3	Salida	Dirección del dispositivo	Todos
cs0n_pad_o	1	Salida	*Chip Select0*	Todos
cs1n_pad_o	1	Salida	*Chip Select1*	Todos
dmarq_pad_i	1	Entrada	Solicitud DMA	OCIDEC3
dmackn_pad_o	1	Salida	DMA *acknowledge*	OCIDEC3
diorn_pad_o	1	Salida	Lectura en dispositivo de IO	Todos
diown_pad_o	1	Salida	Escritura en dispositivo de IO	Todos
iordy_pad_i	1	Entrada	Canal de IO listo	Todos
intrq_pad_i	1	Entrada	Interrupción de dispositivo	Todos

Tabla 2.4. Otras señales.

Puerto	**Tamaño (bits)**	**Dirección**	**Descripción**	**Versión**
arst_i	1	Entrada	*Reset* asíncrono	Todos
DMA_req	1	Salida	Solicitud de DMA a motor DMA externo	OCIDEC3
DMA_ack	1	Entrada	DMA *acknowledge* del motor DMA externo	OCIDEC3

Por las razones expuestas anteriormente para el diseño que se propone se empleará la versión tres del controlador IDE. OCIDEC3 que soporta transferencias DMA y su uso previsto es para interfaz de disco duro / CDROM y sistemas que requieran capacidades ATA.

2.7 Propuesta para el Sistema en un *Chip*.

En la Figura 2.15 se muestra el SoC propuesto. En la memoria interna del módulo del procesador, al igual que en Plasma, se ha incluido un pequeño cargador de programas (*boot loader*), en el arranque el UART se configura para que funcione a 115,200 baud.

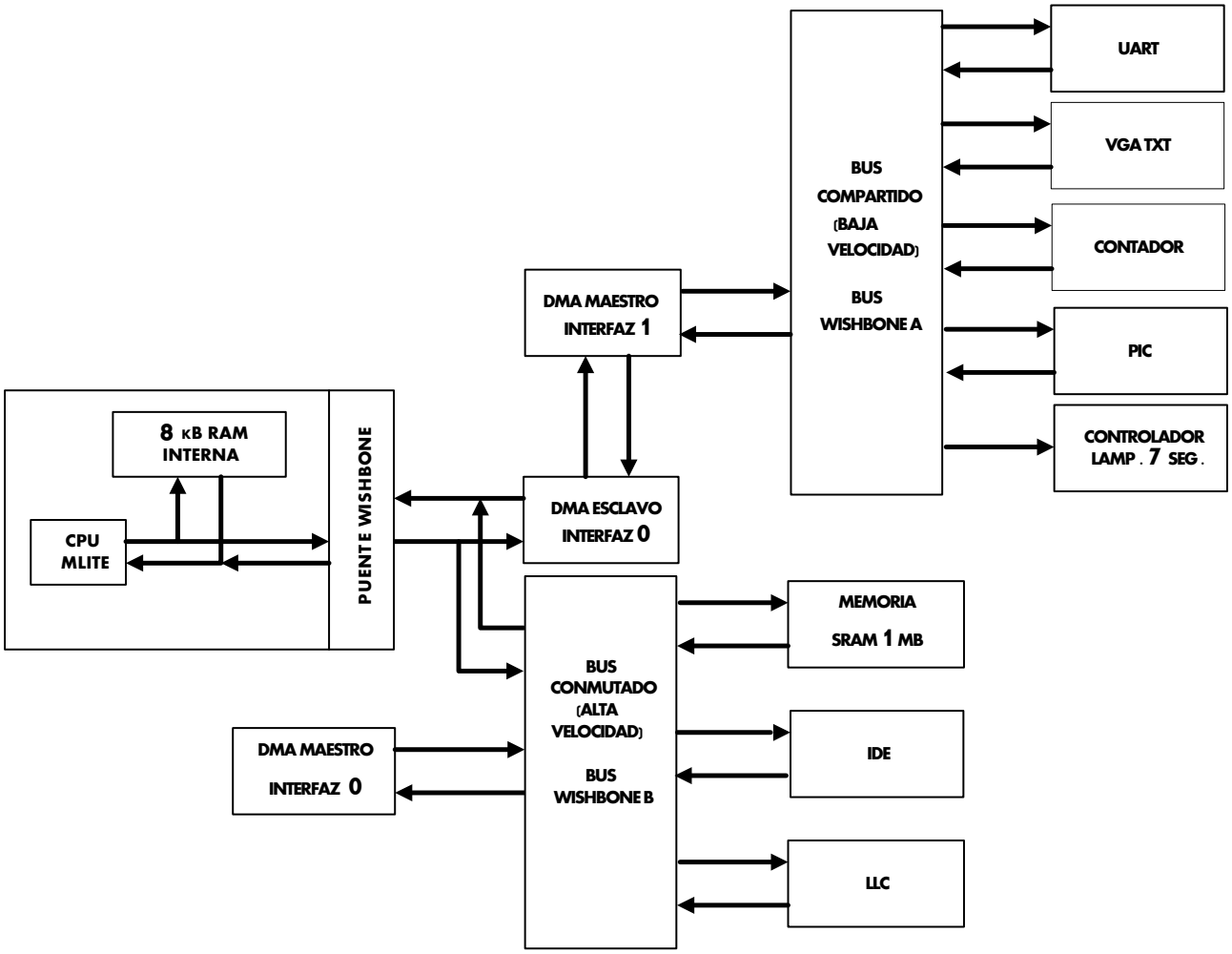


Fig. 2.15 *Arquitectura del SoC Plasma-Wishbone propuesto.*

En la Tabla 2.5 se muestran las direcciones de cada uno de los diferentes dispositivos periféricos en el bus.

Tabla 2.5. Mapa de direcciones del Sistema on Chip propuesto.

Periférico	Dirección
RAM Interna (8 kB)	0x00000000
SRAM Externa (1 MB)	0x10000000
UART 0	0x21000000
VGA Text	0x22000000
Controlador Vis. 7 Seg	0x23000000
Contador	0x24000000
PIC	0x25000000
DMA	0x30000000
IDE	0x40000000
LLC	0x50000000

La arquitectura propuesta es flexible y escalable, pues permite la inclusión de módulos maestros y esclavos con relativo poco esfuerzo y sin afectar ostensiblemente el rendimiento del sistema de manera global. Todo ello, teniendo en cuenta que se ha basado en un estándar de bus mundialmente extendido y para el cual se desarrollan a diario nuevos módulos de propiedad intelectual con nuevas funcionalidades.

2.8 Simulación y resultados.

Como parte del proceso de simulación y prueba se creó un programa en lenguaje C que accede al módulo OCIDEC-3 implementado (ver Anexo 3). El programa se enlaza en la dirección 0x00000000 con el fin de que se ejecute en la memoria interna (8 kB). Se debe aclarar que este no es en ningún sentido el programa que queda en el diseño final pues no provee la funcionalidad del cargador de arranque. En la simulación se realizan accesos a la interfaz IDE para leer en el Registro de Estado [STAT] de dicho módulo la versión implementada, que en el caso de este trabajo es la tres (3) (ver epígrafe 2.6). Una vez leído el número de la versión este se envía al módulo de visualización (visualizadores 7- segmentos) (ver simulaciones en los Anexos del 4 al 6).

2.9 Utilización de los recursos de la FPGA

La Tabla 2.6 muestra los resultados de la síntesis del sistema completo para una FPGA XC3S1000-4FT256.

Tabla 2.6. Utilización de los recursos de la FPGA

Bloques	Utilizados	Disponibles	Costo
S	5944	7680	77%
SFF	4108	15360	26%
4ILUT	8130	15360	52%
IOBs	107	173	61%
RAMB16s	8	24	33%
BUFGMUXs	3	8	37%
DCMs	2	4	50%

Dónde:

S: Bloques lógicos configurables (*Slices*).
SFF: *Slices Flip/Flops*.
4ILUT: Tablas de búsqueda de 4 entradas (LUTs).
IOBs: Bloques de entrada/salida.
RAMB16s: Bloques de RAM.
BUFGMUX: *Buffer* Multiplexor de Reloj Global.
DCM: Administrador de Reloj Digital.

2.10 Valoración socio-económica.

En la actualidad existen diversos dispositivos los cuales se comunican mediante el protocolo IEEE-1394, en caso de ser un periférico de computadora o querer comunicarse con una de ellas, deben existir en la placa base de la computadora módulos con puertos IEEE-1394. En el mercado internacional se pueden encontrar diversos módulos de expansión a través de tiendas y sitios de venta de Internet, los precios varían en dependencia de las características, marca y país. En la Tabla 2.7 se relacionan los precios de algunos de estos módulos.

Tabla 2.7. Precios de diferentes módulos de interfaz IEEE-1394.

Tipo de tarjeta	**Características Generales**	**Precio (dólares)**
Tarjeta Controladora *FireWire* PCI, 4 puertos + cable, 3bumen	Provee funciones *Bus Master OHCI interface*. Compatible con IEEE 1394-1995 y compatible con 1394A. Soporta velocidades de 100, 200 y 400 Mbps. Puertos internos y externos para computadoras de escritorio, con protección independiente para cada puerto.	$ 24.900
Tarjeta Controladora *FireWire* PCI, 4 Puertos + Cable, 3bumen	Fácil conectividad *Plud-n-play*. Transferencia de datos USB 2.0 hasta 480 Mbps, IEEE 1394 hasta 400 Mbps. Compatible con dispositivos USB 1.1. Puertos externos USB2.0 e IEEE 1394 *FireWire* perfectamente adaptables a tu PC. Soporta dispositivos hot swap.	$ 29.000
Tarjeta *FireWire* PCI A IEEE 1394 de 3+1	PCI VIA VT6306 *chipset* de la interfaz. 3 +1 (3 externos y 1 puerto interno) Integrado 400Mbit PHY Soporta el dominio de alto rendimiento de bus. Soporta hasta 400 Mbite/sec.	$ 40.000

Tabla 2.7. Precios de diferentes módulos de interfaz IEEE-1394.

Tipo de tarjeta	**Características Generales**	**Precio (dólares)**
Tarjeta PCI *Express* con 2 puertos *FireWire* 1394a - Startech.com	Esta tarjeta adaptadora *FireWire* 400 se puede instalar en cualquier ranura PCI *Express* para agregar 2 puertos IEEE 1394a, haciéndola una selección ideal para conectar un amplio rango de dispositivos compatibles con *FireWire* 400 como por ejemplo dispositivos de almacenamiento externo, cámaras grabadoras de vídeo digital, cámaras de vídeo HD, multimedia personal y mucho más.	$47.000
Tarjeta Adaptadora PCI *FireWire* 1394a de 4 puertos con *kit* de edición de video digital Startech.com	Tres puertos *FireWire*-400 externos y 1 interno compartido, con soporte para velocidades de transferencia de hasta 400Mbps. Compatible con los estándares 1394a - 2000 y 1394-1995. Incluye *kit* de Edición de Video: *Ulead Video Studio SE* y cable *FireWire*. Soporta *Plug and Play* y *Hot-Swap*.	$ 300.000

A pesar de existir gran variedad de propuestas para la obtención de módulos *IEEE -1394*, por una serie de situaciones se hace difícil la obtención de alguno de estos módulos, imposibilitándose el uso de dispositivos que utilicen este protocolo, como es el caso de la cámara BST-HDCE existente en el laboratorio de microscopía del departamento de Biomédica de la facultad. Dado que en la entidad se cuenta con varios *kits* de desarrollo con Arreglos Programables de Campos de Compuertas (FPGA) para el diseño e implementación de dispositivos digitales, se usa esta disponibilidad como el soporte de *hardware* y la estructura electrónica sobre la cual se desarrolla este trabajo para así obtener una estructura alternativa para almacenar los flujos de datos requeridos por los modos isócronos de este protocolo en un dispositivo de almacenamiento masivo IDE , lo que permitiría el uso de medios que se encontraban en desuso como es el caso de la cámara BST-HDCE y otros equipos que utilicen el protocolo.

CONCLUSIONES

A partir del estudio del estándar *IEEE – 1394*, la interfaz IDE y de los SoC sobre FPGA se realizó el diseño de una arquitectura de nodo *IEEE – 1394* robusta, embebida en una FPGA y basada en bus Wishbone, la cual permite el almacenamiento en tiempo real del flujo de datos requerido por los modos de comunicación isócronos del protocolo *IEEE-1394* en un dispositivo de almacenamiento masivo IDE, lo que constituye un paso adicional hacia la posibilidad de comunicación de un dispositivo que utilice el protocolo *IEEE – 1394* y un dispositivo capaz de almacenar datos en tiempo real.

En el trabajo también se aportan conocimientos sobre el protocolo *IEEE – 1394*, el cual es poco conocido y divulgado a pesar de estar presente en diversos dispositivos comerciales, con un papel significativo en la transmisión de video en tiempo real.

RECOMENDACIONES

1. Añadir al diseño propuesto un controlador de interfaz SATA y probarlo en una tarjeta de desarrollo que soporte el nuevo controlador, para obtener un diseño más eficiente y a la altura de las nuevas tecnologías.

2. Mejorar el diseño de la capa de transacciones y del nivel de manejo del bus para implementar completamente la última versión del protocolo *IEEE – 1394*.

REFERENCIAS BIBLIOGRÁFICAS

[1] Sitio oficial de Apple™. Disponible en: http://www.apple.com. Consultado: 29 de Abril de 2014.

[2] *IEEE 1212-1991: Control and Status Register (CSR) Standard*. Disponible en: http://standards.ieee.org/findstds/standard/1212-1991.html. Consultado: 29 de Abril de 2014.

[3] *IEEE Std 1394-1995, Standard for a High Performance Serial Bus*. 1996. ISBN 1-55937-583-3. Disponible en: http://shop.ieee.org/ieeestore. Consultado: 29 de Abril de 2014.

[4] *IEEE Std 1394a-2000 (Amendment to IEEE Std 1394-1995)*. Disponible en: http://shop.ieee.org/ieeestore. Consultado: 29 de Abril de 2014.

[5] *IEEE 1212-2001: Control and Status Register (CSR) Standard*. Disponible en: http://shop.ieee.org/ieeestore. Consultado: 29 de Abril de 2014.

[6] *IEEE Std 1394b-2002 (High Speed Supplement)*. Disponible en: http://shop.ieee.org/ieeestore. Consultado: 29 de Abril de 2014.

[7] *IEEE 1394.1-2004 Bridging*. Disponible en: http://shop.ieee.org/ieeestore. Consultado: 29 de Abril de 2014.

[8] *IEEE P1394c (S800T) Working Group website*. Disponible en: http://grouper.ieee.org/groups/1394/c/Drafts/1394c.pdf. Consultado: 29 de Abril de 2014.

[9] *1394 Trade Association*. Disponible en: http://www.1394ta.org. Consultado: 7 de Mayo de 2014.

[10] *1394 TA Power Spec Part 1: Cable Power Distribution*. Disponible en: http://www.1394ta.org/Technology/Specifications/Descriptions/TA_1999001-1.htm. Consultado: 7 de Mayo de 2014.

[11] *1394 TA Power Spec Part 2: Suspend/Resume*. Disponible en: http://www.1394ta.org/Technology/Specifications/Descriptions/TA-1999001-2.htm. Consultado: 7 de Mayo de 2014.

[12] *1394 TA Power Spec Part 3: Power State Management*. Disponible en: http://www.1394ta.org/Technology/Specifications/Descriptions/T1999001-3-R95Final.htm. Consultado: 7 de Mayo de 2014.

[13] *1394 Trade Association. FireWire Design Guide 1.0*. 03/02/2010. Disponible en: http://www.1394ta.org/developers/DesignGuide/TAFWDesignGuide.pdf. Consultado: 7 de Mayo de 2014.

[14] *AV/C General Specification*. Disponible en: http://www.1394ta.org/Technology/specifications. Consultado: 7 de Mayo de 2014.

[15] *MPEG4 over 1394.* Disponible en: http://www.1394ta.org/Technology/specifications. Consultado: 7 de Mayo de 2014.

[16] *IEC 61884-4. Standard that describes MPEG Transport Streams (TS).* Disponible en: http://www.iec.ch/webstore/. Consultado: 12 de Mayo de 2014.

[17] IEC 61883-6. Disponible en: http://www.iec.ch/webstore. Consultado: 12 de Mayo de 2014.

[18] *Digital Transmission Licensing Authority (DTLA)*. Disponible en: http://www.mpeg.org/1394. Consultado: 12 de Mayo de 2014.

[19] *1394 Open Host Controller Interface (OHCI) Specification V1.1*. Enero 6, 2000. Disponible en: http://www.microsoft.com/whdc/system/bus/1394/OHCI.mspx . Consultado: 12 de Mayo de 2014.

[20] *Microsoft 1394 Plug & Play Specification.* Disponible en: http://www.microsoft.com/whdc/resources/respec/specs/1394PNP.mspx . Consultado: 12 de Mayo de 2014.

[21] Johansson, P. *IPv4 over IEEE 1394. RFC 2734.* Diciembre 1999. Disponible en: http://www.faqs.org/rfcs/rfc2734.html. Consultado: 12 de Mayo de 2014.

[22] Fujisawa K. *DHCP for IEEE 1394*. RFC2855. Junio 2000. Disponible en: http://www.faqs.org/rfcs/rfc2855.html. Consultado: 12 de Mayo de 2014. http://www.rfc-editor.org/rfc/rfc2855.txt. Consultado: 13 de Mayo de 2014.

[23] Fujisawa K. RFC 3146 - *Transmission of IPv6 Packets over IEEE 1394 Networks*. Octubre 2001. Disponible en: http://www.faqs.org/rfcs/rfc3146.html. Consultado: 12 de Mayo de 2014.

[24] *Serial Bus Protocol 2 (SBP-2)* ftp://ftp.t10.org/t10/drafts/sbp2/sbp2r04.pdf. Consultado: 13 de Mayo de 2014. http://webstore.ansi.org/ansidocstore. Consultado: 16 de Mayo de 2014.

[25] *Serial Bus Protocol 3* (SBP-3) ftp://ftp.t10.org/t10/drafts/sbp3/sbp3r05.pdf. Consultado: 13 de Mayo de 2014. http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbp3r05.pdf. Consultado: 16 de Mayo de 2014.

[26] *1394 Automotive Specification (IDB – 1394) 1.0*. TA2001018. Disponible en: http://www.1394ta.org/developers/specifications/2001018.html. Consultado: 7 de Mayo de 2014.

[27] *PMD for Fiber Optic Wake-on-LAN. Revision 1.0*. TA2004024. Disponible en: http://www.1394ta.org/developers/specifications/2004024.html. Consultado: 7 de Mayo de 2014.

[28] *1394b for Military Applications*. SAE-AS5643. Disponible en: http://www.sae.org/tecnical/standars/AS5643/1. Consultado: 16 de Mayo de 2014.

[29] *Lockheed Martin F-22 Raptor*. Disponible en: http://es.wikipedia.org/wiki/Lockheed_Martin_F-22_Raptor.html. Consultado: 16 de Mayo de 2014.

[30] *Lockheed Martin F-35 Lightning II*. Disponible en: http://www.1394ta.org/industries/aerospace. Consultado: 7 de Mayo de 2014.

[31] *Video Electronics Standards Association Home Network*. CEA 851-A. Disponible en: http://www.ce.org/Standards/browseByCommittee_2739.asp. Consultado: 16 de Mayo de 2014.

[32] *User Interface for Home Networks*. CEA 2027-A. Disponible en: http://www.ce.org/Standards/browseByCommittee_2758.asp. Consultado: 16 de Mayo de 2014.

[33] *Digital Video Broadcasting: 1394 Home Network Segment*. ETSI TS 102 813. Disponible en: http://portal.etsi.org/broadcast/dvb.htm. Consultado: 16 de Mayo de 2014.

[34] *Digital Video Broadcasting: Home Local Network*. ETSI TS 101 225. Disponible en: http://pda.etsi.org/pda/queryform.asp. Consultado: 16 de Mayo de 2014.

[35] *Protocol Adaptation Layer for IEEE 1394 over IEEE 802.15.3*. TA2003010. Disponible en: http://www.1394ta.org/developers/specifications/2003010.html. Consultado: 7 de Mayo de 2014.

[36] *1394 Trade Association. IIDC (Intrumentation & Industrial Digital Camera) 1394-based Digital Camera Specification.* V1.31. Disponible en: http://www.1394ta.org/developers/specifications/2003017.html. Consultado: 7 de Mayo de 2014.

[37] *1394 Trade Association. Industrial & Instrumentation Control Protocol*. Disponible en: http://www.1394ta.org/developers/specifications/1999016.html. Consultado: 7 de Mayo de 2014.

[38] *1394 Trade Association*. *IEEE 488 over 1394*. Disponible en: http://www.1394ta.org/developers/specifications/1999017.html. Consultado: 7 de Mayo de 2014.

[39] Buchanan W., *Computer Busses, Napier University, Edinburgh*. Cap. 22 pp. 345 – 348.

[40] *1394 Automation Protocol.* Disponible en: http://www.1394ta.org/developers/specifications/2005099.html. Consultado: 7 de Mayo de 2014.

[41] TOPCON *Corporation. Retinal Camera Instruction Manual*. TRC-50DX.

[42] TOPCON *Corporation*. Disponible en: http://www.topcon.co.jp. Consultado: 16 de Mayo de 2014.

[43] *Allied Vision Technologies.* Disponible en: http://www.alliedvisiontec.com/. Consultado: 16 de Mayo de 2014.

[44] Tanembaum, Andrew S., Redes de computadoras, 3ra ed., Ed. Pearson, 2001.

[45] Sánchez, Santiago, Introducción al diseño con FPGAS. Universidad de Sevilla.

[46] *Altera Corporation*. Disponible en: http://www.altera.com. Consultado: 19 de Mayo de 2014.

[47] *Xilinx Inc*. Disponible en: http://www.xilinx.com. Consultado: 19 de Mayo de 2014.

[48] *Xilinx Inc., DS299 ChipScope Pro ILA (v. 1.00a, 1.01a, 1.02a),* Marzo 24, 2008.

[49] *IEEE standard VHDL language reference manual - IEEE Std 1076-1987*. Disponible en: http://atc2.aut.uah.es/~marcos_s/LabArqCom/ManualUsuarioVHDL.pdf. Consultado: 19 de Mayo de 2014.

[50] *Mentor Graphics*. Disponible en: http://www.mentor.com/. Consultado: 19 de Mayo de 2014.

[51] Rodriguez M. I. E., Controlador de capa de enlace IEEE – 1394 embebido en una FPGA. Tesis presentada en opción al título de ingeniero biomédico. Universidad de Oriente. Santiago de Cuba, Cuba, Junio 2011.

[52] *Texas Instruments*. TSB12LV31C *Data Manual*. IEEE 1394-1995 *General-Purpose Link-Layer Controller*. Septiembre 1998.

[53] Rhoads, S, *Plasma - most MIPS I(TM) opcodes*. [En línea] Disponible en: http://opencores.org/project,plasma.

[54] *Agere Systems*, *FW803 PHY IEEE 1394A Three-Cable Transceiver/Arbiter Device Data Sheet*. Consultado: 30 de Junio de 2014.

[55] *Philips Semiconductors*, *PDI1394P11A 3-port physical layer interface Data Sheet*, Marzo 1999.

[56] SGS – *Thomson Microelectronics*, *SBPH400-3 IEEE1394 3-Port 400Mbps Physical Layer*, Marzo 1998.

[57] Sitio oficial de *Western Digita*l. Disponible en: http://www.wdc.com/sp/. Consultado: 30 de Junio de 2014.

[58] Sitio oficial de *Seagate Technology*. Disponible en: http://www.seagate.com/. Consultado: 30 de Junio de 2014.

[59] Sitio oficial de *Compaq Computer Corporation*. Disponible en: http://www.compaq.com/. Consultado: 30 de Junio de 2014.

[60] *OpenCores* organization, *OpenCores* Project. Disponible en: http://www.opencores.org. Consultado: 4 de Julio de 2014.

[61] *Silicore Corp. Specifications for*: WISHBONE System-*on-Chip* (SoC) *Interconnection Architecture for Portable IP Cores. Revisión: B.3*. Consultado: 4 de Julio de 2014.

[62] Rhoads S. "*Plasma Web Server*". 2005. Disponible en: http://plasmacpu.no-ip.org:8080/. Consultado: 4 de Julio de 2014.

[63] *Mips Lite success story*. Disponible en: http://wwwhni.upb.de/~gruenewa/research/mlite_success_story.php. Consultado: 4 de Julio de 2014.

[64] *IBM Microelectronics*, *Core connect bus architecture*. Disponible en: http://www.ibm.com/chips/products/coreconnect. Consultado: 4 de Julio de 2014.

[65] *ARM Limited, AMBA specifications v2.0*, 1999. Disponible en: http://www.arm.com. Consultado: 4 de Julio de 2014.

[66] *Altera Corp., Avalon bus specification: Reference manual*. Disponible en: http://www.altera.com. Consultado: 4 de Julio de 2014.

[67] *Draft standard omi 324: Pi-bus, rev. 0.3d, open microprocessor systems initiative, Siemens AC, Munich, 1994*. Disponible en: http://www.cordis.lu/esprit/src/omihome.htm. Consultado: 4 de Julio de 2014.

[68] K. Lahiri, S. Dey, A. Raghunathan, *Design of communication architectures for high-performance and energy-efficient systems-on-chip, in Multiprocessor Systems on- Chips, A. A. Jerraya and W.Wolf, Eds. Amsterdam: Elsevier, 2005, pp. 187–222.*

[69] *OpenCores Organization, User's Manual Wishbone Builder*. Disponible en: http://www.opencores.org/project,wb_builder,downloads. Consultado: 4 de Julio de 2014.

[70] Usselmann R., *Wishbone DMA/Bridge IP Core*, 2002. Disponible en: http://www.opencores.org/project,wb_dma,downloads. Consultado: 4 de Julio de 2014.

[71] Gorban, J., *UART IP Core Specification*, 2002. Consulta: Disponible en: http://www.opencores.org/project,uart16550,downloads. Consultado: 4 de Julio de 2014.

[72] Herveille R., *Simple Programmable Interrupt Controller*, 2002. Disponible en: http://www.opencores.org/project,simple_pic,downloads. Consultado: 4 de Julio de 2014.

[73] Valcarce García, Javier., *Monochrome Text-Mode VGA Video Display Adapter*. Disponible en: http://opencores.org/project,interface_vga80x40. Consultado: 5 de Marzo de 2013.

[74] Herveille Richard, *ATA/ATAPI – 5 Core Specification*, 2002. Disponible en: http://www.opencores.org/ . Consulta: 2 de mayo de 2014.

ANEXOS

Anexo I: Características de la Cámara Digital BST – HDCE– 10 (20).

Parámetros	**HDCE-10**	**HDCE-20**
Sensor de imagen	1/3″CCD	1/2″CCD
Resolución YUV 4:2:2	1280×1024(1.3M Pixel)	1600×1200(2M Pixel)
Tamaño de pixel	4.65μm×4.65μm	4.65μm×4.65μm
Salida digital	24–bit (color)	24–bit (color)
Formato de la imagen YUV 4:2:2	1280×1024	1600×1200
	800×600	1024×768
	640×480	800×600
		640×480
Iluminación más baja	1.5 lux	2.2 lux
Exposición	Proceso de exposición Manual/Auto, Tiempo de exposición ajustable (1~500ms).	Proceso de exposición Manual/Auto, Tiempo de exposición ajustable (1~500ms).
SNR	> 45dB	> 45dB
Rango dinámico	62 dB	62 dB
Modos de Conexión	Insertar directamente en el tubo del ocular del microscopio o usar el montaje C estándar.	Insertar directamente en el tubo del ocular del microscopio o usar el montaje C estándar.
Salida de imagen	Conectar y desconectar por IEEE 1394 para la alimentación y el control automático de energía.	Conectar y desconectar por IEEE 1394 para la alimentación y el control automático de energía.
Dimensión	80 mm×73 mm×45mm	80 mm×73 mm×45mm
Peso	400g	400g

Anexo II: Distribución de terminales y descripción de las señales del dispositivo IDE de 40 terminales.

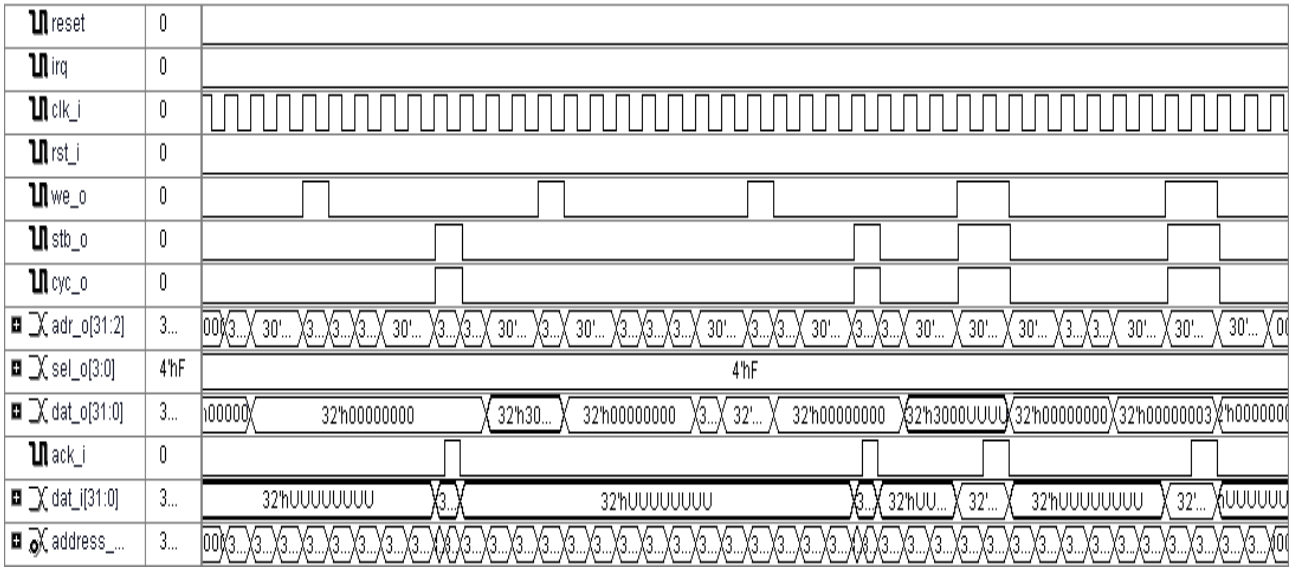
Description	Source	Pin	Acronym
Reset	Host	1	RESET-
	n/a	2	Ground
Data bus bit 7	Host/Device	3	DD7
Data bus bit 8	Host/Device	4	DD8
Data bus bit 6	Host/Device	5	DD6
Data bus bit 9	Host/Device	6	DD9
Data bus bit 5	Host/Device	7	DD5
Data bus bit 10	Host/Device	8	DD10
Data bus bit 4	Host/Device	9	DD4
Data bus bit 11	Host/Device	10	DD11
Data bus bit 3	Host/Device	11	DD3
Data bus bit 12	Host/Device	12	DD12
Data bus bit 2	Host/Device	13	DD2
Data bus bit 13	Host/Device	14	DD13
Data bus bit 1	Host/Device	15	DD1
Data bus bit 14	Host/Device	16	DD14
Data bus bit 0	Host/Device	17	DD0
Data bus bit 15	Host/Device	18	DD15
Ground	n/a	19	Ground
(keypin)	n/a	20	Reserved
DMA Request	Device	21	DMARQ
Ground	n/a	22	Ground
I/O Write	Host	23	DIOW-
Ground	n/a	24	Ground
I/O Read	Host	25	DIOR-
Ground	n/a	26	Ground
I/O Ready	Device	27	IORDY
Spindle Sync or Cable Select	(note 1)	28	SPSYNC:CSEL
DMA Acknowledge	Host	29	DMACK-
Ground	n/a	30	Ground
Interrupt Request	Device	31	INTRQ
16 Bit I/O	Device	32	IOCS16-
Device Address Bit 1	Host	33	DA1
PASSED DIAGNOSTICS	(note 1)	34	PDIAG-
Device Address Bit 0	Host	35	DAO
Device Address Bit 2	Host	36	DA2
Chip Select 0	Host	37	CS0-
Chip Select 1	Host	38	CS1-
Device Active or Slave (Device 1) Present	(note 1)	39	DASP-
Ground	n/a	40	Ground

Anexo III: Secuencia de arranque para comprobar el correcto funcionamiento del controlador de interfaz IDE.

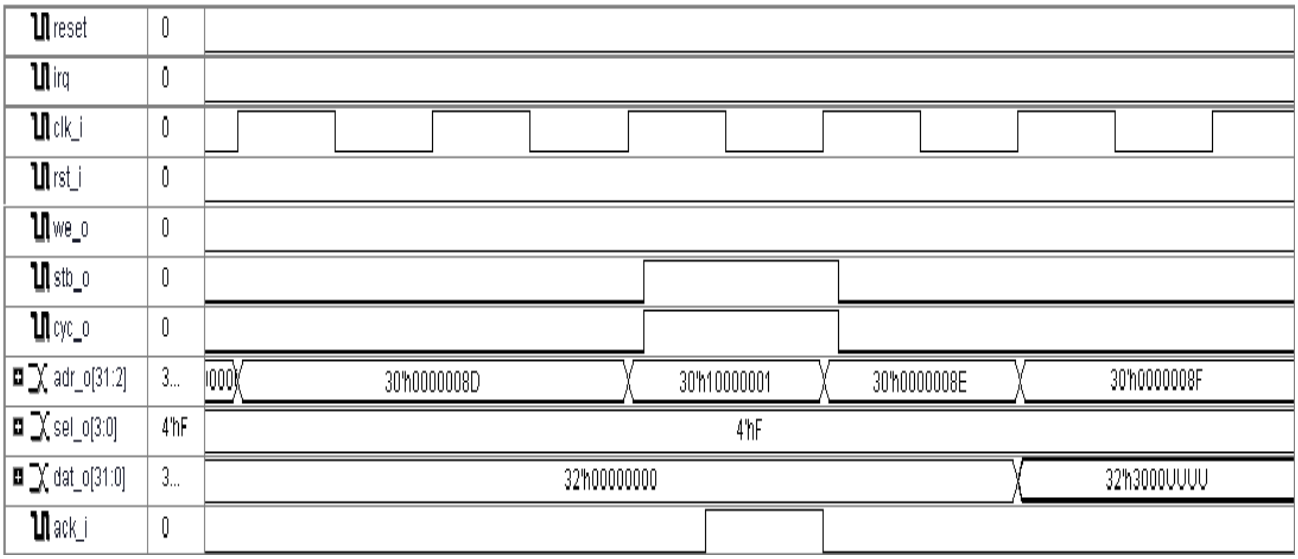
```
#include "plasma_ide.h"

#define MemoryRead(A) (*(volatile unsigned long*)(A))
#define MemoryWrite(A,V) *(volatile unsigned long*)(A)=(V)

void OS_InterruptServiceRoutine(void)
{
}

int main()
{
	
	unsigned long int Val;
	
	Val = MemoryRead(IDE_STAT);
	
	Val >>= 28;
	
	MemoryWrite(LAMP_7SEG, MemoryRead(IDE_STAT));
		
	MemoryWrite(LAMP_7SEG, Val);
	
	return 0;
}
```

En esta secuencia de arranque se definen dos funciones, una de lectura y otra de escritura. Primero se guarda en la variable Val el valor leído en el Registro de Estado [STAT] de OCIDEC-3, como este registro es de 32 bits, se toma de esta lectura los bits del 28 al 32 que son los de interés ya que definen la versión del controlador. Luego se manda a escribir en los visualizadores 7 segmentos el valor guardado en Val correspondiente a la versión implementada del controlador IDE.

Anexo IV: Simulaciones.

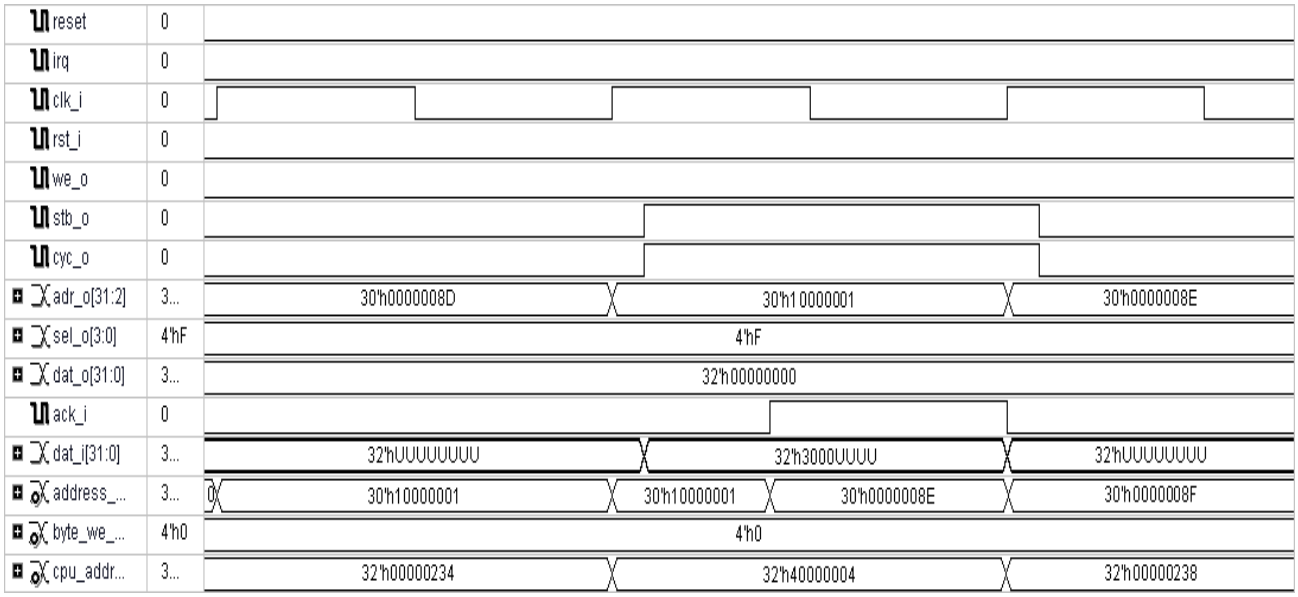


Procesos de lectura del STAT de OCIDEC y escritura en los visualizadores 7 segmentos.

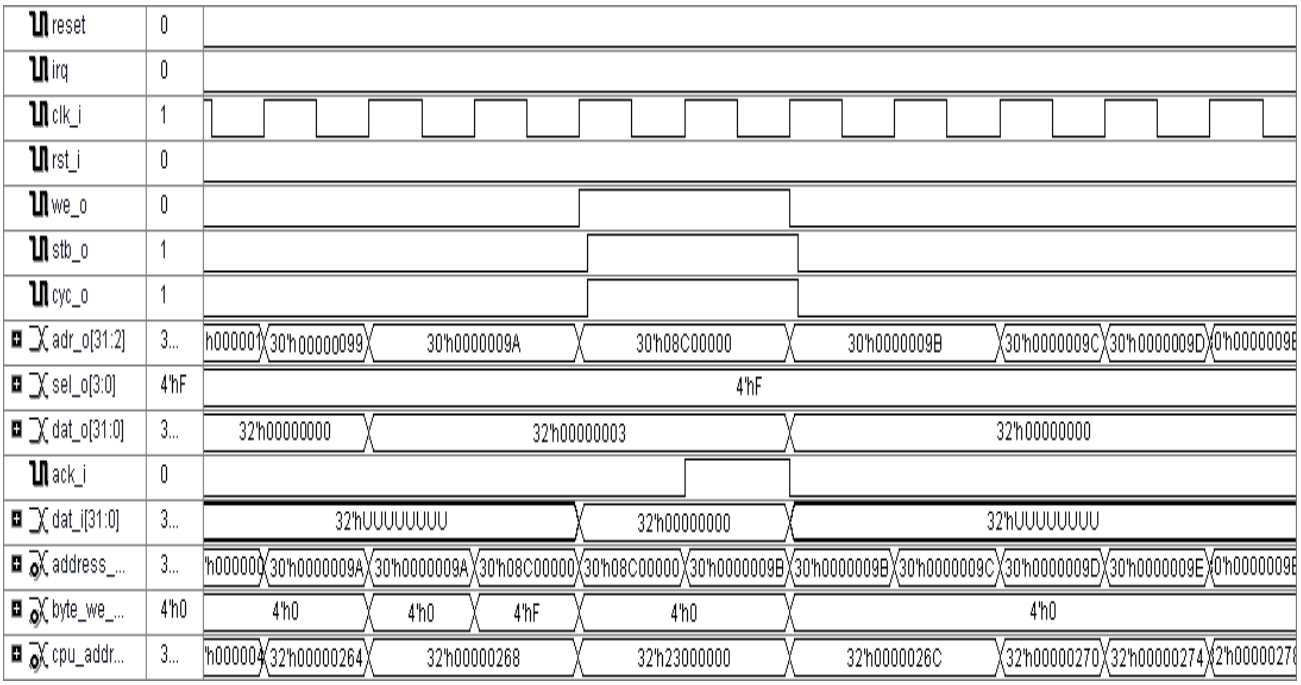


Primera lectura del STAT de OCIDEC.

Anexo V: Simulaciones.

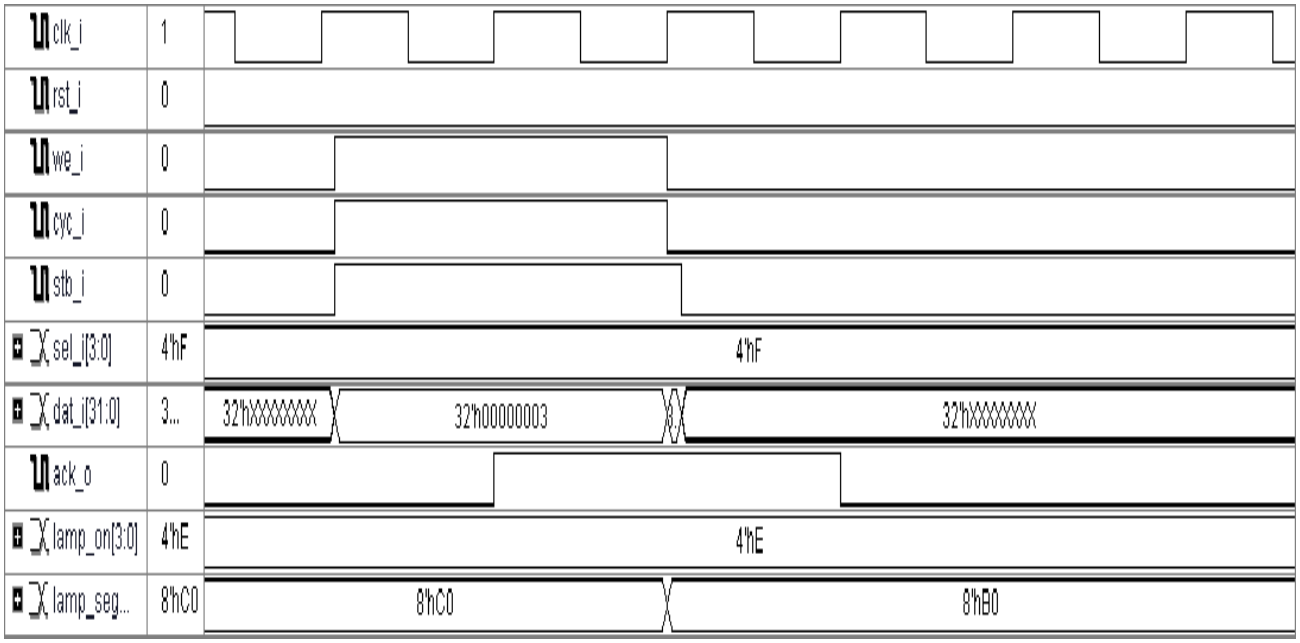


Segunda lectura del STAT de OCIDEC.



Procesos de escritura en los visualizadores 7 segmentos de valor leído en STAT.

Anexo VI: Simulaciones.



Proceso de escritura en los visualizadores 7 segmentos de la versión correspondiente a OCIDEC guardada en la variable Val. En este caso se pasa a visualizar el valor 8'hB0 que corresponde al valor 8'b1011000 (visualizándose en los segmentos 3, versión implementada de OCIDEC).

dp	g	f	e	d	c	b	a
1	0	1	1	0	0	0	0