UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA

PROYECTO FIN DE CARRERA

I.T.I: ELECTRÓNICA INDUSTRIAL

DISEÑO DE UNA CÁMARA ESTÉREO CON MICROPROCESADOR EMBEBIDO PARA APLICACIONES BIOMETRICAS.

Autor: Amadeo Ángel Velasco García

Director: Michael V. García Lorenz

Septiembre, 2010

Título: Diseño de una cámara estéreo con microprocesador embebido para aplicaciones biométricas.

Autor: Amadeo Ángel Velasco García

Director: Michael V. García Lorenz

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día ____ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A todos mis seres queridos.

Resumen

Este proyecto consiste en el diseño de una cámara estéreo, adaptada para que su control se realice mediante un micro-controlador embebido. La **cámara estéreo** tendrá unos registros internos, en los cuales el micro-controlador puede escribir órdenes y leer el estado de la misma.

El micro-controlador que se utilizará para el control de la **cámara estéreo** será **Microblaze**. A este micro-controlador se le introducirá el Sistema Operativo **Petalinux**, que servirá de interfaz entre el usuario y **Microblaze**.

Microblaze se comunicará con sus periféricos a través del bus de comunicación **PLB** (Processor Local Bus), por lo que se diseñará un módulo de la **cámara estéreo** que permita la comunicación a través de este protocolo.

La **cámara estéreo** captará varias imágenes instantáneamente, para lograr con la unión de estas una única imagen con efecto en **3D** (3 Dimensiones).

Las imágenes se almacenarán en varias memorias externas de tipo **RAM** (Random-Access Memory), y el número de estas será igual al número de imágenes que se quiera captar. En este proyecto se captarán tres imágenes como máximo.

Para el diseño de la **cámara estéreo** se utilizará un lenguaje de programación hardware, llamado **VHDL** (Very High Description Language).

Una vez diseñada la **cámara estéreo** se procederá a la comprobación de su funcionamiento mediante simulación con la ayuda de la herramienta Modelsim.

Palabras Clave:

Cámara estéreo, Microblaze, Petalinux, PLB, 3D, RAM, VHDL.

Abstract

This project consists of the design of a **stereo camera**, adapted to be controlled by an embedded micro-controller. The **stereo camera** will have some internal registers, on which the micro-controller can write commands and read the status of the **stereo camera**.

Microblaze will be the micro-controller used for the control of the **stereo camera**. In this micro-controller will be inserted **Petalinux** operating system, which will serve as interface between user and **Microblaze**. The **Microblaze** will communicate with its peripherals via the communication bus **PLB** (Processor Local Bus), so it designed a **stereo camera** module capable of communicate through this protocol.

The **stereo camera** will capture multiple images instantly, and with the union of this, achieves a single image, with effect **3D** (three dimensions). The images will be stored in external memories RAM (Random-Access Memory), and the number of these will be equal to the number of pictures that you want to capture. At this project will be a maximum of three images.

For the design of the stereo camera is used a hardware programming language called VHDL (Very High Description Language).Once designed the stereo camera, it will proceed to the verification of its efficiency through simulations with the assistance of ModelSim tool.

Keywords:

Stereo camera, Microblaze, Petalinux, PLB, 3D, RAM, VHDL.

Índice general

Introducción y objetivos 23
1.1 Introducción 23
1.2 Objetivos 24
1.3 Fases de desarrollo 26
1.4 Medios empleados 28
1.5 Estructura de la memoria 29
Estado del arte 37
2.1 Introducción 37
2.2 Estado del arte 37
Diseño del sistema 51
3.1 Introducción 51
3.2 Diseño de la cámara estéreo 53
3.2.1 Diseño PLB_camara 57
3.2.1.1 Diseño Interfaz_PLBcam 63
3.2.1.2 Diseño Cámara 74
3.2.2 Diseño control_imagenes_ext 80
3.2.3 Diseño NimagenN 85
3.2.4 Diseño de la memoria externa 87
3.2.5 Diseño DCM_48MHZ 96
3.3 Actualización del periférico OPB_cypress_usb 97
3.4 Conexión cámara estéreo al Microblaze 101
3.4.1 Configuración de Microblaze 101
3.4.2 Integración de periféricos al Microblaze 107

3.4.3 Compilación del sistema 122
3.4.4 Inclusión del sistema operativo al Microblaze 123
3.5 Diseño del software de control para la cámara estéreo 126
3.5.1 Desarrollo del programa de control 127
3.5.2 Introducción del software en el Microblaze 129
3.6 Cargador de Petalinux 129
Descripción del sistema 133
4.1 Introducción 133
4.2 Recursos utilizados por el periférico cámara estéreo 133
4.2.1 PLB_camara 134
4.2.2 Control_imagenes_ext 136
4.2.3 DCM48mhz 137
4.2.4 Nimagen 138
4.2.5 Memoria_externa 139
4.3 Recursos utilizados por el periférico PLB_cypress_usb 140
4.4 Recursos utilizados por la cámara estéreo y Microblaze 142
Pruebas 147
5.1 Introducción 147
5.2 Simulaciones hardware de la cámara estéreo 148
5.2.1 Interfaz_PLBcam 148
5.2.2 Funcionamiento de la PLB_cam 152
5.2.3 Funcionamiento Control_imagenes_ext 158
5.2.4 NimagenN y mem_800x600_8BITS 160
5.3 Pruebas del software 164
5.3.1 Funcionamiento de Petalinux 164
5.3.2 Comprobación de la comunicación entre el sistema y la cámara estéreo 171
Conclusiones y trabajos futuros 177
6.1 Conclusiones 177
6.2 Trabajos futuros 178

Referencias **179**

Presupuesto y Diagrama de Gantt **181**

Glosario **187**

Índice de figuras

Figura 2.1: Gafas anáglifo [6] **39**
Figura 2.2: casco HMD [6] **40**
Figura 2.3: Representación de la conversión a 3D a partir de varias imágenes **41**
Figura 2.4: Geometría del cálculo de profundidad [4] **42**
Figura 2.5: Aparato estereoscópico del siglo XIX [6] **44**
Figura 2.6: cámara estéreo Verascope [6] **45**
Figura 2.7: cámara estéreo Kodak Stereo [6] **46**
Figura 2.8: cámara estéreo ImageTech 3D Trio [6] **46**
Figura 2.9: cámara estéreo Vivitar 3D [6] **47**
Figura 2.10: cámara estéreo 3DTech Camera [6] **47**
Figura 3.1: Esquema de la cámara estéreo **56**
Figura 3.2: Puertos de la cámara estéreo **58**
Figura 3.2 (Bis): Puertos de la cámara estéreo **59**
Figura 3.3: Estructura PLB_camara **63**
Figura 3.4: Estructura del bus PLB [7] **64**
Figura 3.5: Fases de la comunicación del PLB [7] **65**
Figura 3.6: Proceso lectura del PLB [7] **69**
Figura 3.7: Proceso de escritura del PLB [7] **70**
Figura 3.8: Puertos de la interfaz_PLBcam **71**
Figura 3.9: Estructura de la interfaz_PLBcam **73**
Figura 3.10: Esquema del modulo cámara **75**
Figura 3.11: Evolución de la estructura ROM_i2c a ROM_cam2 ..... **76**
Figura 3.12: Puertos del modulo cámara **78**
Figura 3.13: Diagrama de estados de la máquina de estados control_imagenes_ext **82**

Figura 3.14: Puertos del modulo control_imagenes_ext 84
Figura 3.15: Estructura del modulo nimagen 86
Figura 3.16: Puertos del modulo nimagen 87
Figura 3.17: Ventana Bienvenida Core Generator 88
Figura 3.18: Ventana de ubicación del proyecto 89
Figura 3.19: Ventana de selección de placa Xilinx 90
Figura 3.20: Ventana selección de la Block Memory Generator 91
Figura 3.21: Ventana diseño de la memoria 92
Figura 3.22: Ventana selección de las dimensiones de memoria 93
Figura 3.23: Ventana selección del pin reset 94
Figura 3.24: Ventana generando memoria 95
Figura 3.25: Puertos del modulo memoria externa 96
Figura 3.26: Estructura cámara estéreo (con solo una cámara) 97
Figura 3.27: Puertos del modulo PLB_cypress_usb 99
Figura 3.27 (Bis): Puertos del modulo PLB_cypress_usb 100
Figura 3.28: Ventana ubicación del proyecto en XPS 101
Figura 3.29: Ventana selección placa Xilinx en XPS 102
Figura 3.30: Ventana selección frecuencia en XPS 103
Figura 3.31: Ventana selección de periféricos en XPS 104
Figura 3.32: Ventana configuración de aplicaciones en XPS 105
Figura 3.33: Ventana resumen de Microblaze en XPS 106
Figura 3.34: Ventana selección parámetros SPLB del periférico 109
Figura 3.35: Ventana selección de interrupción 110
Figura 3.36: Periféricos conectados al Bus interfaz PLB 112
Figura 3.37: asignación de direcciones del bus PLB a periféricos 113
Figura 3.38: Puertos externos definidos en system.mhs 119
Figura 3.39: Puertos externos eliminados del archivo system.mhs 120
Figura 3.40: Conexión de interrupciones por prioridades 121

Figura 3.41: Conexión de pines de la cámara a los conectores de expansión de la placa 122
Figura 3.42: Activación del Mark to Initialize BRAMs para compilación 123
Figura 3.43: Ventana de selección de Petalinux 124
Figura 3.44: Ventana selección parámetros de Petalinux..... 125
Figura 3.45: Diagrama de flujo del programa control cámara..... 128
Figura 5.1: Simulación proceso de escritura..... 149
Figura 5.2: Simulación proceso de lectura 150
Figura 5.3: Simulación funcionamiento simultaneo de cámaras 151
Figura 5.4: Simulación inicio configuración sensor 152
Figura 5.5: Simulación fin configuración sensor 153
Figura 5.6: Zoom de simulación fin configuración sensor..... 153
Figura 5.7: Simulación encendido sensor..... 154
Figura 5.8: Imagen 1 de simulación apagado de sensor..... 155
Figura 5.9: Imagen 2 de simulación apagado de sensor..... 156
Figura 5.10: Simulación fin de almacenamiento de imágenes..... 157
Figura 5.11: Simulación de lectura de las imágenes almacenadas..... 158
Figura 5.12: Simulación de almacenamiento ordenado de imágenes 159
Figura 5.13: Simulación lectura de los datos de imagen..... 160
Figura 5.14: Simulación escritura de la imagen 161
Figura 5.15: Zoom simulación escritura de la imagen..... 162
Figura 5.16: Simulación lectura de la imagen..... 163
Figura 5.17: Login de Petalinux..... 169
Figura 5.18: Código para el Test Camera Stereo 173
Figura 5.19: Funciones de escritura y lectura de memoria del Microblaze..... 174
Figura 5.20: Resultado Test Camera Stereo 174

Índice de tablas

Tabla 4.1: Recursos utilizados por el periférico PLB_camara..... 134
Tabla 4.2: Dispositivos utilizados por la PLB_camara 135
Tabla 4.3: Recursos utilizados por el periférico Control_imagenes_ext 136
Tabla 4.4: Dispositivos utilizados por Control_imagenes_ext..... 136
Tabla 4.5: Recursos utilizados por el periférico dcm48mhz..... 137
Tabla 4.6: Dispositivos utilizados por dcm48mhz 138
Tabla 4.7: Recursos utilizados por el periférico nimagen..... 138
Tabla 4.8: Dispositivos utilizados por nimagen 139
Tabla 4.9: Recursos utilizados por el periférico memoria externa 139
Tabla 4.10: Dispositivos utilizados por la memoria externa..... 140
Tabla 4.11: Recursos utilizados por el periférico PLB_cypress_usb..... 141
Tabla 4.12: Dispositivos utilizados por PLB_cypress_usb 141
Tabla 4.13: Recursos utilizados por la cámara estéreo y Microblaze 143
Tabla 4.14: Dispositivos utilizados por la cámara estéreo y Microblaze..... 143

Capítulo 1

Introducción y objetivos

1.1 Introducción

Este proyecto titulado "*Diseño de una cámara estéreo con microprocesador embebido para aplicaciones biométricas*", forma parte de un proyecto global, cuya misión es obtener un dispositivo de reconocimiento biométrico, a partir de las características faciales humanas, en tres dimensiones.

Este proyecto se centra en la obtención de las imágenes y en el control de los dispositivos captadores.

El diseño de esta cámara estéreo viene motivado por el hecho de intentar realizar un dispositivo de captación de imágenes más rápido que los existentes, realizando el diseño mediante la tecnología de la microelectrónica, que ofrece resultados óptimos, y mediante la reducción de las tareas del micro-controlador, las cuales serán tareas de control básicas, como por ejemplo, el encendido y apagado de la cámara, mediante la lectura y escritura de registros.

Indicar que para la realización de la cámara estéreo, se va a tomar como referencia el proyecto *"Diseño y control de una cámara de video digital"*, que implementa el control de un sensor de imagen, y el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*, que diseña el micro-controlador Microblaze, introduciéndole el sistema operativo Petalinux y añadiéndole periféricos al sistema.

Dicho lo anterior, en este proyecto, se realizará un sistema captador de imágenes, cuyo diseño se realizará mediante el lenguaje hardware VHDL.

Para el control de la cámara estéreo se utilizará el microprocesador Microblaze de 32bits, usando Petalinux como Sistema Operativo.

Dicha cámara estéreo estará compuesta por dos o tres cámaras, de iguales características, las cuales captarán tres imágenes, que al unirse y mediante algunos algoritmos específicos, formarán una imagen estéreo. Esta imagen se analizará para el reconocimiento biométrico. En este proyecto sólo se captarán las imágenes, dejando para futuros proyectos el diseño del software para la unión y reconocimiento de las imágenes.

La manera de proceder será diseñando el control de una cámara (un sensor), y luego multiplicarla, según el número de imágenes que se quiera captar, para formar la cámara estéreo.

La cámara consistirá en una adaptación de la cámara previamente diseñada en el proyecto *"Diseño y control de una cámara de video digital"*, la cual se modificará para que pueda comunicarse con un micro-controlador embebido, en este caso Microblaze.

Además se actualizará el diseño del micro-controlador realizado en el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*, el cual se ha quedado obsoleto porque se utiliza el bus OPB para la comunicación con los periféricos.

1.2 Objetivos

El objetivo fundamental del proyecto es el de diseñar una cámara estéreo, capaz de almacenar dos o tres imágenes, de forma rápida, mediante lenguajes hardware, como VHDL, reduciendo las tareas del micro-controlador, que en este caso sólo tendrá tareas de escritura y lectura sobre los registros de la cámara estéreo.

En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Comunicar la cámara estéreo al Microblaze:

Para realizar el control de la cámara desde el exterior, será necesario introducir a la cámara estéreo un micro-controlador. Este micro-controlador transferirá datos a sus periféricos a través de un bus de comunicaciones. Este bus de comunicaciones se controlará mediante un protocolo de comunicación que deberá ser entendido por el

micro-controlador y por sus periféricos. El micro-controlador que se utilizará es Microblaze que dispone de varios protocolos de comunicación: OPB, PLB, FSL, etc.

Se realizará una parte lógica que se introducirá dentro de la cámara estéreo, y cuya función será entender el protocolo de comunicación utilizado por Microblaze. En este caso, el protocolo escogido para que se puedan comunicar entre ellos será PLB.

- Adaptar la cámara para su control mediante registros:

La cámara que se utilizará como referencia en este proyecto, está diseñada para la captación de una imagen, cuyos controles básicos son el encendido y el apagado.

El objetivo de este punto será introducir dentro de la cámara unos registros a los que se pueda acceder desde el exterior, lo cual facilitará las tareas de control de la misma. Para ello, se introducirá un registro de control y un registro de estado, cuya función será informar al usuario del estado de la cámara en cada momento.

También se insertarán registros de datos, donde se almacenarán los datos de configuración de la cámara (resolución, zoom, etc.) y las características de la imagen. Gracias a la introducción de estos registros en la cámara estéreo, el micro-controlador solamente tendrá que leer y escribir en estos registros, dejando las demás tareas al diseño hardware.

- Aumentar la resolución de la imagen captada.

La cámara de referencia capta una imagen con la mínima resolución, por tanto, en este nuevo diseño de la cámara, se intentarán captar imágenes con resoluciones mayores, creando memorias externas.

En este proyecto, el tamaño de la memoria externa estará limitado por la placa de Xilinx utilizada.

- Aumentar los bits que definen el color de la imagen. (De 1bit a 8bits)

Al igual que la resolución captada por la cámara que se tomará como referencia, el número de bits que definirán a la imagen será sólo uno (1 bit), obteniéndose así una imagen en blanco y negro. En el nuevo diseño se realizará una captación de la imagen con 8 bits.

- Configurar Microblaze.

Se realizará la configuración de Microblaze necesaria para el buen funcionamiento del sistema.

- Realizar una interfaz cámara estéreo – usuario.

Por último, se desarrollará un programa de control que actuará de interfaz entre usuario y cámara estéreo. Las funciones de control de este programa serán básicas, como pueden ser el encendido y el apagado de la cámara, configuración, y lectura de las imágenes.

1.3 Fases de desarrollo

En primer lugar, se procederá a desglosar en módulos las distintas funcionalidades de la cámara estéreo para facilitar el desarrollo.

Los distintos módulos que se observan en la cámara estéreo son:

- Módulo “PLB_cam”:

La función de este módulo será el control del sensor de imagen de la cámara.

Para el desarrollo de este módulo se realizará un estudio de la cámara diseñada en el proyecto *"Diseño y control de una cámara de video digital"*. A partir de este estudio se realizarán diversas modificaciones para la adaptación al nuevo sistema.

Para realizar el diseño del código que permitirá a la cámara estéreo comunicarse con el micro-controlador, se estudiarán las características y el funcionamiento de los distintos protocolos de comunicación de los que dispone Microblaze.

- Módulo "control_imagenes_ext":

La función de este módulo es la gestión del almacenamiento de las imágenes en sus respectivas memorias.

- Módulo "nimagen" y "memoria externa":

La función de estos módulos será el acondicionamiento de la imagen, y su almacenamiento, respectivamente.

Para la realización de estos módulos se estudiarán los recursos disponibles en la placa Virtex4 Ml402sx que se utilizará en este proyecto.

Será necesaria la obtención del tamaño máximo de cada imagen y se procederá a la creación de la memoria externa con la ayuda de la herramienta Core Generator.

Posteriormente se realizará la actualización del periférico **"OPB_cypress_usb"**, diseñado en el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*, para que pueda comunicarse con Microblaze mediante el protocolo PLB.

El siguiente paso será la configuración del micro-controlador Microblaze y la unión al mismo de los diferentes módulos diseñados, por medio del bus PLB.

A continuación, se realizará la introducción del software Petalinux dentro de Microblaze, siguiendo los pasos descritos en el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*.

También se realizará el diseño del software que servirá de interfaz entre el usuario y la cámara estéreo.

Por último, se realizarán pruebas mediante simulación de los códigos implementados, donde se comprueba que todas las señales se activan de forma correcta, demostrando el buen funcionamiento de la cámara.

1.4 Medios empleados

Para la realización de este proyecto se utilizarán las siguientes herramientas software:

ISE de Xilinx: Para la sintetización del diseño hardware de la cámara estéreo.

Core Generator de Xilinx: Para la creación del código de la memoria externa.

XPS de Xilinx: Para la creación y configuración del Microblaze, así como para la conexión de la cámara estéreo a este, mediante el bus PLB.

Modelsim de Mentor Graphics: Herramienta para la simulación de los diseños realizados en el lenguaje hardware VHDL.

GCC de GNU: Herramienta para la compilación de códigos en lenguaje C, en este caso se utilizara para la compilación del sistema operativo Petalinux.

También se utilizarán los siguientes Sistemas Operativos:

Petalinux: Sistema Operativo que se instalará en Microblaze para la interacción del microprocesador con el exterior.

Sistema Operativo Fedora 10: Sistema Operativo que se utilizará para la configuración del Software de Petalinux.

Se ha escogido esta plataforma porque sólo se puede realizar la configuración de Petalinux bajo un sistema Unix.

1.5 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

Capítulo 2: Estado del arte:

En este capítulo se realizará una pequeña descripción de la técnica de la estereoscopía, indicando en qué consiste, los diferentes métodos para conseguir los efectos de la estereoscopía, así como el modelo matemático para la formación de imágenes estéreo a partir de dos o más imágenes.

También se realizará una pequeña descripción de la historia de las cámaras estéreo, así como una lista de las mismas que han ido surgiendo a lo largo de su historia hasta los días actuales, con algunas de sus características, donde se comprueba que se comienzan a utilizar los sensores digitales.

Por último, se expondrán algunas de las aplicaciones de las cámaras estéreo.

Capítulo 3: Diseño del sistema

En este capítulo se describirán los pasos que se realizarán para el diseño del sistema. El sistema estará compuesto por varias partes, cada una con su cometido. Estas partes son: la cámara estéreo, Microblaze, Petalinux y un software de control básico de la cámara estéreo. La cámara estéreo recibirá órdenes de control de Microblaze, como apagado, encendido, lectura de imágenes, configuración del sensor de imagen, etc. Estas señales de control u órdenes se recibirán a través del bus de comunicación. Las señales de control que enviará Microblaze a la cámara estéreo serán seleccionadas por el usuario a través del software de control de la cámara estéreo, que se ejecutará bajo el Sistema Operativo Petalinux, instalado previamente en el micro-controlador Microblaze.

Por tanto, en este capítulo, se realizará un análisis del protocolo que se utilizará en la comunicación de la cámara estéreo con el micro-controlador, indicando cuáles son los protocolos disponibles en el micro-controlador Microblaze, la elección y el por qué de esa elección.

Para la descripción de la cámara estéreo se realizará un desglose en módulos. Uno de estos módulos será la Camara_PLBcam, que recibirá mediante el bus PLB, una palabra de 32 bits, que irá a parar a los registros internos de esta. La información contenida en estos registros determinará el control y el estado de la cámara estéreo.

El módulo Camara_PLBcam, estará conectado al módulo Control_imagenes_ext, que gestionará el almacenamiento de las imágenes, y al módulo Nimagen, que acondicionará los datos para almacenarlos en la memoria externa.

El módulo Control_imagenes_ext a su vez estará conectado a la memoria externa.

A continuación se realizará un pequeño resumen de los módulos que forman esta cámara estéreo.

Camara_PLBcam → Aquí se describirá el funcionamiento del módulo y los registros internos que lo forman, y se mencionarán los puertos de los que estará compuesto.

Este módulo a su vez se ha dividido en otros dos módulos llamados:

- Interfaz_PLBcam: La función de este módulo es la gestión de las señales para la buena comunicación de la cámara estéreo con el micro-controlador por medio del bus PLB.

  En este punto se realizará una descripción de las características, del funcionamiento del protocolo de comunicación que se va a utilizar junto con las señales más importantes de éste.

  También se describirá el desarrollo de la interfaz_PLBcam, donde se mencionará los puertos que forman este módulo y las señales más importantes que lo componen.

- Cámara: En este punto se mencionarán las modificaciones que se realizarán de los códigos de referencia que se utilizarán para la adaptación al sistema, del proyecto *"Diseño y control de una cámara de video digital"*, cuyo cometido es el control y configuración del sensor de imagen que dependen de las señales de control enviadas por el micro-controlador. Por último, se mencionarán los puertos que forman parte del módulo y se describirá la función de las señales más importantes que lo componen.

Control_imagenes_ext → Aquí se describirá la función de este módulo, que es la de gestionar el almacenamiento de las imágenes, en el orden y memoria correctos. También se realizará una descripción de las señales y de los puertos más importantes que lo forman.

Nimagen → En este punto se describirá el funcionamiento de este módulo, cuya función es la de acondicionar los datos que le llegan de la cámara para su posterior almacenamiento en la memoria externa.

Memoria_externa → En este punto se describirá los pasos que se seguirán para la creación de una memoria externa a partir de la herramienta CORE Generator de Xilinx que será utilizada para el almacenamiento de las imágenes.

A continuación se realizará la descripción de la actualización que se llevará a cabo del periférico "OPB_cypress_usb". Se mencionará la causa de la actualización de este periférico, así como los pasos seguidos para su actualización.

Posteriormente, se describirá la conexión de la cámara estéreo con Microblaze. Se importarán los periféricos necesarios, y por último se compilará el sistema.

Una vez compilado el sistema se introducirá el software Petalinux dentro de Microblaze, y se diseñará un software sencillo de control básico de la cámara estéreo que servirá de interfaz entre el usuario y esta última.

Capítulo 4: Descripción del sistema:

En este capítulo se mostrará una descripción de los recursos utilizados por cada módulo del periférico de la cámara estéreo. Se destacarán los detalles de los componentes más utilizados y se llegará a la conclusión de que el diseño no consume demasiados recursos a excepción de algunos componentes como las RAMB16s que se necesitarán para la creación de las memorias externas.

Se realizará un análisis de los recursos utilizados por el periférico actualizado "PLB_cypress_usb". Se demostrará que este periférico tampoco consume demasiados recursos.

También se realizará una descripción de los recursos utilizados por todo el conjunto, es decir, de los utilizados por Microblaze y sus periféricos, incluyendo la cámara estéreo. Se observará que los recursos utilizados por el sistema completo son en su mayoría, utilizados por Microblaze y los periféricos necesarios, para su buen funcionamiento.

Capítulo 5: Pruebas:

En este capítulo se realizarán las simulaciones para observar si los códigos diseñados funcionan correctamente. Para realizar un análisis más detallado se realizará la simulación de los módulos más importantes por separado. En primer lugar, se realizará la simulación del módulo "interfaz_PLBcam" (módulo encargado del control de la comunicación de la cámara estéreo con el Microblaze a través del bus PLB) y se comprobará los procesos de escritura y lectura, para observar que las señales que actúan de árbitro en la comunicación se activan en los instantes y en el orden correctos. Más tarde se realizará la simulación del funcionamiento del módulo "PLB_cam", para comprobar los procesos de escritura y lectura de sus registros internos. Se observará a su vez las operaciones de configuración, encendido, apagado, lectura de imágenes y con las indicaciones de fin de almacenamiento de imágenes.

El siguiente módulo se simulará el "control_imagenes_ext", y se comprobará el almacenamiento ordenado de las imágenes y la lectura de los datos de las imágenes.

Por último, se realizará la simulación de los módulos "nimagen" y "mem_externa", y se observará el buen funcionamiento de los procesos de conversión, escritura y lectura de datos.

Para acabar, se realizará la comprobación del software mediante la observación de la correcta ejecución del Sistema Operativo y la comunicación satisfactoria entre la cámara estéreo y Microblaze, a través del bus PLB.

Capítulo 2

Estado del arte

2.1 Introducción

En este capítulo se realizará una pequeña descripción de la técnica de la estereoscopía, indicando en qué consiste, los diferentes métodos para conseguir los efectos de la estereoscopía, así como el modelo matemático para la formación de imágenes estéreo a partir de dos o más imágenes.

También se realizará una pequeña descripción de la historia de las cámaras estéreo, así como una lista de las mismas que han ido surgiendo a lo largo de su historia hasta los días actuales, con algunas de sus características, donde se comprueba que se comienzan a utilizar los sensores digitales.

Por último, se expondrán algunas de las aplicaciones de las cámaras estéreo.

2.2 Estado del arte

Este proyecto consiste, en el diseño de una cámara estéreo, que permita captar un número de imágenes en 2D y en un instante de tiempo, para generar posteriormente una imagen en tres dimensiones.

Las cámaras estéreo o cámaras 3D, se basan en la técnica de la estereoscopía.

La estereoscopía:

La estereoscopía es cualquier técnica para conseguir información tridimensional de una imagen creando una ilusión de profundidad de la misma. Para ello se captan dos o más imágenes en 2D.

La manera más sencilla de obtener una ilusión tridimensional consiste en captar dos imágenes con diferentes ángulos de captación, imitando al sistema visual del ser humano, el cual el ojo izquierdo capta una imagen ligeramente diferente al del ojo derecho. Esto se puede comprobar mirando a un objeto fijamente, y tapándose primero el ojo izquierdo y después el derecho, observando que la perspectiva del objeto cambia. El cerebro es el encargado de juntar las dos imágenes para obtener así una sensación espacial. Con este ejercicio de taparse el ojo izquierdo, y luego el derecho, se observa también, que la imagen sigue manteniendo su ilusión de profundidad. Esto se debe a que el cerebro realiza una construcción intuitiva de la imagen gracias a varios factores que se verán a continuación. Este tipo de ilusión tridimensional que se consigue con una sola imagen, se le denomina visión monocular.

Los factores que consiguen que el cerebro asimile la imagen 2D como una imagen 3D, son los siguientes:

- Distribución de luces y sombras sobre el objeto: La iluminación es un factor intuitivo del volumen muy importante, ya que la sombra y el contraste nos aportan gran sensación de relieve.

- Superposición de imágenes: Cuando un objeto se encuentra en superposición a otro, es decir, se encuentra en la realidad ante otro, el más cercano (delante) cubre al más lejano (detrás).[3]

- Perspectiva: El efecto de *perspectiva* produce una clara sensación de profundidad. Las líneas paralelas horizontales parecen converger en el horizonte.

- Diplopía fisiológica: Para que el cerebro pueda interpretar una imagen en tercera dimensión, requiere de datos sobre la distancia de los objetos. Dicha información se obtiene gracias a que tenemos dos ojos, así cada uno de ellos percibe los elementos de la escena desde un ángulo distinto, dando como resultado una triangulación de la cual el cerebro obtiene la distancia al objeto.

- Movimiento de paralelaje: El desplazamiento del observador produce la impresión de que se mueven los objetos de la escena en un sentido u otro dependiendo de su posición.

Algunas de las técnicas para conseguir una percepción de imagen tridimensional a partir de imágenes 2D, son las siguientes:

- Anáglifos: Los anáglifos son estereofotografías tomadas o tratadas con filtros de distintos colores sobrepuestas en una sola imagen. Se observan por medio de unas gafas llamadas gafas anáglifo y que tienen un filtro de diferente color para cada ojo. La misión de estos filtros es hacer llegar a cada ojo únicamente la imagen que le corresponde. [3]

Figura 2.1: Gafas anáglifo [6]

- Polarización: Se utiliza luz polarizada para separar las imágenes izquierda y derecha. El sistema de polarización no altera los colores, aunque hay una cierta pérdida de luminosidad. Se usa tanto en proyección de cine 3D como en monitores de ordenador mediante pantallas de polarización alternativa. Hoy día es el sistema más económico para una calidad de imagen aceptable.[6]

- Cascos (Head Mounted Display, HMD): Un HMD es un casco estereoscópico que porta dos pantallas y dos sistemas ópticos para cada ojo, de forma que la imagen se genera en el propio dispositivo. Su principal uso hasta ahora ha sido la Realidad Virtual, a un costo prohibitivo y de forma experimental, aunque al bajar de precio aparecen otras aplicaciones lúdicas, como los videojuegos. [6]

Figura 2.2: casco HMD [6]

- Alterantivo (Estéreo Activo): Con este sistema se presentan en secuencia y alternativamente las imágenes de la izquierda y derecha, sincronizadamente con unas gafas dotadas con obturadores de cristal líquido (denominadas LCS, Liquid Crystal Shutter glasses o LCD, Liquid Crystal Display glasses), de forma que cada ojo ve solamente su imagen correspondiente. A una frecuencia elevada, el parpadeo es imperceptible. Se utiliza en monitores de ordenador, TV y cines 3D de última generación. [6]

- Sistema Cromatek: El sistema cromatek utiliza lo que se conoce como rejilla de difracción.

  La rejilla de difracción funciona de manera semejante a un prisma de cristal: la luz que la atraviesa se descompone en colores que cambia de angulación según su tonalidad ya que ésta, está asociada a su frecuencia y por tanto, a su longitud de onda.

  Éste cambio de ángulo que cada color sufre al ser difractado incide en el ojo y hace que los objetos parezcan tener una profundidad distinta según su color. El inconveniente es que para que la desviación del ángulo al difractarse sea notoria respecto a la luz directa que llega al otro ojo, las imágenes tienen que tener colores intensos; por lo que el rango cromático que podremos utilizar queda limitado [3].

- Efecto Pulfritch: El sistema pulfritch está basado en un dato fisiológico respecto al cerebro y dice que éste tarda un poco más en procesar las imágenes oscuras que las claras. Así si se pone un filtro oscuro en un solo ojo y se observa un objeto en movimiento, el cerebro tardará más tiempo en procesar las imágenes procedentes de este ojo. Por lo que si la escena que se observa está en continuo movimiento lateral, la imagen del ojo con filtro parecerá estar en una posición o ángulo distinto con respecto al observado directamente sin filtro, que tendrá la imagen procesada instantes antes.

  La gran ventaja de esta técnica es que las imágenes pueden verse de manera normal si no se utilizan los filtros; pero tiene un inconveniente, y es que requiere que todo el tiempo exista movimiento lateral y en el mismo sentido. Si no, no se percibirá el retraso interpretativo por parte del cerebro, del ojo filtrado respecto al ojo directo [3].

Todos los métodos anteriores, disponen las imágenes para que el cerebro haga la función de juntarlas y así obtener la imagen 3D, pero si se quiere obtener la transformación de la imagen en 3D sin ayuda del cerebro se tendrá que recurrir a modelos matemáticos y a algoritmos que permitan construir la imagen en 3D, a partir de las imágenes en 2D.

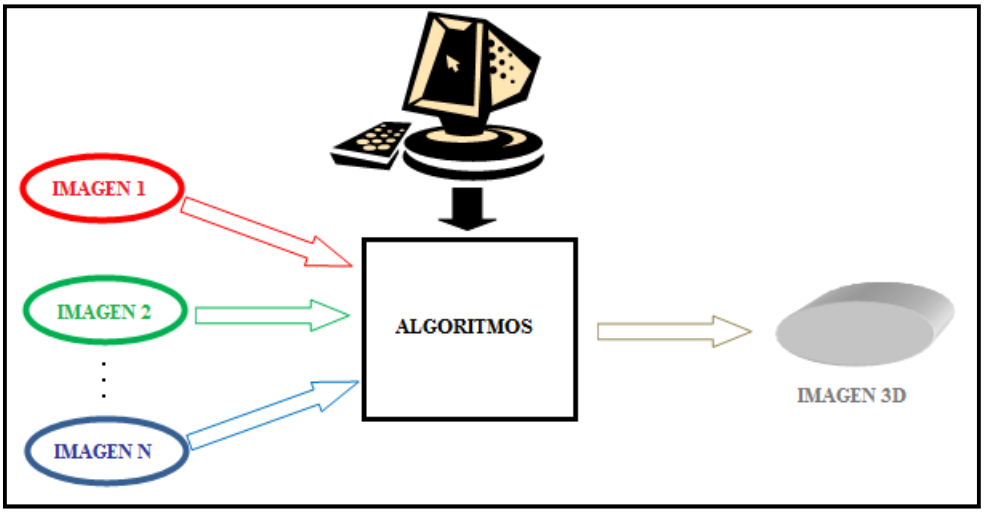


Figura 2.3: Representación de la conversión a 3D a partir de varias imágenes

Uno de los algoritmos más sencillos para la obtención de las imágenes en 3D, es a partir del cálculo de la profundidad con la información obtenida de las imágenes en 2D.

El proceso de obtención de la imagen en 3D es obtener dos imágenes en 2D, identificar los puntos correspondientes en ambas imágenes. Posteriormente mediante geometría se calcula la profundidad (Z) de cada punto – en base a la distancia entre los puntos correspondientes (disparidad) en las imágenes y los parámetros de los objetivos de las cámaras (longitud focal, distancia entre objetivos).

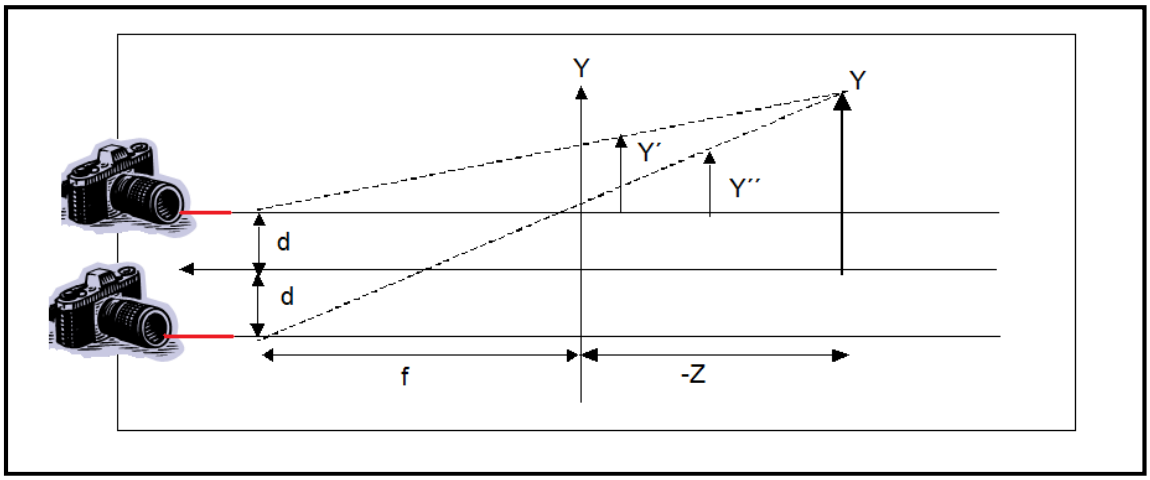


Figura 2.4: Geometría del cálculo de profundidad [4]

Donde la fórmula para obtener la profundidad Z es:

$$Z = f - \frac{2df}{(y'-y'')}$$ [4]

Como se puede observar el algoritmo es sencillo, lo realmente complejo en el proceso de construcción de una imagen en 3D es la de identificar la correspondencia entre puntos.

Dos de las maneras de encontrar la correspondencia de puntos son:

- Estéreo “denso” – se hace la correspondencia en toda la imagen, a nivel pixel (template matching, relajación, …).
- Basado en características – se busca la correspondencia entre ciertas características o patrones distintivos (orillas, esquinas, SIFT,…).

La primera de las maneras utiliza algoritmos bastante más complejos, con un esfuerzo computacional importante, mientras que el segundo realiza los mismos algoritmos que el método anterior, pero sólo se aplica a puntos importantes, como bordes, esquinas, etc. Este método no realiza tanto esfuerzo computacional pero la calidad de la imagen se reduce bastante [4].

Para más información de los algoritmos y técnicas que convierten las imágenes en 3D, consultar las referencias [4].

Como se ha visto anteriormente, es importante la disposición y las características de los objetivos.

Por tanto a la hora de diseñar la estructura de la cámara estéreo, es importante saber cuál es la distancia entre los mismos. A la hora de tomar la decisión de cuál va a ser la distancia entre los ellos de la cámara estéreo, sólo hace falta fijarse cuál es la distancia que separa a nuestros ojos, y así podemos obtener la distancia a la que se pondrá los objetivos para obtener de manera correcta la imagen estéreo. Esta distancia es de más o menos 65mm.

Cuando la base estereoscópica es mayor a 65 mm, el tamaño aparente de los objetos, así como la distancia a que parecen situarse, disminuye en la misma proporción. En cierto modo, es como si el observador fuese de mayor tamaño. A este fenómeno se le denomina hiperestereoscopia [5].

El efecto contrario es el de hipoestereoscopía, cuando la base estereoscópica es menor a 65 mm.

Las cámaras estéreo:

Sus antecedentes se remontan a la época del Renacimiento, cuando Leonardo da Vinci, inspirándose en conocimientos que provienen de la Antigüedad con los trabajos de Euclides y de Galiano, explicó el fenómeno de la visión binocular cuya síntesis ofrece la sensación de relieve, al igual que la doble percepción del sonido estereofónico que se logra por medio de la síntesis auditiva. En el siglo XVI, Della Porta estudió a su vez el fenómeno.

Pero el primero en idear un aparato para proporcionar la visión en relieve, fue el físico inglés Charles Wheatstone (1802-1875); que ideó el estereoscopio, presentado en Londres en 1838. El aparato permitía la visión correspondiente a los 65 mm de distancia que hay entre los ojos. Pero sin la aportación de la fotografía, todo ello no hubiese pasado de ser un experimento óptico, dada la dificultad de realizar a mano dibujos con la visión disociada de cada ojo.

A Sir David Brewster (1781-1868), físico escocés e inventor del caleidoscopio, se debe el propósito de obtener imágenes estereoscópicas por medio de la fotografía. Sus primeros intentos en este campo datan de 1844, cinco años después de la divulgación del guerrotipo. Brewster, que estaba relacionado con Talbot, Adamson y D. O. Hill, pudo adaptar sus trabajos a la incipiente técnica fotográfica.

Figura 2.5: Aparato estereoscópico del siglo XIX [6]

Tras haber pretendido en vano interesar a los ópticos británicos, se dirigió a París, donde el abate Moignot y los ópticos Soleil y Duboscq acogieron sus experimentos. El aparato construido por Duboscq cosechó grandes éxitos en la Exposición Universal de Londres de 1851, donde la reina Victoria recibió un ejemplar de lujo que supuso una excelente publicidad para el invento.

Después de haber realizado las dobles vistas con una cámara cuyo objetivo se desplazaba horizontalmente sobre una plancha graduada, en 1849 lo sustituyó Brewster por una cámara binocular que al sacar sincrónicamente las dos imágenes permitía realizar retratos estereoscópicos.

La técnica estereoscópica evolucionó a lo largo de la segunda mitad del siglo XIX, adaptándose a las mejoras sucesivas de los procedimientos: del estereodaguerrotipo al veráscopo de Richard, que tuvo gran aceptación a principios del siglo XX, hubo múltiples variantes del invento de Brewster. El empleo de la placa de cristal que permitía la visión de imágenes transparentes fue decisivo. Muchos fotógrafos del siglo XIX y de principios del XX realizaron vistas estereoscópicas. El gran aficionado que fue Santiago Ramón y Cajal, utilizó mucho la cámara binocular. Y hasta hace poco se siguieron utilizando los populares aparatos denominados «View-Master». [6]

Algunas de las primeras cámaras estéreo comerciales se exponen a continuación con algunas de sus características:

Verascope 1946: Fabricada en Paris (Francia), por Jules Richard. Fabricada en el año 1946. La separación entre sus lentes es de aproximadamente 63,25 mm.

Figura 2.6: cámara estéreo Verascope [6]

Kodak Stereo: 1954 Fabricada en Nueva York (Estados Unidos), en el año 1954, con una separación entre sus lentes de 70 mm.

Figura 2.7: cámara estéreo Kodak Stereo [6]

Cámaras más actuales son las siguientes:

ImageTech 3D Trio: Cámara fabricada en Georgia, Estados Unidos, por la compañía Image Technology, Inc. Como se puede observar en la fotografía siguiente esta cámara está compuesta por tres lentes separadas unas de las otras a 30 mm.

Figura 2.8: cámara estéreo ImageTech 3D Trio [6]

Vivitar 3D: Fabricada por Vivitar Corp, en Estados Unidos, en el año 2002.

Figura 2.9: cámara estéreo Vivitar 3D [6]

Y por último existen las cámaras estéreo profesionales con aplicaciones industriales, y en las cuales se empiezan a utilizar los sensores de imágenes digitales. Algunos ejemplos son:

3DTech Camera: Aunque no se trata de una cámara estéreo propiamente dicha esta cámara, según el tipo de necesidad, puede ser útil ya que proporciona información 3D (distancias en el mundo real) sobre el entorno de forma rápida y fiable. Incluye un localizador láser, la propia cámara digital, lentes, flash y un proyector.

La cámara utiliza sensor CCD, y el tiempo de procesamiento de la imagen es de 30 segundos para cada imagen, hay que destacar esta característica. Su resolución es de 3072 x 2048, y el color es RGB (8 bits) [6].

Figura 2.10: cámara estéreo 3DTech Camera [6]

Las aplicaciones que se le pueden dar a las imágenes en 3D, son varias, sobre todo en los videojuegos de realidad virtual, donde gracias a unos cascos especiales, puedes introducirte dentro del juego. También en la industria cinematográfica se está empezando a desarrollar películas con tecnología 3D, las cuales poco a poco están generando una ventaja competitiva frente a películas en 2D.

Otras de las aplicaciones de las imágenes en 3D, son las aplicaciones en reconocimiento biométrico, esta utilización de las imágenes en 3D se debe a que estas introducen en la imagen mucha más información, lo que conlleva que el reconocimiento sea más fiable y rápido.

Capítulo 3

Diseño del sistema

3.1 Introducción

En este capítulo se describirán los pasos que se realizarán para el diseño del sistema. El sistema estará compuesto por varias partes, cada una con su cometido. Estas partes son: la cámara estéreo, Microblaze, Petalinux y un software de control básico de la cámara estéreo. La cámara estéreo recibirá órdenes de control de Microblaze, como apagado, encendido, lectura de imágenes, configuración del sensor de imagen, etc. Estas señales de control u órdenes se recibirán a través del bus de comunicación. Las señales de control que enviará Microblaze a la cámara estéreo serán seleccionadas por el usuario a través del software de control de la cámara estéreo, que se ejecutará bajo el Sistema Operativo Petalinux, instalado previamente en el micro-controlador Microblaze.

Por tanto, en este capítulo, se realizará un análisis del protocolo que se utilizará en la comunicación de la cámara estéreo con el micro-controlador, indicando cuáles son los protocolos disponibles en el micro-controlador Microblaze, la elección y el por qué de esa elección.

Para la descripción de la cámara estéreo se realizará un desglose en módulos. Uno de estos módulos será la “Camara_PLBcam”, que recibirá mediante el bus PLB, una palabra de 32 bits, que irá a parar a los registros internos de esta. La información contenida en estos registros determinará el control y el estado de la cámara estéreo.

El módulo “Camara_PLBcam”, estará conectado al módulo “Control_imagenes_ext”, que gestionará el almacenamiento de las imágenes, y al módulo “Nimagen”, que acondicionará los datos para almacenarlos en la “memoria externa”.

El módulo “Control_imagenes_ext” a su vez estará conectado a la “memoria externa”.

A continuación se realizará un pequeño resumen de los módulos que forman esta cámara estéreo.

Camara_PLBcam → Aquí se describirá el funcionamiento del módulo y los registros internos que lo forman, y se mencionarán los puertos de los que estará compuesto.

Este módulo a su vez se ha dividido en otros dos módulos llamados:

- Interfaz_PLBcam: La función de este módulo es la gestión de las señales para la buena comunicación de la cámara estéreo con el micro-controlador por medio del bus PLB.

  En este punto se realizará una descripción de las características, del funcionamiento del protocolo de comunicación que se va a utilizar junto con las señales más importantes de éste.

  También se describirá el desarrollo de la "interfaz_PLBcam", donde se mencionará los puertos que forman este módulo y las señales más importantes que lo componen.

- Cámara: En este punto se mencionarán las modificaciones que se realizarán de los códigos de referencia que se utilizarán para la adaptación al sistema, del proyecto *"Diseño y control de una cámara de video digital"*, cuyo cometido es el control y configuración del sensor de imagen que dependen de las señales de control enviadas por el micro-controlador. Por último, se mencionarán los puertos que forman parte del módulo y se describirá la función de las señales más importantes que lo componen.

Control_imagenes_ext → Aquí se describirá la función de este módulo, que es la de gestionar el almacenamiento de las imágenes, en el orden y memoria correctos. También se realizará una descripción de las señales y de los puertos más importantes que lo forman.

Nimagen → En este punto se describirá el funcionamiento de este módulo, cuya función es la de acondicionar los datos que le llegan de la cámara para su posterior almacenamiento en la memoria externa.

Memoria_externa → En este punto se describirá los pasos que se seguirán para la creación de una memoria externa a partir de la herramienta CORE Generator de Xilinx que será utilizada para el almacenamiento de las imágenes.

A continuación se realizará la descripción de la actualización que se llevará a cabo del periférico "OPB_cypress_usb". Se mencionará la causa de la actualización de este periférico, así como los pasos seguidos para su actualización.

Posteriormente, se describirá la conexión de la cámara estéreo con Microblaze. Se importarán los periféricos necesarios, y por último se compilará el sistema.

Una vez compilado el sistema se introducirá el software Petalinux dentro de Microblaze, y se diseñará un software sencillo de control básico de la cámara estéreo que servirá de interfaz entre el usuario y esta última.

3.2 Diseño de la cámara estéreo

Para el diseño de la cámara estéreo es fundamental la comunicación de la cámara con el micro-controlador para el que control de la misma.

Para la comunicación entre la cámara estéreo y Microblaze se disponen de varios protocolos. Estos protocolos son los siguientes:

OPB (On-chip Peripheral Bus): Bus síncrono utilizado para conectar periféricos con tiempos de acceso variables. Soporta varios maestros y la conectividad de muchos periféricos es sencilla gracias a su identificación por multiplexación distribuida. Soporta transferencias de tamaño de palabra dinámico [8].

PLB (Processor Local Bus): Bus síncrono de alta velocidad, utilizado para conectar periféricos y bloques de memoria interna de la FPGA. Admite varios maestros en su implementación para Microblaze y puede ser utilizado tanto para instrucciones como para datos [8].

LMB (Local Memory Bus): Es igual que el bus PLB con la diferencia de que este sólo admite un maestro, tampoco permite tamaños de palabra diferentes a 32 bits [8].

DCR (Device Control Register): Bus síncrono diseñado para una conexión tipo anillo de periféricos con ancho de banda muy bajo. Sólo soporta un maestro y está diseñado para minimizar el uso de lógica interna [8].

FSL (Fast Simple Link): Este bus permite la comunicación con el procesador a través de sus propios registros internos. El periférico actuaría a modo de co-procesador y la conexión se realiza de manera sencilla a través de unos registros de desplazamiento de 32 bits de ancho [8].

Según las características de los protocolos, se podrían utilizar los buses OPB, PLB ó LMB. El bus LMB se ha descartado por ser poco flexible. En el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"* utiliza el bus OPB para la conexión del periférico USB al Microblaze.

Sin embargo, se ha elegido el bus PLB porque es mucho más rápido que el OPB, con la desventaja de que tiene un funcionamiento más complejo, y utiliza muchas señales de control, lo cual aumenta el tamaño de ocupación del circuito. Con todo ello, se sigue optando por utilizar el PLB, debido a que su complejidad y tamaño es proporcional al número de maestros acoplados al PLB, que en nuestro caso es solamente uno.

Además la herramienta utilizada para el diseño recomienda la instanciación de periféricos mediante el bus PLB, ya que el OPB poco a poco se está quedando obsoleto.

Por tanto, como sólo se utilizará un maestro muchas de las señales de control de PLB se eliminarán porque no harán falta.

Además este protocolo PLB, tiene señales que controlan el tamaño de los buses de datos y direcciones dinámicamente. Señales que también se omiten porque los buses que se utilizarán serán de tamaño fijo.

A la hora de realizar el acoplo, la herramienta de Xilinx XPS, facilita un módulo llamado IPIF que tiene el cometido de acoplar el periférico del usuario al bus PLB, para que el usuario no tenga que estudiarse el protocolo PLB.

Se ha decidido no utilizar este módulo porque utiliza todas las señales de control del PLB, ocupando más espacio de lo necesario.

Se ha realizado un módulo llamado "Interfaz_PLBcam" que incluirá sólo las señales necesarias que se han utilizado en el proceso de comunicación.

Comentado lo anterior, la cámara “PLB_cam” se compone de los siguientes módulos:

- Interfaz_PLBcam
- Cámara

También se ha diseñado un módulo para la gestión del almacenamiento de las tres imágenes, llamado “control_imagenes_ext”.

Y otros tres módulos, “nimagen1”, “nimagen2” e “nimagen3”, que se conectan a las distintas memorias RAM.

Todos estos módulos componen a la cámara estéreo.

En la siguiente página se muestra una figura con la estructura de todo el sistema completo.

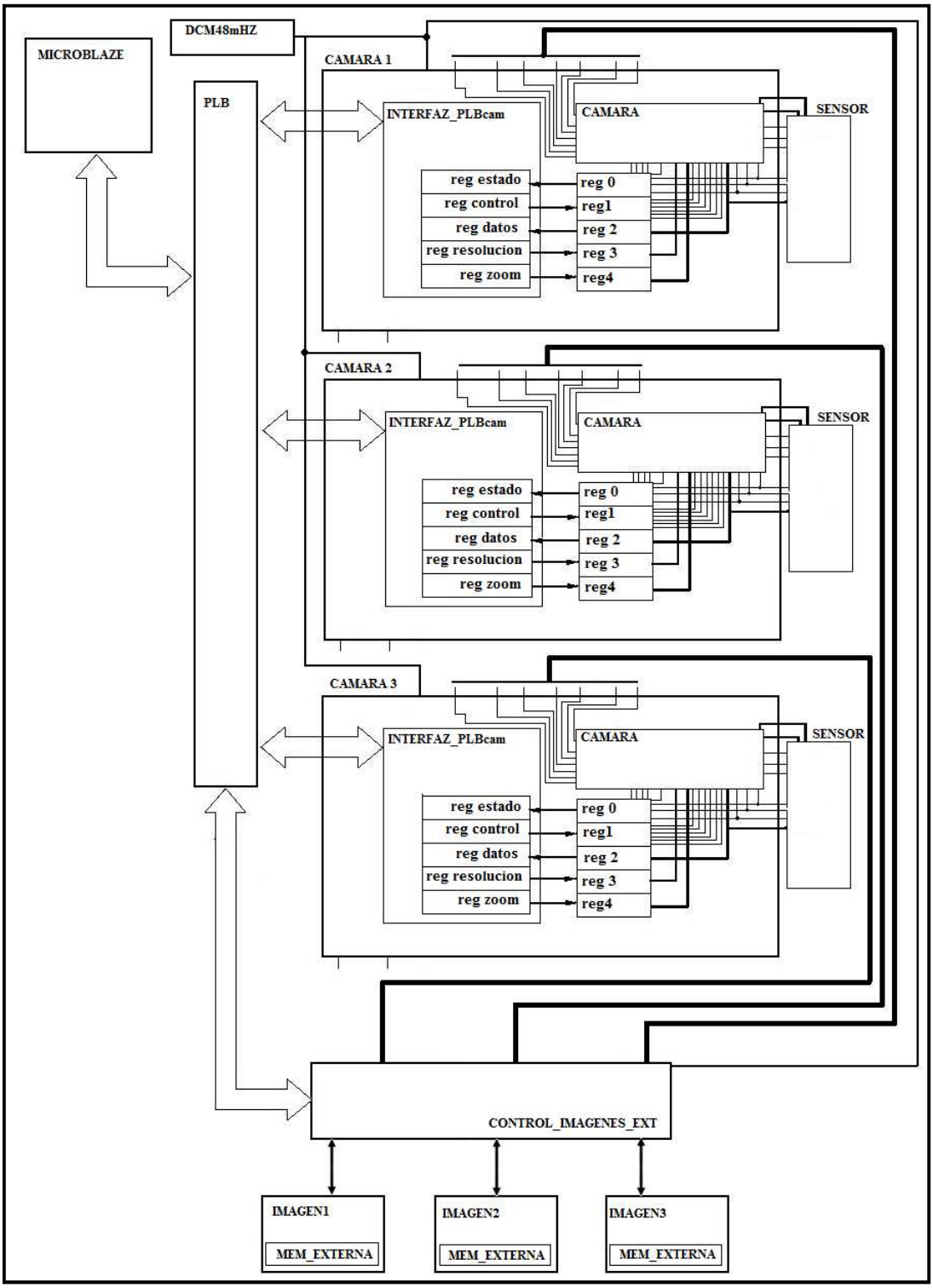


Figura 3.1: Esquema de la cámara estéreo

3.2.1 Diseño PLB_camara

La “PLB_camara” está formada por el módulo “cámara” y por un módulo interfaz que ayuda a la cámara a comunicarse con Microblaze, llamado “interfaz_PLBcam”.

El cometido de la “PLB_camara”, además de unir a los módulos “interfaz_PLBcam” y “cámara”, es la de rellenar los registros con los bits que controlan a la cámara estéreo, así como los bits que indican el estado de la misma.

Los puertos de este módulo final son:

```
port
(
 sys_clk                        : in std_logic;
 sys_rst                        : in std_logic;
 sdata_cam                      : inout std_logic;
 sclk_cam                       : inout std_logic;
 cam_data                       : in std_logic_vector (9 downto 0);
 cam_line_valid                 : in std_logic;
 cam_frame_valid                : in std_logic;
 cam_pixclk                     : in std_logic;
 clkin_cam                      : out std_logic;
 oe_cam                         : out std_logic;
 reset_cam                      : out std_logic;
 act_mem                        : out   std_logic;
 f_col                          : out   std_logic;
 data_out                       : out   std_logic_vector(9 downto 0);
 done                           : out   std_logic;
 fv                             : out   std_logic;
 s_off                          : out   std_logic;
 read                           : out   std_logic;
 f_img                          : in    std_logic;
 SPLB_Clk                       : in  std_logic;
 SPLB_Rst                       : in  std_logic;
 PLB_ABus                       : in  std_logic_vector(0 to 31);
 PLB_UABus                      : in  std_logic_vector(0 to 31);
 PLB_PAValid                    : in  std_logic;
 PLB_SAValid                    : in  std_logic;
 PLB_rdPrim                     : in  std_logic;
 PLB_wrPrim                     : in  std_logic;
 PLB_masterID                   : in  std_logic_vector
                                (0 to C_SPLB_MID_WIDTH-1);
 PLB_abort                      : in  std_logic;
```

Figura 3.2: Puertos de la cámara estéreo

```
PLB_busLock                     : in  std_logic;
PLB_RNW                         : in  std_logic;
PLB_BE                          : in  std_logic_vector
                                (0 to C_SPLB_DWIDTH/8-1);
PLB_MSize                       : in  std_logic_vector(0 to 1);
PLB_size                        : in  std_logic_vector(0 to 3);
PLB_type                        : in  std_logic_vector(0 to 2);
PLB_lockErr                     : in  std_logic;
PLB_wrDBus                      : in  std_logic_vector
                                (0 to C_SPLB_DWIDTH-1);
PLB_wrBurst                     : in  std_logic;
PLB_rdBurst                     : in  std_logic;
PLB_wrPendReq                   : in  std_logic;
PLB_rdPendReq                   : in  std_logic;
PLB_wrPendPri                   : in  std_logic_vector(0 to 1);
PLB_rdPendPri                   : in  std_logic_vector(0 to 1);
PLB_reqPri                      : in  std_logic_vector(0 to 1);
PLB_TAttribute                  : in  std_logic_vector(0 to 15);
Sl_addrAck                      : out std_logic;
Sl_SSize                        : out std_logic_vector(0 to 1);
Sl_wait                         : out std_logic;
Sl_rearbitrate                  : out std_logic;
Sl_wrDAck                       : out std_logic;
Sl_wrComp                       : out std_logic;
Sl_wrBTerm                      : out std_logic;
Sl_rdDBus                       : out std_logic_vector
                                (0 to C_SPLB_DWIDTH-1);
Sl_rdWdAddr                     : out std_logic_vector(0 to 3);
Sl_rdDAck                       : out std_logic;
Sl_rdComp                       : out std_logic;
Sl_rdBTerm                      : out std_logic;
Sl_MBusy                        : out std_logic_vector
                                (0 to C_SPLB_NUM_MASTERS-1);
Sl_MWrErr                       : out std_logic_vector
                                (0 to C_SPLB_NUM_MASTERS-1);
Sl_MRdErr                       : out std_logic_vector
                                (0 to C_SPLB_NUM_MASTERS-1);
Sl_MIRQ                         : out std_logic_vector
                                (0 to C_SPLB_NUM_MASTERS-1)
);
```

Figura 3.2 (Bis): Puertos de la cámara estéreo

La “PLB_cámara” está compuesta por 5 registros internos, 2 son de lectura y 3 de escritura.

Descripción de los registros

Todos los registros son del tamaño de 32 bits.

- Registros de lectura

REGISTRO 0 (REGISTRO DE ESTADO CÁMARA)

Dirección cámara 1 → 0x8200_0000

Dirección cámara 2 → 0x8200_0100

Dirección cámara 3 → 0x8200_0200

BIT 0 → `SENSOR_DONE`. Este bit indica que el sensor ha terminado de captar un frame.

BIT 1 → `F_IMG`. Este bit indica que los sensores que están conectados al bus PLB, han terminado de captar los `frames`. Cuando sólo hay conectado un sólo sensor este bit tiene el mismo cometido que el bit1 de este mismo registro.

BIT 2 → no se utiliza

BIT 3 → no se utiliza

BIT 4 → `CONFIG_DONE`. Este bit se activa para indicar que el sensor ha completado su configuración.

BIT 5 → No se utiliza

BIT 6 → `SENSOR_ACTIVO`. Este bit indica que el sensor está activo y que hay captura de imagen.

BIT 7 → `SENSOR_PARADO`. Este bit indica que el sensor esta en stop y que no hay captura de imagen.

BIT 8 al BIT 30 → No se utiliza.

BIT 31 → ESTOY. Este bit indica que el hardware de la cámara está conectado al bus PLB, es decir, que existe comunicación.

REGISTRO 2 (REGISTRO INFORMACIÓN IMAGEN)

Dirección cámara 1 → 0x8200_0008

Dirección cámara 2 → 0x8200_0108

Dirección cámara 3 → 0x8200_0208

BIT 0 al BIT 9 → Información del numero de columnas que tiene la imagen que se ha guardado en la memoria.

BIT 10 al BIT 19 → Información del número de filas que tiene la imagen que se ha guardado en la memoria.

BIT 20 al BIT 31 → No se utiliza.

- Registros de escritura

REGISTRO 1 (REGISTRO CONTROL CÁMARA)

Dirección cámara 1 → 0x8200_0004

Dirección cámara 2 → 0x8200_0104

Dirección cámara 3 → 0x8200_0204

BIT 0 → I2C_CONFIG. Activando este bit se inicia el proceso de configuración del sensor.

BIT 1 → SENSOR_START. Activando este bit se inicia la captura de imágenes del sensor.

BIT 2 → SENSOR_STOP. Activando este bit se para la captura de imagen del sensor.

BIT 3 → DEFECTO_SENSOR. Activando este bit, los registros internos del sensor tienen los valores por defecto.

BIT 4 → No se utiliza.

BIT 5 → READ. El microprocesador activa este bit cuando se quiere leer las imágenes de la memoria.

BIT 6 al BIT 30 → No se utiliza.

BIT 31 → ESTAS. Este bit se activa para la comprobación de la comunicación entre Microblaze y la cámara estéreo.

REGISTRO 3 (REGISTRO RESOLUCIÓN)

Dirección cámara 1 → 0x8200_000C

Dirección cámara 2 → 0x8200_010C

Dirección cámara 3 → 0x8200_020C

Contiene la resolución que el usuario desea que tenga la imagen captada.

BIT 0 al BIT 7 → Datos Tamaño Columna Byte Bajo (LOW)

BIT 8 al BIT 15 → Dato Tamaño Columna Byte Alto (HIGH)

BIT 16 al BIT 23 → Dato Tamaño Fila Byte Bajo (LOW)

BIT 24 al BIT 31 → Dato Tamaño Fila Byte Alto (HIGH)

REGISTRO 4 (REGISTRO ZOOM)

Dirección cámara 1 → 0x8200_0010

Dirección cámara 2 → 0x8200_0110

Dirección cámara 3 → 0x8200_0210

Contiene el zoom que el usuario desea que tenga la imagen captada.

BIT 0 al BIT 7 → Dato Inicio Columna Byte Bajo (LOW)

BIT 8 al BIT 15 → Dato Inicio Columna Byte Alto (HIGH)

BIT 16 al BIT 23 → Dato Inicio Fila Byte Bajo (LOW)

BIT 24 al BIT 31 → Dato Inicio Fila Byte Alto (HIGH)

Los registros que contienen BITS sin utilizar quedan reservados para futuras aplicaciones.

A continuación se muestra la estructura de la "PLB_cámara" de forma grafica.

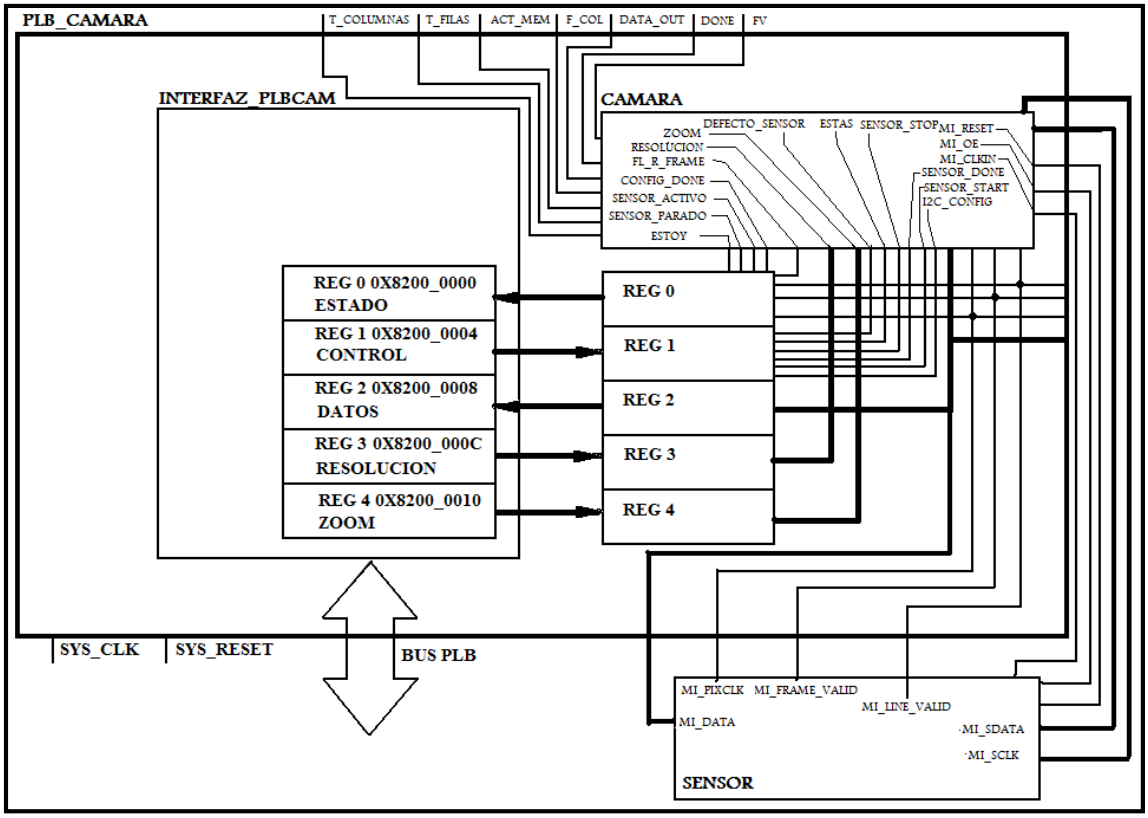


Figura 3.3: Estructura PLB_camara

3.2.1.1 Diseño Interfaz_PLBcam

Para el desarrollo de este módulo, es indispensable hacer un pequeño estudio del funcionamiento del protocolo PLB.

- Protocolo PLB

Descripción:

El Processor Local Bus (PLB) es un bus de alto rendimiento con 64-bits en el bus de direcciones y 128-bits en el bus de datos.

El PLB soporta la transferencia de lectura y escritura entre maestros y esclavos, a través de las señales del PLB.

Cada maestro es añadido al PLB con buses de direcciones, datos de escritura, datos de lectura y señales de control de forma independiente.

El PLB es completamente síncrono con la señal `PLB_clk`, común a todos los dispositivos añadidos al bus.

En la siguiente figura se muestra las conexiones del bus PLB.

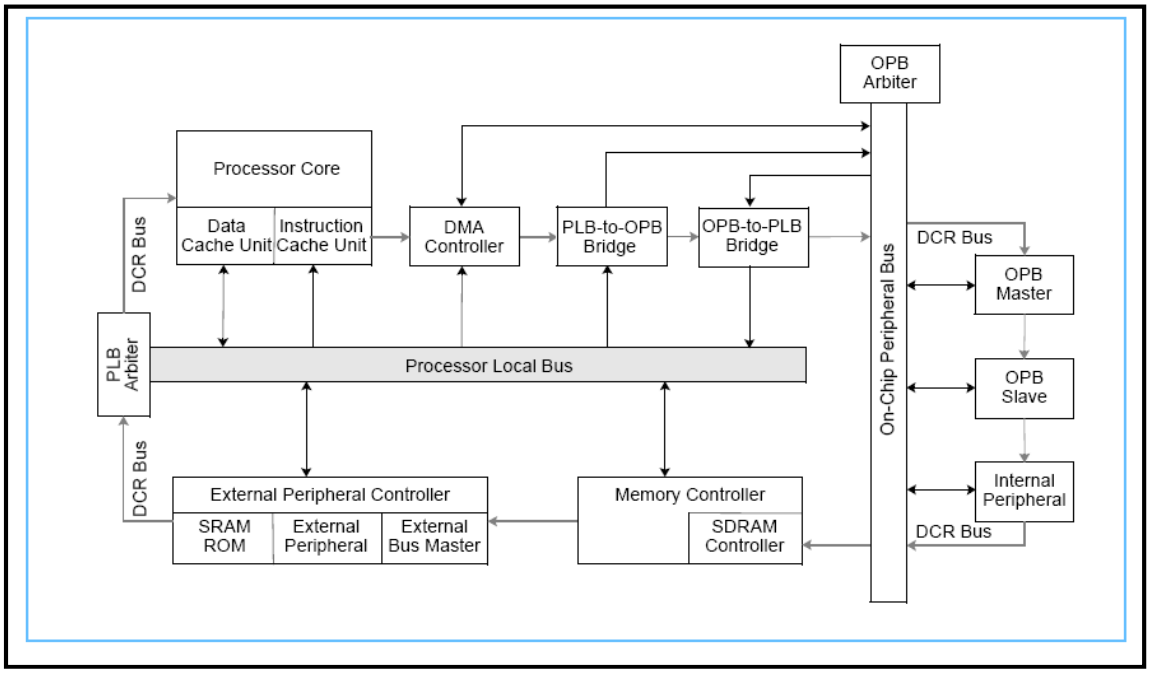


Figura 3.4: Estructura del bus PLB [7]

Hay que destacar que el número de esclavos y maestros añadidos a PLB, afectan proporcionalmente, al rendimiento de las operaciones.

Funcionamiento del protocolo PLB:

La comunicación entre el maestro y el esclavo se compone del ciclo de direccionamiento y el ciclo de datos, como se muestra en la siguiente figura.

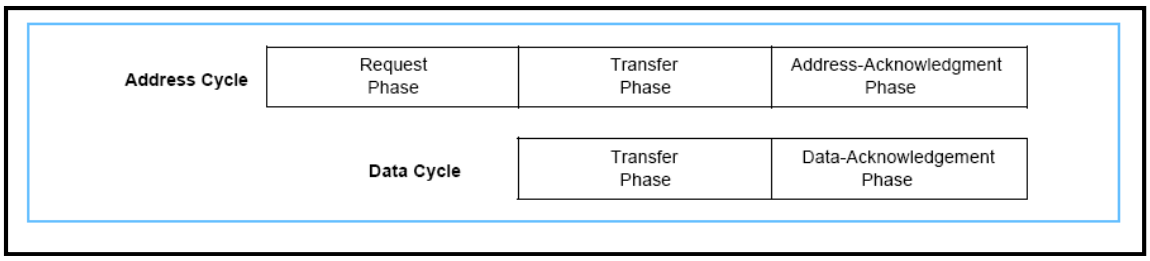


Figura 3.5: Fases de la comunicación del PLB [7]

A su vez el ciclo de direccionamiento se compone de tres fases: petición, transferencia y reconocimiento de dirección.

Petición: Esta fase consiste en que el maestro, solicita el control del bus al árbitro del PLB para que este se lo conceda en el momento que sea preciso.

Transferencia: Una vez el árbitro ha concedido el control del bus al maestro, este muestra en el bus de direcciones del PLB, la dirección del esclavo con el que quiere comunicar, así como los datos y señales que controlan la comunicación.

Reconocimiento de la dirección: Esta fase es en la que el esclavo cuya dirección coincide con la enviada por el maestro, activa la señal indicadora de que esa es su dirección y que está listo para recibir o enviar de datos.

Aquí finaliza el ciclo de direccionamiento, y comienza el ciclo de datos.

El ciclo de datos se compone a su vez de dos fases: transferencia, reconocimiento de dato.

Transferencia: En esta fase se produce la transferencia de datos, del maestro al esclavo en el caso de que se esté realizando la operación de escritura, y de esclavo a maestro, en el caso contrario.

Reconocimiento de dato: Esta fase es donde el esclavo indica al maestro que se ha completado la lectura o escritura de datos, y consecuentemente el maestro, deja libre el bus.

Y con esta última fase se concluye la comunicación.

Señales PLB:

Nota: Las señales de PLB aquí descritas serán las utilizadas para el buen funcionamiento de este proyecto. Para más información sobre el bus PLB, protocolos, funcionamiento, señales, etc. Consultar el documento plbIBM disponible en las referencias de este documento [7].

Señales del sistema:

`Sys_plbClk` → Esta señal se conecta a todos los dispositivos añadidos al bus PLB, y todas sus señales de entrada y salida se modifican en el flanco de subida de esta señal.

`Sys_plbReset` → Esta señal se activa para la inicialización del bus, así como para la parada de cualquier comunicación que este en proceso.

Señales de arbitraje:

Nota: Las señales que empiezan por Mn_... serán señales de salida del maestro.

Las señales que empiezan por Sl_... serán señales de salida del esclavo.

Las señales que empiezan por PLB_... serán señales de entrada o salida del árbitro del bus.

Señales que utiliza el árbitro del bus PLB para el control de la comunicación.

En este proyecto solo existe un maestro, por lo que muchas de las señales de este tipo no son necesarias.

`Mn_Request` → Esta señal es activada por el maestro, para indicarle al árbitro que solicita el control del bus PLB, y así iniciar una comunicación con el esclavo deseado.

`PLB_PAValid` → Esta señal se activa cuando la dirección enviada por el maestro, es válida, es decir, la dirección se corresponde con la dirección del algún esclavo conectado al PLB.

La señal antes mencionada permanece activada hasta que algunas de estas señales se activen:

`Sl_AddrAck, Sl_rearbitre`, que el maestro aborte la comunicación ó el árbitro la finalice por superación del límite de tiempo del proceso de comunicación.

`Sl_Wait` → Esta señal es activada por el esclavo cuando la dirección enviada por el maestro se corresponde con la suya, es decir, sea válida.

`Sl_AddrAck, PLB_MnAddrAck` → Esta señal es activada cuando el esclavo reconozca la dirección enviada como suya, y guarda los datos y valores de las señales conectadas al bus.

Nota: `Sl_AddrAck` y `PLB_MnAddrAck` son la misma señal sólo que la primera es la salida del dispositivo esclavo y la primera es la entrada al dispositivo árbitro.

`Mn_Abort, PLB_abort` → Esta señal se activa cuando la comunicación entre maestro y esclavo es abortada, perdiendo el esclavo todos los datos transferidos.

Señales de estado del bus:

Estas señales no se utilizan en este proyecto.

Señales de caracterización de la transferencia:

`Mn_RNW, PLB_RNW` → Esta señal indica si la operación de comunicación es tipo escritura o lectura.

`Mn_ABus(0-31),PLB_ABus(0-31)` → Este señal de 32 bits es el bus de direcciones del PLB.

Señales del bus de datos de lectura:

`Sl_rdDBus,PLB_MnRdDBus` → Este es el bus de datos de lectura.

`Sl_rdDAck, PLB_MnRdDAck` → Esta señal se activa indicando que el dato ha sido introducido en el bus `Sl_rdDBus` por el esclavo.

`Sl_rdComp` → Esta señal indica que la transferencia del dato ha sido completada.

Señales del bus de datos de escritura:

`Mn_wrDBus, PLB_wrDBus` → Este es el bus de datos escritura.

`Sl_wrDAck, PLB_MnWrDAck` → Esta señal la activa el esclavo indicando que el dato que ha sido introducido en el bus `Sl_wrDBus,` ha sido registrado.

`Sl_wrComp` → Esta señal es activada por el esclavo para indicar que la transferencia del dato ha sido completada.

Cronogramas de operaciones escritura y lectura:

Lectura:

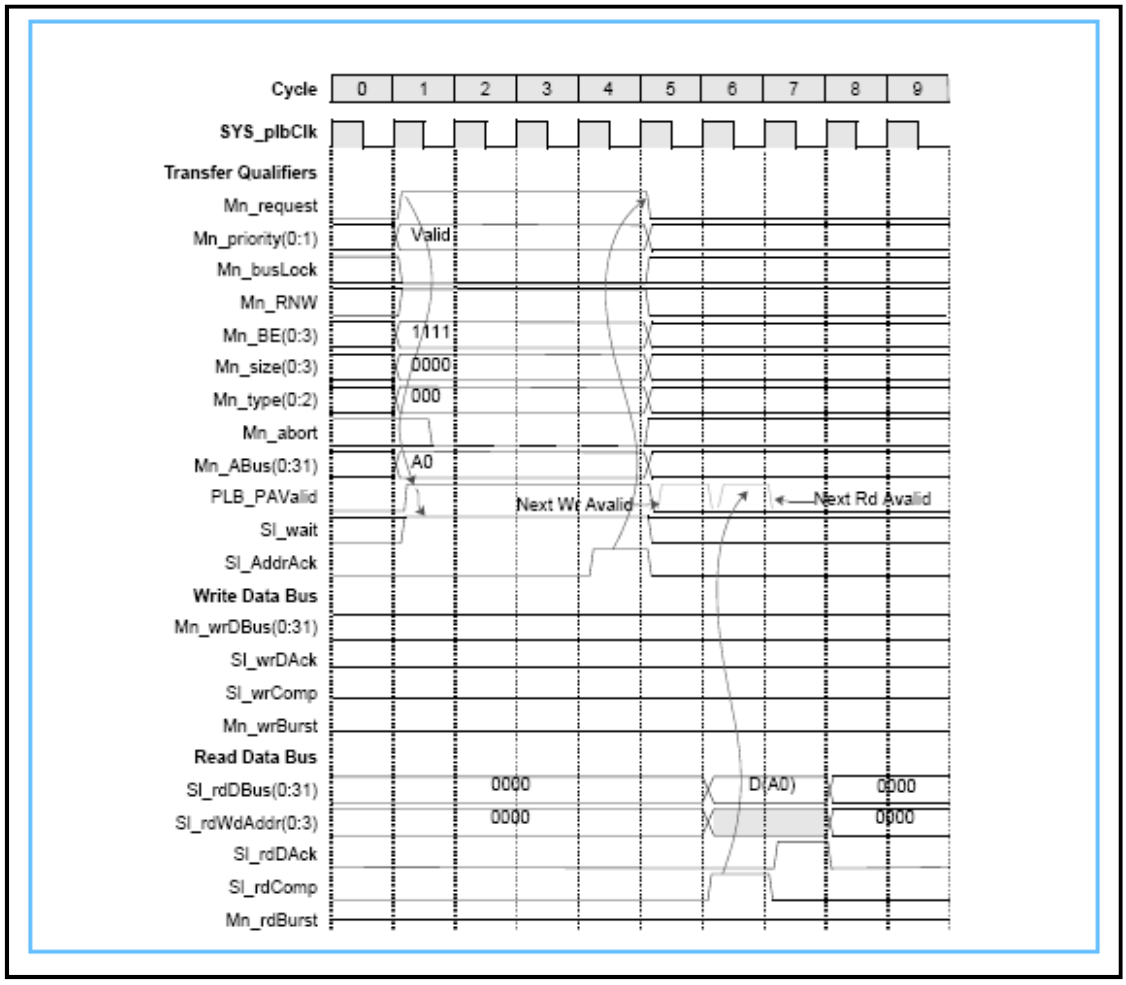


Figura 3.6: Proceso lectura del PLB [7]

Escritura:

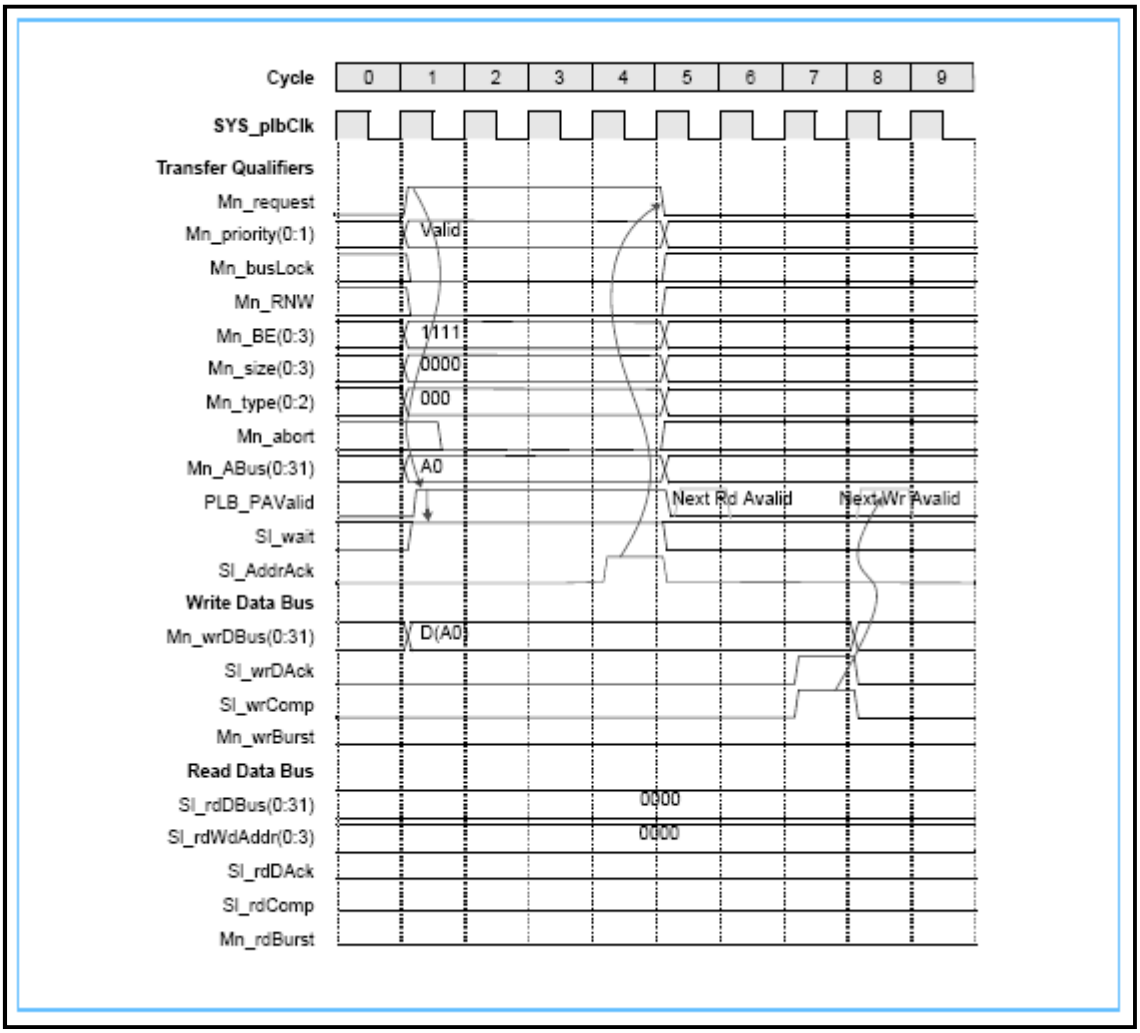


Figura 3.7: Proceso de escritura del PLB [7]

- Implementación interfaz_PLBcam

Los puertos que forman parte de este módulo son los siguientes:

```
port
(

  SPLB_Clk                    : in  std_logic;
  SPLB_Rst                    : in  std_logic;
  PLB_ABus                    : in  std_logic_vector(0 to 31);
  PLB_UABus                   : in  std_logic_vector(0 to 31);
  PLB_PAValid                 : in  std_logic;
  PLB_SAValid                 : in  std_logic;
  PLB_rdPrim                  : in  std_logic;
  PLB_wrPrim                  : in  std_logic;
  PLB_masterID                : in  std_logic_vector(0 to C_SPLB_MID_WIDTH-1);
  PLB_abort                   : in  std_logic;
  PLB_busLock                 : in  std_logic;
  PLB_RNW                     : in  std_logic;
  PLB_BE                      : in  std_logic_vector(0 to C_SPLB_DWIDTH/8-1);
  PLB_MSize                   : in  std_logic_vector(0 to 1);
  PLB_size                    : in  std_logic_vector(0 to 3);
  PLB_type                    : in  std_logic_vector(0 to 2);
  PLB_lockErr                 : in  std_logic;
  PLB_wrDBus                  : in  std_logic_vector(0 to C_SPLB_DWIDTH-1);
  PLB_rdBurst                 : in  std_logic;
  PLB_wrPendReq               : in  std_logic;
  PLB_rdPendReq               : in  std_logic;
  PLB_wrPendPri               : in  std_logic_vector(0 to 1);
  PLB_rdPendPri               : in  std_logic_vector(0 to 1);
  PLB_reqPri                  : in  std_logic_vector(0 to 1);
  PLB_TAttribute              : in  std_logic_vector(0 to 15);
  Sl_addrAck                  : out std_logic;
  Sl_SSize                    : out std_logic_vector(0 to 1);
  Sl_wait                     : out std_logic;
  Sl_rearbitrate              : out std_logic;
  Sl_wrDAck                   : out std_logic;
  Sl_wrComp                   : out std_logic;
  Sl_wrBTerm                  : out std_logic;
  Sl_rdDBus                   : out std_logic_vector(0 to C_SPLB_DWIDTH-1);
  Sl_rdWdAddr                 : out std_logic_vector(0 to 3);
  Sl_rdDAck                   : out std_logic;
  Sl_rdComp                   : out std_logic;
  Sl_rdBTerm                  : out std_logic;
  Sl_MBusy                    : out std_logic_vector(0 to C_SPLB_NUM_MASTERS-1);
  Sl_MWrErr                   : out std_logic_vector(0 to C_SPLB_NUM_MASTERS-1);
  Sl_MRdErr                   : out std_logic_vector(0 to C_SPLB_NUM_MASTERS-1);
  Sl_MIRQ                     : out std_logic_vector(0 to C_SPLB_NUM_MASTERS-1);
  REGISTRO0                   : in std_logic_vector(0 to 31);
  REGISTRO1                   : out std_logic_vector(0 to 31);
  REGISTRO2                   : in std_logic_vector(0 to 31);
  REGISTRO3                   : out std_logic_vector(0 to 31);
  REGISTRO4                   : out std_logic_vector(0 to 31)

);
```

Figura 3.8: Puertos de la interfaz_PLBcam

Se observa que hay muchos puertos dedicados al bus PLB, muchas de ellas no se utilizan en este módulo, en cambio, hay que presentarlas porque sino la herramienta de Xilinx que más adelante conecta la cámara estéreo con Microblaze, da un error exponiendo que faltan puertos del bus PLB, aunque estos no se utilicen.

“Interfaz_PLBcam” tiene el cometido de gestionar la comunicación con el bus PLB, es decir, activa y desactiva las señales en el instante de tiempo apropiado, así como de presentar o guardar los datos de los buses de lectura o escritura, respectivamente.

Dentro de este módulo, se introducen los registros donde se almacenan los datos de control o datos de estado de la cámara.

En la siguiente figura se muestran los bloques que componen la “Interfaz_PLBcam”.

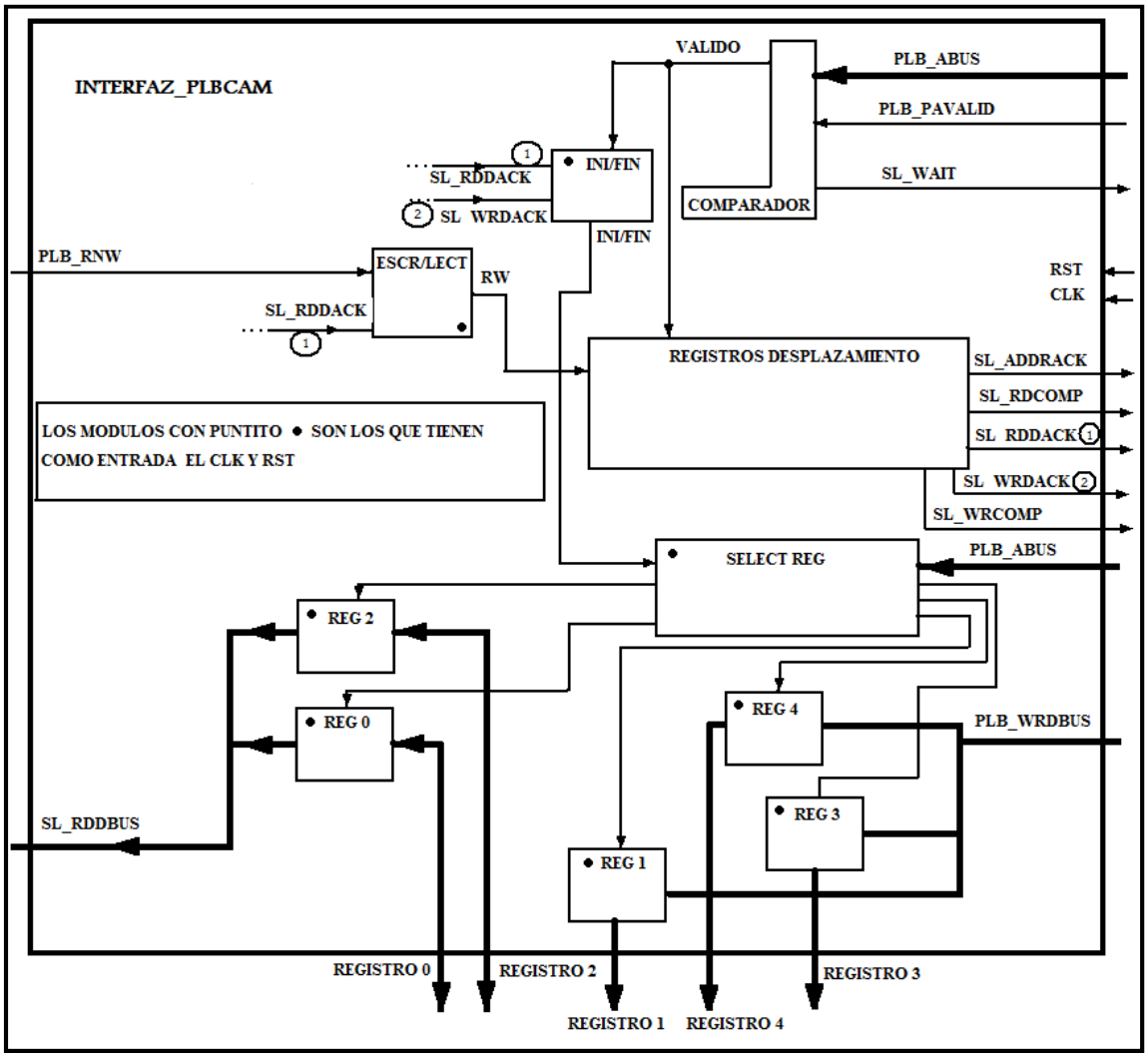


Figura 3.9: Estructura de la interfaz_PLBcam

Se compone de un módulo “COMPARADOR” el cual recibe la dirección enviada por el maestro, y detecta si dicha dirección se encuentra en el rango de direcciones del esclavo.

Si es así, el “COMPARADOR” activa la señal `Sl_wait`, y da permiso al resto de módulos de la interfaz para que ejecuten su función.

Los módulos “INI/FIN” y “ESCR/LECT” reconocen si la comunicación ha comenzado o finalizado y si se trata de una comunicación de escritura o lectura, respectivamente.

El módulo “REGISTRO DESPLAZAMIENTO” está compuesto de varios registros de desplazamiento internos, con la función de activar y desactivar las señales en los instantes de tiempo establecidos por el protocolo PLB.

El “SELECT REG” activa la señal de `Enable` de los registros, dependiendo de la dirección enviada por el maestro.

Y por último los registros de datos, el registro 0 y el registro 2, son de lectura y los registros 1, 3 y 4 son de escritura.

3.2.1.2 Diseño Cámara

Para la realización del diseño de la “cámara” se realizan modificaciones en los componentes de la cámara, realizada en el proyecto *“Diseño y control de una cámara de video digital”* de Ruiz Salcedo, Carlos.

De forma muy resumida, la cámara del proyecto *“Diseño y control de una cámara de video digital”* funciona de la siguiente manera.

En la cámara, se pueden configurar los registros internos del sensor de imagen para obtener así la resolución deseada.

La configuración de estos registros, se realiza a través de un bus I2C. Dicho bus I2C se controla con el módulo “control_I2C” que se encarga de leer una “ROM” con los datos introducidos a priori por el programador, y llevarlos a los registros internos del sensor [2].

Una vez configurado el sensor, se inicia la captura de datos, los cuales tienen un tamaño de 10 bits y que debido a las limitaciones de memoria, se guardan como un solo bit, siendo la imagen en blanco y negro. Posteriormente, esta imagen es leída desde memoria y mostrada por una VGA [2].

Todo este proceso se hace en una SPARTAN3-E con una frecuencia de reloj de 50Mhz.

Para una información más detallada consultar el proyecto *“Diseño y control de una cámara de video digital”* de Ruiz Salcedo, Carlos.

- Adaptación de la cámara

La cámara está estructurada de la siguiente manera:

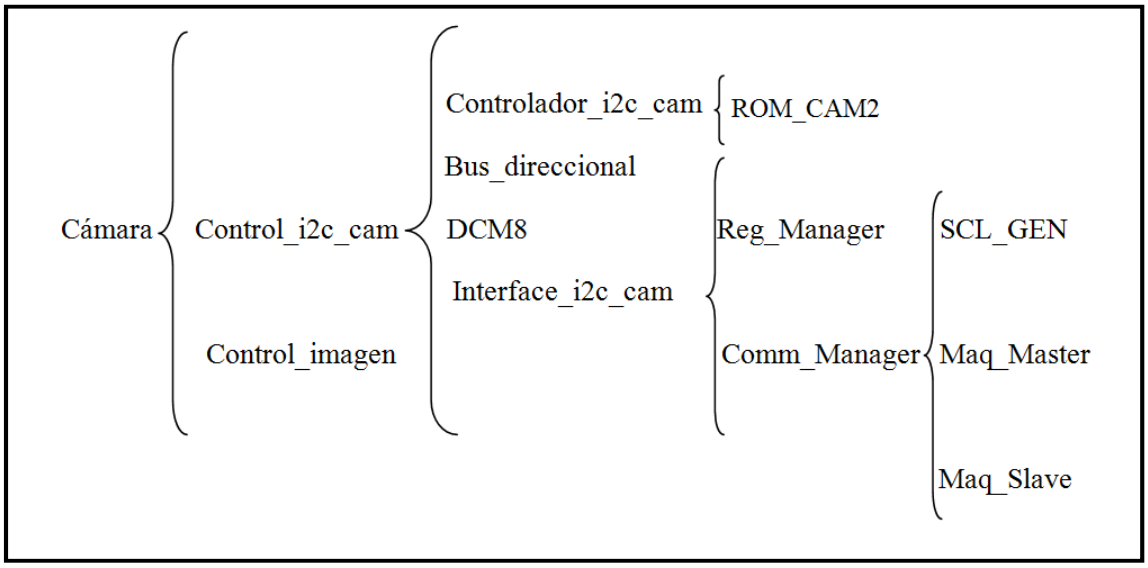


Figura 3.10: Esquema del módulo cámara

Esta estructura es muy similar a la expuesta en el proyecto "*Diseño y control de una cámara de video digital*", en la cual el módulo "interface_i2c_cam" pertenece al proyecto "*Diseño de interfaces serie integrados: Uart y Bus I2C*".

Eliminación y modificación de módulos

La primera de las diferencias es la eliminación del módulo de control de VGA porque en la nueva cámara no existirá esta opción, ya que su cometido solo es de almacenamiento.

El módulo "Control_i2c_cam", no se le realiza cambios, a excepción de los puertos. A este módulo se le añaden los puertos resolución y zoom de tamaño 32 bits, que van conectados a la nueva "ROM_i2c", cuyo nuevo nombre es "ROM_CAM2", y que sirve para que el usuario pueda configurar el sensor, sin tener que modificar el código interno.

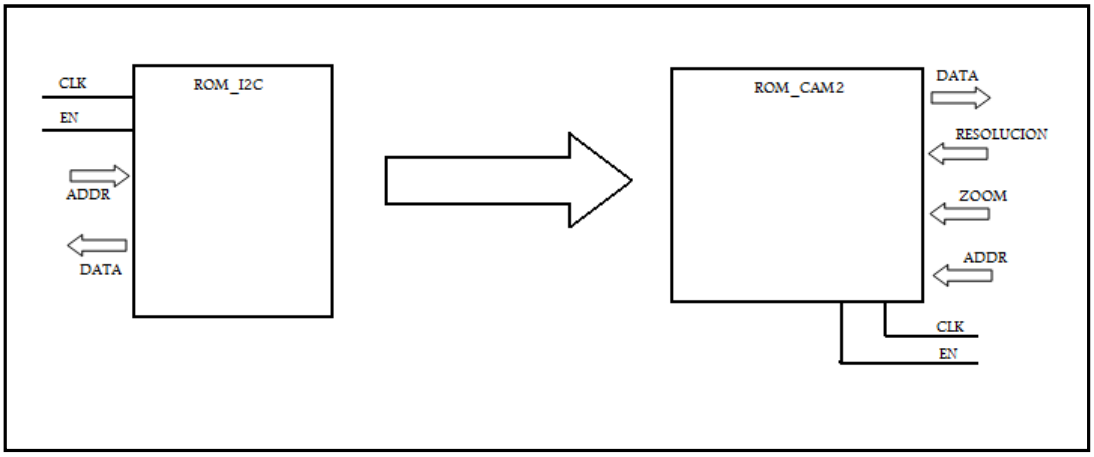


Figura 3.11: Evolución de la estructura ROM_i2c a ROM_cam2

Se diseña una nueva "ROM_i2c", la cual se le añaden los puertos de resolución y zoom antes mencionados y cuyos parámetros que afectan a los registros 0X01, 0X02, 0X03 y 0X04 (registros que configuran la resolución y el zoom del sensor) son dinámicos.

Las imágenes no se introducen en la SRAM o BLOCKRAM, cuyos tamaños son limitados, y se procede a guardarlos en una memoria externa RAM creada gracias a la herramienta `Core Generator` donde se guardan los datos con un tamaño de 8 bits y que más adelante se explica su funcionamiento (El tamaño de la memoria externa depende de la disponible en la placa de Xilinx). Este nuevo módulo está compuesto por unos contadores de columnas y filas de la imagen, así como una máquina de estados que controla los contadores.

Se diseña un nuevo módulo control de imágenes que aporta las señales necesarias para que el módulo "control_imagen_ext" (que se menciona más adelante), funcione correctamente.

A continuación, se modifica el módulo "DCM8". Su cometido es adaptar la frecuencia de la placa SPARTAN-3E de 50Mhz, a la frecuencia de 8Mhz (DCM8), para el buen funcionamiento del bus I2C. En este caso se modifica los valores de los parámetros internos, para que adapten la frecuencia de la placa (100 MHz, Virtex4), como sigue:

MODULO DCM8

En el módulo "DCM8" se cambia el valor de las señales:

- `CLKDV_DIVIDE` es el valor que divide la frecuencia de entrada. Por ello si se quiere obtener a la salida un valor de 8 MHz, se divide la frecuencia por un valor igual a 12.000.000.

- `CLKIN-PERIOD` indica el periodo de la señal de entrada, por tanto, para una frecuencia de 100 MHz el periodo tiene un valor de 10 ns, luego al parámetro `CLKIN-PERIOD` se le da un valor igual a 10.000.000.

La cámara diseñada por el proyecto "*Diseño y control de una cámara de video digital*", tiene un módulo interior llamado "DCM48", que realiza la misma función que el módulo que se ha descrito anteriormente, con la diferencia que a su salida proporciona una señal de 48 MHz, en vez 8 MHz. Este módulo se saca fuera de la cámara, y se instancia como un nuevo módulo del sistema cámara estéreo porque esta señal de reloj la utiliza también el módulo "control_imagenes_ext".

Por último, se modifica algunas señales de control de la cámara, y se añaden señales de estado de la misma. Estas señales son las que se comunican con el microprocesador.

- Función de los puertos que formarán la cámara

Los puertos de este módulo "cámara" son los siguientes:

```
    port
      (
        sys_clk                         : IN    std_logic;
        sys_reset                       : IN    std_logic;
        clk_48mhz                       : IN    std_logic;
        mi_data                         : IN    std_logic_vector (9 downto 0);
        mi_line_valid                   : IN    std_logic;
        mi_frame_valid                  : IN    std_logic;
        mi_pixclk                       : IN    std_logic;
        mi_clkin                        : OUT   std_logic;
        mi_OE                           : OUT   std_logic;
        mi_RESET                        : OUT   std_logic;
        sensor_start                    : IN    std_logic;
        sensor_stop                     : IN    std_logic;
        sensor_done                     : OUT   std_logic;
        config_done                     : OUT   std_logic;
        sensor_activo                   : OUT   std_logic;
        sensor_parado                   : OUT   std_logic;
        estas                           : IN    std_logic;
        estoy                           : OUT   std_logic;
        defecto_sensor                  : IN    std_logic;
        mi_sclk_i                       : IN    std_logic;
        mi_sclk_o                       : OUT   std_logic;
        mi_sdata_i                      : IN    std_logic;
        mi_sdata_o                      : OUT   std_logic;
        i2c_config                      : IN    std_logic;
        Resolucion                      : IN    std_logic_vector (31 downto 0);
        Zoom                            : IN    std_logic_vector (31 downto 0);
        t_columnas                      : OUT   integer range 0 to 800;
        t_filas                         : OUT   integer range 0 to 600;
        act_mem                         : OUT   std_logic;
        f_col                           : OUT   std_logic;
        data_out                        : OUT   std_logic_vector(9 downto 0);
        done                            : OUT   std_logic;
        fv                              : OUT   std_logic
      );
    end camara;
```

Figura 3.12: Puertos del modulo cámara

Señales genéricas

`sys_clk` → señal de reloj de 100 MHz

`sys_reset` → señal de `reset` de todo el sistema.

`clk_48mhz` → señal de reloj de 48 MHz

Señales de control para el micro-controlador

`sensor_start` → señal para iniciar la captura de imágenes del sensor.

`sensor_stop` → señal para parar la captura de imágenes.

`estas` → señal que indica que está conectado el software de la cámara.

`defecto_sensor` → señal para configurar el sensor con sus valores por defecto.

`i2c_config` → señal para iniciar la configuración del sensor.

Señales de estado para el micro-controlador

`sensor_done` → señal que indica que el sensor ha capturado una imagen.

`config_done` → señal que indica que el sensor ha finalizado su configuración.

`sensor_activo` → señal que indica que el sensor está capturando imágenes.

`sensor_parado` → señal que indica que el sensor no está capturando imágenes.

`estoy` → señal que indica que está conectado el software de la cámara.

Señales de datos para el micro-controlador

`Resolución` → señal que indica al sensor con que valor de resolución debe configurar al sensor.

`Zoom` → señal que indica al sensor con que valor de zoom debe configurar al sensor.

`t_columnas` → señal que indica el número de columnas que tiene la última imagen captada por el sensor.

`t_filas` → señal que indica el número de filas que tiene la última imagen captada por el sensor.

Estas dos últimas señales ayudan al microprocesador a leer las imágenes de memoria.

Señales de estado para el "control_imagen_ext"

`act_mem` → señal que indica que el sensor esta capturando imágenes.

`f_col` → señal que indica que el sensor ha terminado una fila de la imagen.

`data_out` → señal con el valor del pixel que devuelve el sensor (10 bits).

`Done` → señal que indica que el sensor ha finalizado una imagen.

`Fv` → señal que indica que ha habido un flanco de subida de la señal `frame_valid` del sensor.

Los puertos que no se han mencionado pertenecen a señales provenientes del sensor y que están explicadas en detalle en el proyecto "*Diseño y control de una cámara de video digital*".

Finalizado con el diseño del módulo "interfaz_PLBcam" y del módulo "cámara" se unen para formar la "PLB_camara" como se muestra en la figura 3.1 (la del PLB_cam).

3.2.2 Diseño control_imagenes_ext

Este módulo tiene el cometido de gestionar el almacenamiento de las imágenes de las dos o más cámaras conectadas al sistema cámara estéreo, en el cual existe un proceso de escritura y otro de lectura.

En el proceso de escritura de las imágenes, es decir, cuando el usuario activa la cámara estéreo, el "control_imagenes_ext" inicia un análisis para saber que cámaras están activadas, ya que el usuario puede elegir el número de imágenes que quiere captar, así como, con que cámaras captarlas.

Después de este análisis comienza a guardar la primera imagen en el momento preciso y en su memoria correspondiente, es decir, si se capta la imagen con la "PLB_camara" número 1, la imagen se almacena en el módulo "nimagen1".

Para este proceso de escritura se desarrolla unas señales provenientes de cada "PLB_camara" que sirven para el control del almacenamiento de las imágenes.

Estas señales son:

`Fv` → señal que indica que la "PLB_camara", esta activada.

`f_col` → señal que controla la inicialización del contador columnas.

`act_mem` → señal que indica que se puede comenzar a almacenar la imagen.

`Done` → señal que indica que se ha completado la imagen, y por tanto su almacenamiento.

`f_img` → señal que indica que se han almacenado todas las imágenes. Esta señal esta directamente conectada al registro de estado de la "PLB_camara".

Todo el proceso anteriormente descrito se puede realizar gracias a que este módulo se compone por los siguientes elementos, un contador de página, un contador de filas, un contador de columnas y una máquina de estados, la cual se encarga de almacenar en el orden y lugar correcto las imágenes. En la siguiente figura se describe gráficamente el funcionamiento de la máquina de estados:

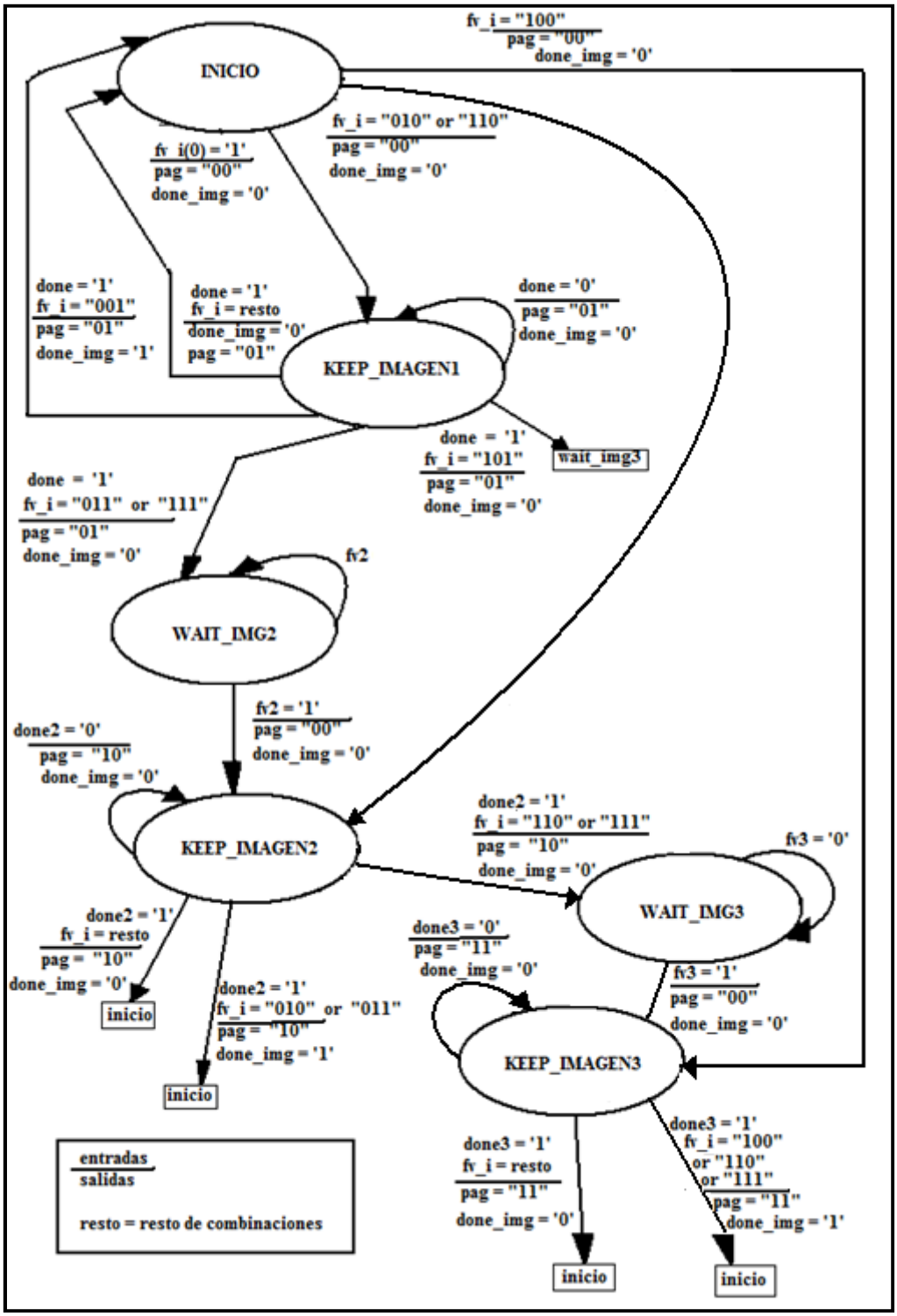


Figura 3.13: Diagrama de estados de la máquina de estados control_imagenes_ext

Keep_imagen1 → manda almacenar los datos de la imagen 1.

Keep_imagen2 → manda almacenar los datos de la imagen 2.

Keep_imagen3 → manda almacenar los datos de la imagen 3.

Wait_img2 → espera a que comience la imagen 2.

Wait_img3 → espera a que comience la imagen3.

Para el proceso de lectura de las imágenes, lo único que se realiza es una conexión directa con el bus PLB. Cuando el usuario desde la interfaz manda al controlador leer las imágenes, éste envía por el bus, las direcciones correspondientes a los módulos donde están almacenadas las imágenes. El "control_imagenes_ext", recoge los datos de los módulos de "nimagen", y se los envía al micro-controlador por el bus de datos del PLB. Este proceso esta desactivado si las "PLB_camaras" están en modo escritura y viceversa.

A continuación, se describe las señales que están conectadas a los módulos de almacenamiento de las imágenes ("nimagen1", "nimagen2" y "nimagen3").

`addrs` → esta señal es de un tamaño de 21 bits, que se descompone de la siguiente manera, los bits 0 al 1, leyendo la señal de izquierda a derecha, son los que definen la página, es decir, son los que definen si se lé ó se escribe la "nimagen1", "nimagen2" o "nimagen3". Los restantes bits indican la dirección donde está almacenada la imagen.

`r_w` → esta señal indica si se quiere leer la imagen `r_w` = 1 ó se quiere escribir `r_w` = 0.

Por último se muestra los puertos que forman el módulo "control_imagen_ext".

```
port
(

  clk48mhz                          : in std_logic;
  reset_in                          : in std_logic;

  act_mem1                          : in std_logic;
  f_col1                            : in std_logic;
  done1                             : in std_logic;
  fv1                               : in std_logic;
  data_pix1                         : in std_logic_vector(6 downto 0);
  f_img1                            : out std_logic;
  read1                             : in  std_logic;
  act_mem2                          : in std_logic;
  f_col2                            : in std_logic;
  done2                             : in std_logic;
  fv2                               : in std_logic;
  data_pix2                         : in std_logic_vector(6 downto 0);
  f_img2                            : out std_logic;
  read2                             : in  std_logic;
  act_mem3                          : in std_logic;
  f_col3                            : in std_logic;
  done3                             : in std_logic;
  fv3                               : in std_logic;
  data_pix3                         : in std_logic_vector(6 downto 0);
  f_img3                            : out std_logic;
  read3                             : in  std_logic;
  addrs                             : out std_logic_vector(20 downto 0);
  r_w                               : out std_logic;
    •
    •
    •
);
```

Figura 3.14: Puertos del modulo control_imagenes_ext

Los puntos suspensivos indican los puertos conectados al PLB, que por su tamaño se han obviado en la figura.

3.2.3 Diseño NimagenN

Se diseña un módulo “nimagen” que tiene el cometido de realizar la conversión de los 10 bits que manda el sensor de imagen, definiendo el color del pixel, a 8 bits.

Como se ha dicho este módulo, se multiplica por dos o tres veces, dependiendo del número de imágenes que se quieran utilizar, formando los módulos “nimagen1”, “nimagen2” y “nimagen3”.

Este módulo recibe las direcciones por parte del “control_imagenes_ext”, a través del puerto `addrs` donde los dos primeros bits de la dirección indican cual es la imagen que se quiere leer o escribir, activando el pin `ENA` de la memoria correspondiente. Los datos de la imagen se obtienen de la “PLB_camara”, a través del puerto `data_in`, al igual que se puede sacar la información de la memoria con el puerto `data_out`. Las ordenes de lectura o escritura que manda el “control_imagenes_ext”, se reciben a través de los puertos `read` y `write`.

Este módulo recibe también la señal de resetear la RAM, y este se lo indica a la memoria.

Este módulo se conecta al módulo de la memoria que en este caso se llama “mem_800x600_8bits”, identificando el tamaño de la memoria, la cual puede ser modificada dependiendo del tamaño disponible en la placa.

En la siguiente figura se muestra la estructura de este módulo.

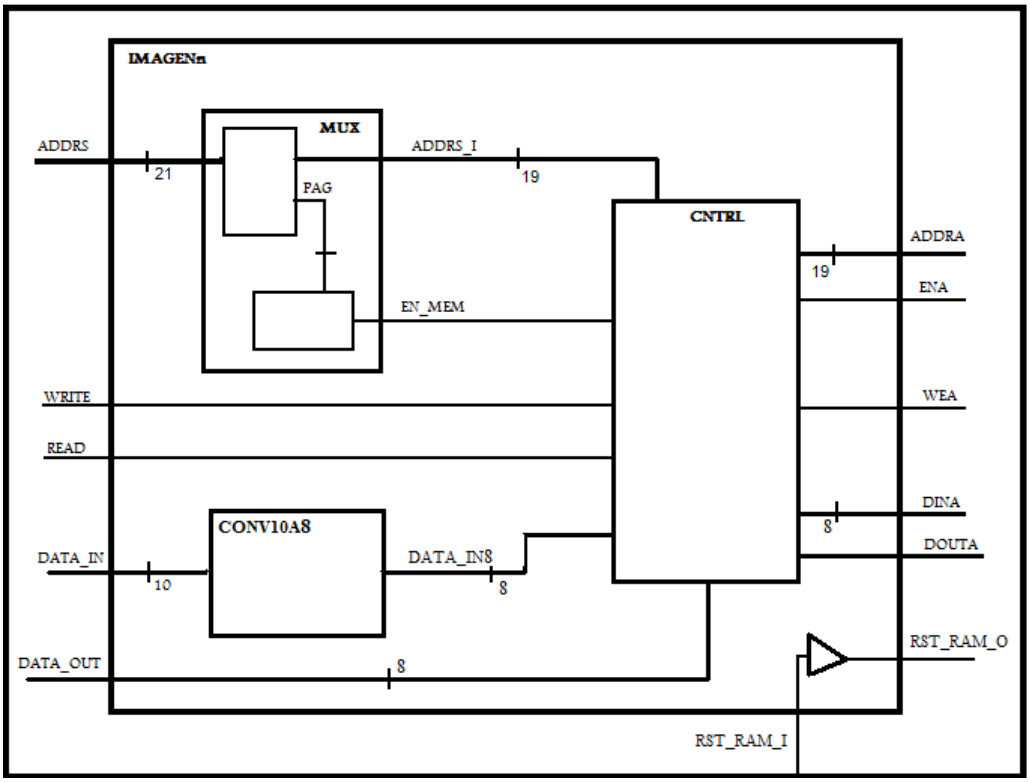


Figura 3.15: Estructura del modulo nimagen

Por tanto los puertos que componen este módulo son los que se muestra en la siguiente figura:

```
port
(
  sys_clk                          : in  std_logic;
  addrs                            : in  std_logic_vector (20 downto 0);
  data_in                          : in  std_logic_vector (9 downto 0);
  data_out                         : out std_logic_vector (7 downto 0);
  read                             : in  std_logic;
  write                            : in  std_logic;
  rst_ram_I                        : in std_logic;
  rst_ram_O                        : out std_logic;
  dina                             : out std_logic_vector(7 downto 0);
  addra                            : out std_logic_vector(18 downto 0);
  ena                              : out std_logic;
  wea                              : out std_logic_vector(0 downto 0);
  douta                            : in  std_logic_vector(7 downto 0)

);
```

Figura 3.16: Puertos del modulo nimagen

3.2.4 Diseño de la memoria externa

Para este módulo se ha realizado el diseño de una memoria RAM externa. Para su diseño se utiliza la herramienta disponible en Xilinx ISE, llamada `Core Generator` [10] y [11]. El tamaño de la memoria externa se encuentra limitada por la FPGA. En el mercado existen placas de Xilinx que disponen de un gran tamaño de memoria y que son compatibles con la cámara estéreo diseñada en este proyecto.

Nota: Se crearán dos memorias de tamaños diferentes. Una de ellas será "mem_800x600_8bits" para las simulaciones. La otra será la "mem_320x240_8bits" para la Virtex4 (Ml402). Todo esto es debido a las limitaciones que tiene la FPGA teniendo que reducir el tamaño de las memorias para poder almacenar tres imágenes.

Esta herramienta se ejecuta yendo en Windows vista a `Inicio → Todos los programas → Xilinx ISE Design Suite 12.2 → ISE → Accessories → CORE Generator`.

Se realiza los siguientes pasos:

Paso 1: Bienvenida

En la primera pantalla que aparece tras ejecutar `CORE Generator`

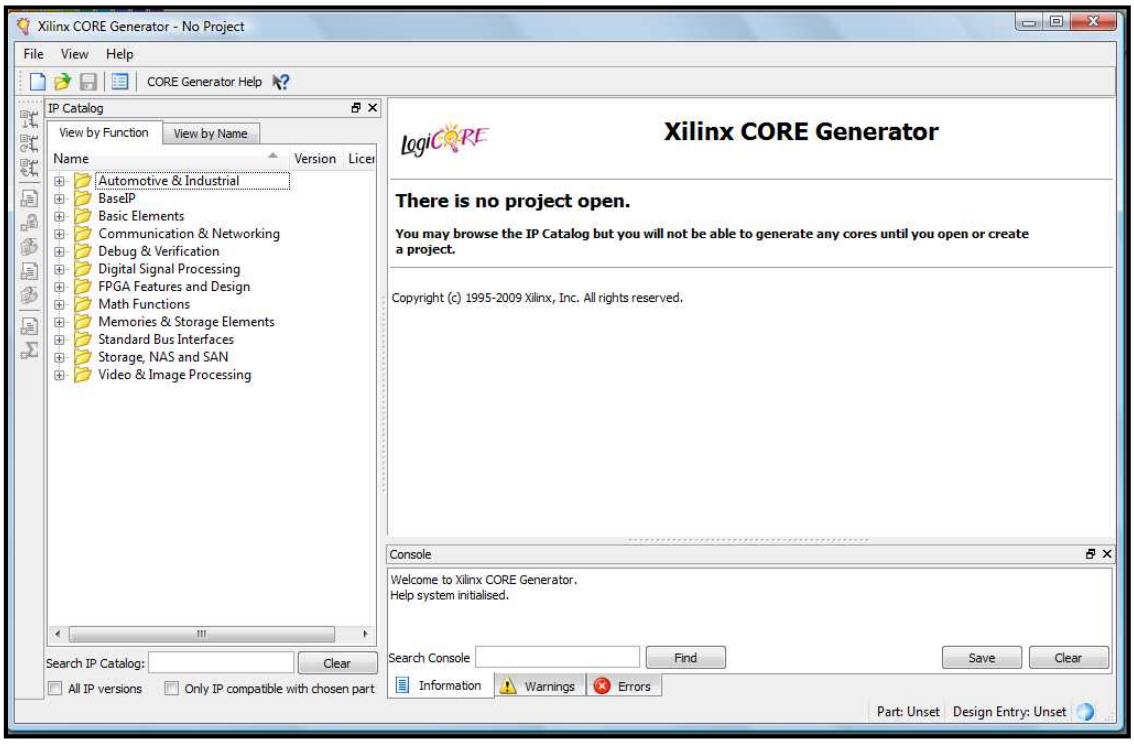


Figura 3.17: Ventana Bienvenida Core Generator

Paso 2: Creación del proyecto

Se pincha en la opción `File` → `new Project`.

Paso 3: Ubicación del proyecto

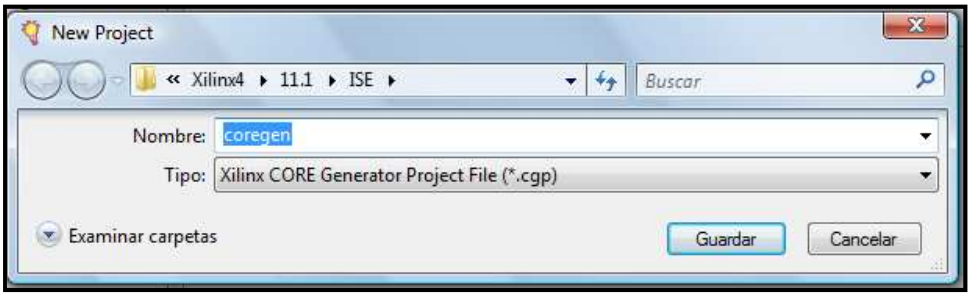


Figura 3.18: Ventana de ubicación del proyecto

Paso 4: Selección de la placa de Xilinx

Donde se elige las opciones indicadas en la figura siguiente.

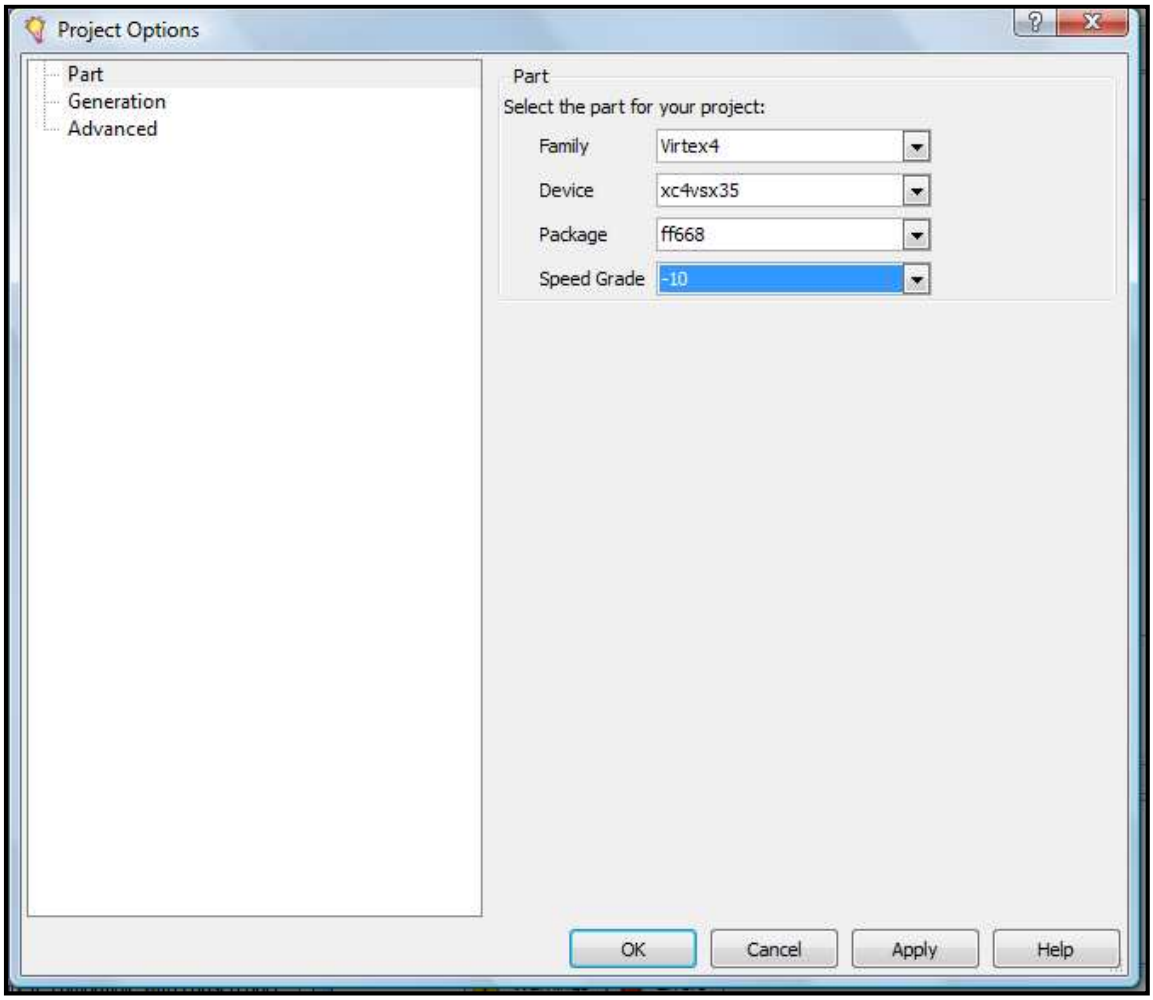


Figura 3.19: Ventana de selección de placa Xilinx

Paso 5: Selección de la `Block Memory Generator`

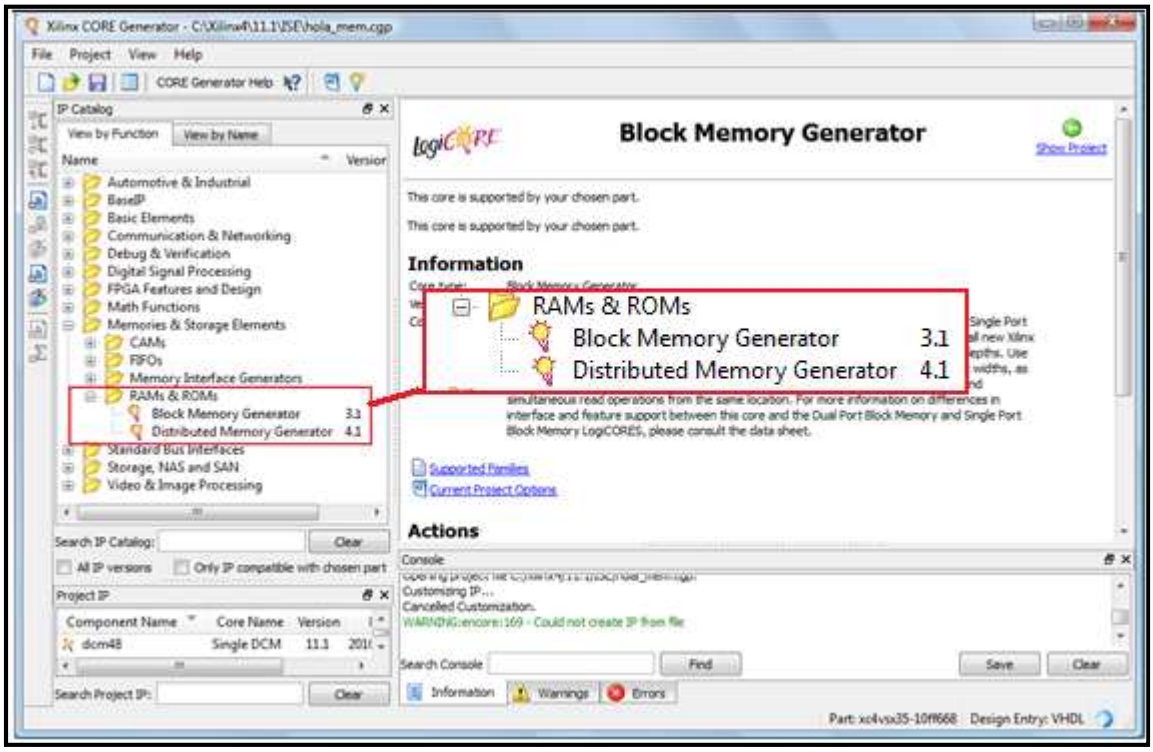


Figura 3.20: Ventana selección de la Block Memory Generator

En esta ventana se pincha en la carpeta `RAMs & ROMs`, y en el menú que se despliega se selecciona `Block Memory Generator 3.1`. Se realiza "doble click" en el elemento seleccionado para comenzar el diseño de la memoria.

Paso 6: Diseño de la memoria

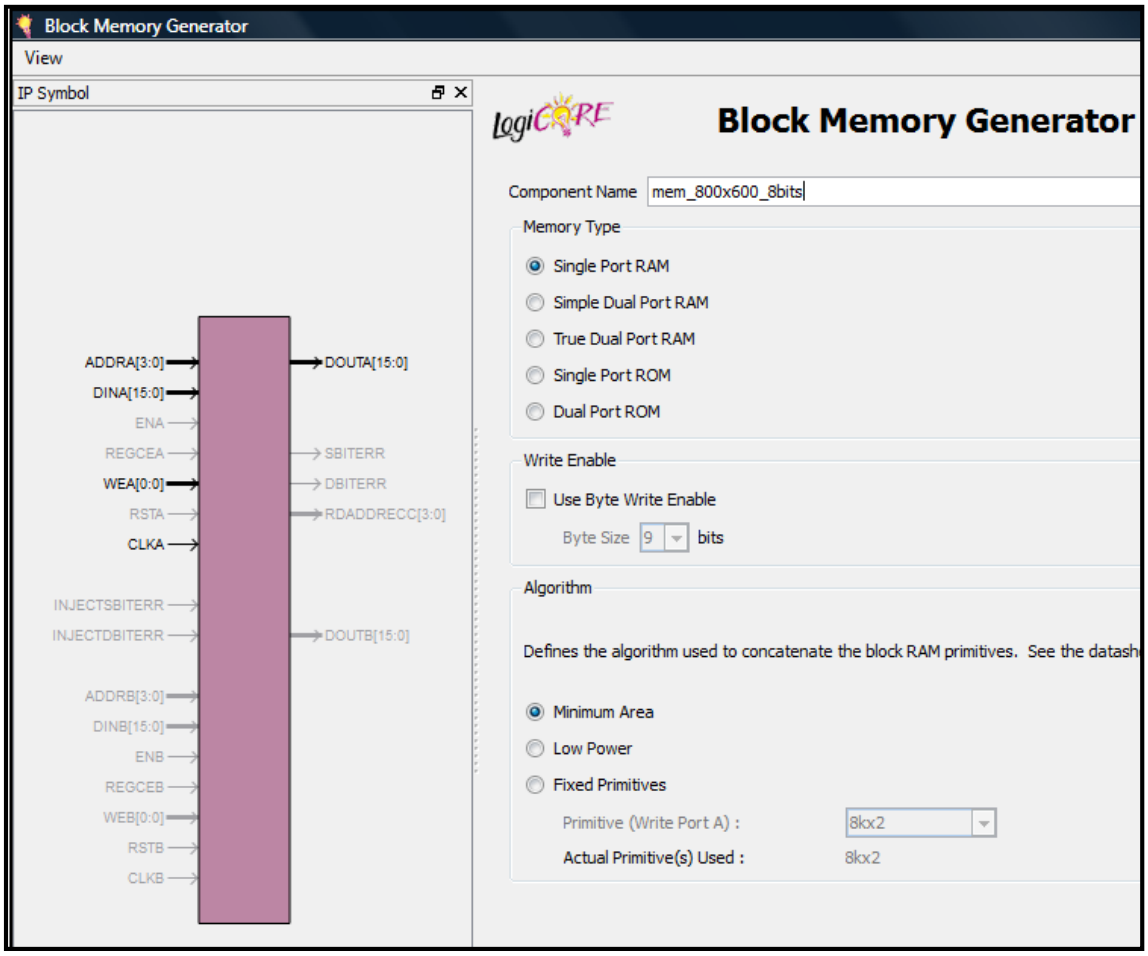


Figura 3.21: Ventana diseño de la memoria

En esta ventana se pone nombre al componente, se marca la opción `single Port RAM` ya que sólo se quiere un puerto de datos.

Paso 7: Selección de las dimensiones de memoria

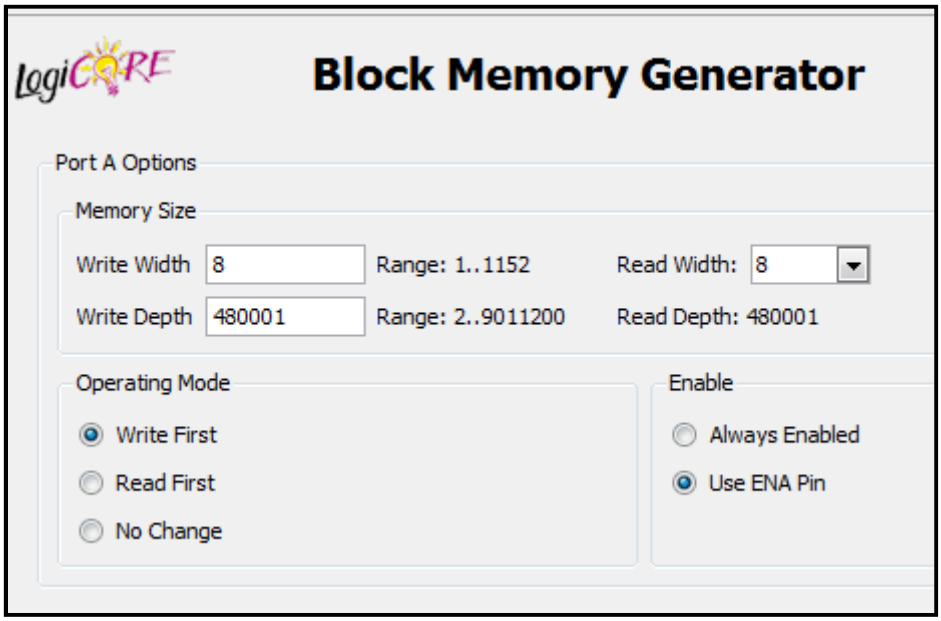


Figura 3.22: Ventana selección de las dimensiones de memoria

Se elige el tamaño del bus de datos, que en este caso es de 8 bits que coincide con el número de bits que definen el color del pixel de la imagen.

También se elige el número de celdas que tiene la memoria para un almacenamiento de imágenes máximo de 800 columnas y 600 filas, por tanto el número de celdas que se eligen es el resultado de la multiplicación 800x600, es decir, 480000 celdas.

Se elige la opción `write first` del `operating mode`, y la opción `Use ENA pin` de `Enable`.

Se sigue avanzando hasta llegar a la ventana del paso 8.

Paso 8: Selección del pin `Reset`

Se selecciona la opción `Use RSTA Pin`, para que se active a nivel alto.

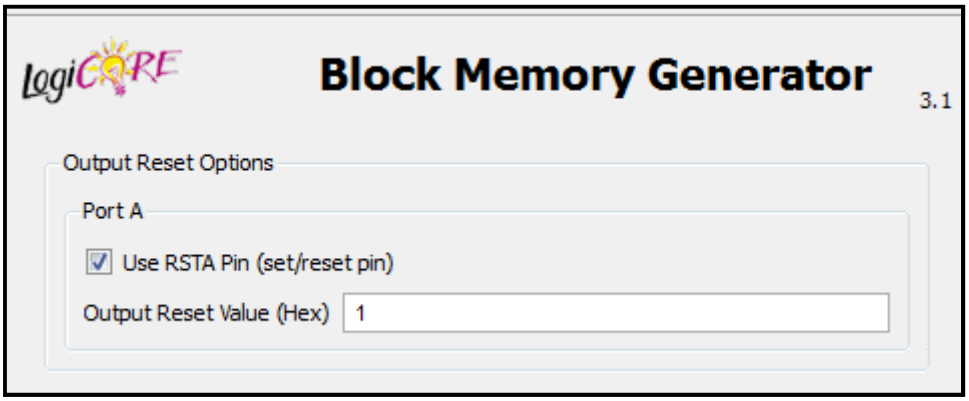


Figura 3.23: Ventana selección del pin reset

Paso 9: Generando memoria

Se pincha en `Generate`

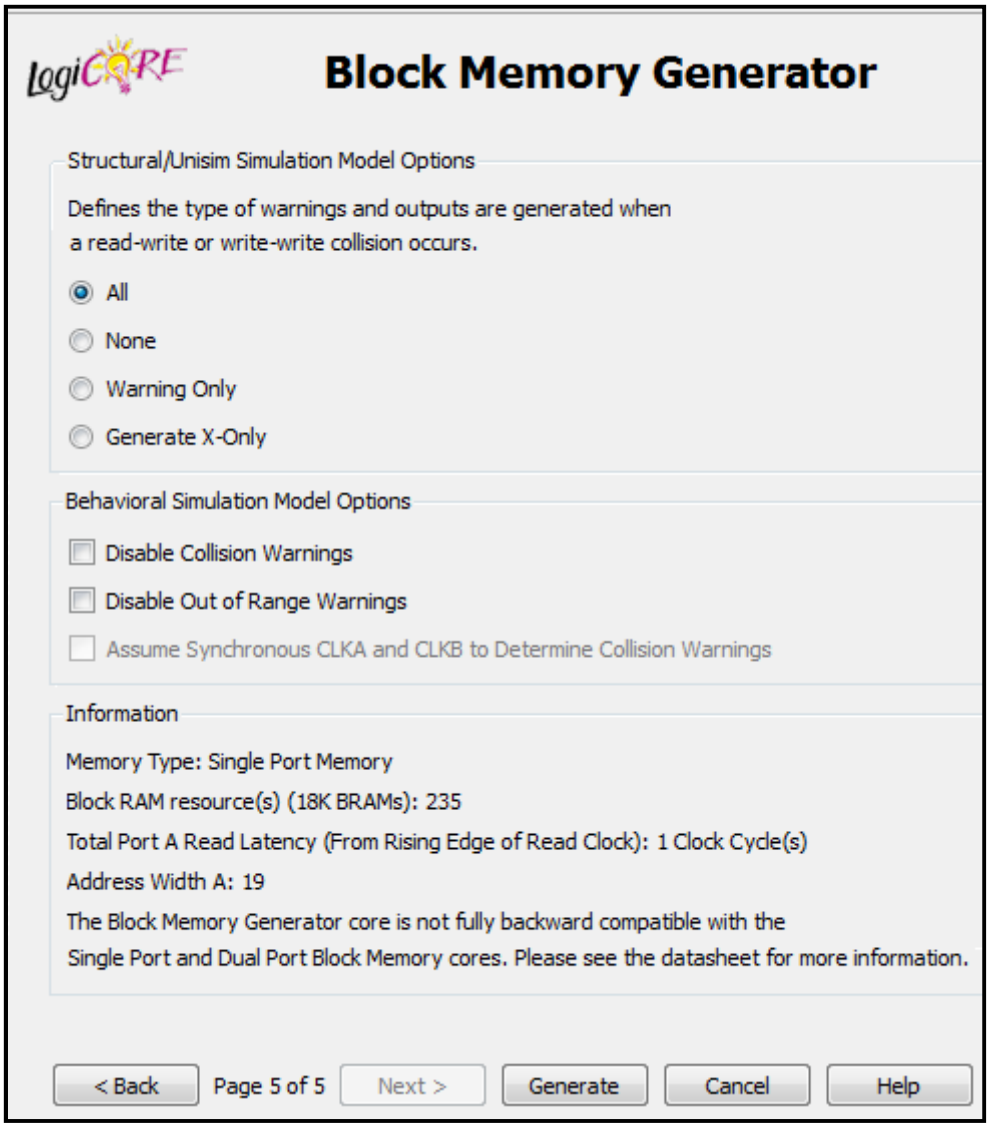


Figura 3.24: Ventana generando memoria

Para obtener el componente creado se va a `Equipo` → `C: /` → `Xilinx` → `12.2` → `ISE` y ahí se encuentran los documentos "mem_800x600_8bits", de tipo `VHD` y `VHO`. El documento `VHD` contiene un código necesario para la compilación de esta memoria, y el `VHO` es un documento que indica como instanciar en el diseño, el bloque de memoria realizado. En el CD facilitado con este documento están estos documentos.

Se instancia la memoria externa en la cámara estéreo, como expone el documento con extensión `VHO`.

Los puertos que componen el módulo “mem_800x600_8bits” son los siguientes:

```
port (
clka: IN std_logic;
rsta: IN std_logic;
ena: IN std_logic;
wea: IN std_logic_VECTOR(0 downto 0);
addra: IN std_logic_VECTOR(18 downto 0);
dina: IN std_logic_VECTOR(7 downto 0);
douta: OUT std_logic_VECTOR(7 downto 0));
```

Figura 3.25: Puertos del modulo memoria externa

3.2.5 Diseño DCM_48MHZ

Este módulo como se comentó anteriormente tiene el cometido de proporcionar una señal de reloj de 48Mhz, que se conecta a varios módulos de la cámara estéreo.

El código de este módulo es el mismo que el código perteneciente al “DCM48” del proyecto *“Diseño y control de una cámara de video digital”*, exceptuando algunos cambios en los siguientes parámetros.

- `CLKFX_MULTIPLY` este parámetro multiplica el valor del periodo de entrada. Se le da el valor de 12.000.000, para así obtener una señal de salida de 48 MHz.

- `CLKIN-PERIOD` como se indicó anteriormente, este parámetro indica el periodo de la señal de entrada que es igual a 10.000.000.

Con el diseño de este último módulo se completa la cámara estéreo, quedando el sistema como se muestra en la siguiente figura.

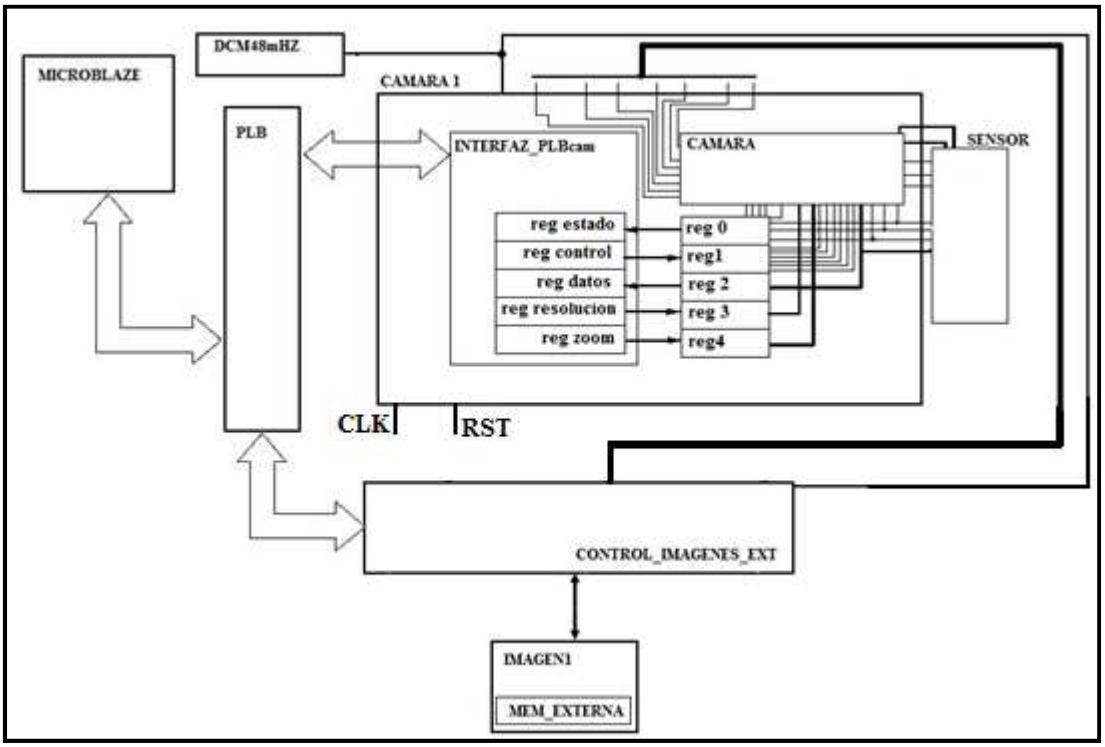


Figura 3.26: Estructura cámara estéreo (con solo una cámara)

El siguiente paso es unir Microblaze con los módulos antes diseñados, y con los periféricos necesarios para el buen funcionamiento del sistema completo, mediante la herramienta de Xilinx XPS.

3.3 Actualización del periférico OPB_cypress_usb

La actualización del "OPB_cypress_usb" [1] a "PLB_cypress_usb", se realiza debido a que el bus OPB se está quedando obsoleto, y además porque todos los periféricos de la cámara estéreo están conectados mediante el bus PLB.

Para la realización de la actualización del periférico, se tiene dos alternativas, la primera es conectar el periférico a otro periférico llamado “OPB_Bridge_PLB”, cuya función es la de realizar la conversión del protocolo OPB al protocolo PLB y viceversa.

Esta alternativa ahorra el modificar el código interno del periférico “OPB_cypress_usb”, a cambio de añadir a nuestro sistema otro periférico. La segunda alternativa es la de modificar el código interno del periférico para adaptarlo al bus PLB, y así no tener que añadir un nuevo periférico que lo adapte al bus PLB.

Se escoge la segunda alternativa, debido a que así, no se añade ningún otro periférico, ahorrando espacio en el sistema. Además se realiza un análisis del código interno del periférico “OPB_cypress_usb” y se observa que la adaptación sólo consiste en la modificación del uso de las librerías, y de algunos puertos externos.

Por tanto, para la adaptación del periférico “OPB_cypress_usb” a “PLB_cypress_usb”, se abre el documento donde está el código interno del “OPB_cypress_usb” y en donde se definen las librerías, se sustituye la librería `OPB_emc` por la librería llamada `PLB_emc`. Esta última librería al igual que la del `OPB_emc`, pertenecen a las librerías disponibles en la herramienta del XPS de Xilinx.

Después de esta sustitución se eliminan los puertos que pertenezcan al bus OPB y se añaden los puertos pertenecientes al PLB, que son los que se muestran en la siguiente figura.

```
SPLB_Clk          : in  std_logic;
SPLB_Rst          : in  std_logic;
PLB_abort         : in  std_logic;
PLB_ABus          : in  std_logic_vector(0 to C_SPLB_AWIDTH-1);
PLB_BE            : in  std_logic_vector(0 to (C_SPLB_DWIDTH/8)-1);
PLB_busLock       : in  std_logic;
PLB_compress      : in  std_logic;
PLB_guarded       : in  std_logic;
PLB_lockErr       : in  std_logic;
PLB_masterID      : in  std_logic_vector(0 to C_SPLB_MID_WIDTH-1);
PLB_MSize         : in  std_logic_vector(0 to 1);
PLB_ordered       : in  std_logic;
PLB_PAValid       : in  std_logic;
PLB_RNW           : in  std_logic;
PLB_size          : in  std_logic_vector(0 to 3);
PLB_type          : in  std_logic_vector(0 to 2);
Sl_addrAck        : out std_logic;
Sl_MBusy          : out std_logic_vector(0 to C_NUM_MASTERS-1);
Sl_MErr           : out std_logic_vector(0 to C_NUM_MASTERS-1);
Sl_rearbitrate    : out std_logic;
Sl_SSize          : out std_logic_vector(0 to 1);
Sl_wait           : out std_logic;
PLB_rdPrim        : in  std_logic;
```

Figura 3.27: Puertos del modulo PLB_cypress_usb

```
PLB_SAValid      : in  std_logic;
PLB_wrPrim       : in  std_logic;
PLB_wrBurst      : in  std_logic;
PLB_wrDBus       : in  std_logic_vector(0 to C_SPLB_DWIDTH-1);
Sl_wrBTerm       : out std_logic;
Sl_wrComp        : out std_logic;
Sl_wrDAck        : out std_logic;
PLB_rdBurst      : in  std_logic;
Sl_rdBTerm       : out std_logic;
Sl_rdComp        : out std_logic;
Sl_rdDAck        : out std_logic;
Sl_rdDBus        : out std_logic_vector(0 to C_SPLB_DWIDTH-1);
Sl_rdWdAddr      : out std_logic_vector(0 to 3) ;
Sl_MRdErr        : out std_logic_vector(0 to C_NUM_MASTERS-1);
Sl_MWrErr        : out std_logic_vector(0 to C_NUM_MASTERS-1);
Sl_MIRQ          : out std_logic_vector(0 to C_NUM_MASTERS-1);
PLB_pendReq      : in  std_logic;
PLB_pendPri      : in  std_logic_vector(0 to 1);
PLB_reqPri       : in  std_logic_vector(0 to 1);
PLB_UABus        : in  std_logic_vector(0 to C_SPLB_AWIDTH-1);
PLB_rdPendPri    : in  std_logic_vector(0 to 1);
PLB_wrPendPri    : in  std_logic_vector(0 to 1);
PLB_rdPendReq    : in  std_logic;
PLB_wrPendReq    : in  std_logic;
PLB_TAttribute   : in  std_logic_vector(0 to 15);
```

Figura 3.27 (Bis): Puertos del modulo PLB_cypress_usb

Con estos dos pasos queda el periférico actualizado al bus PLB.

3.4 Conexión cámara estéreo al Microblaze

El primer paso para la conexión de la cámara con Microblaze, es la configuración de éste. Para ello se utiliza la herramienta de XPS mencionada al comienzo de este proyecto.

3.4.1 Configuración de Microblaze

Los pasos para crear el micro-controlador son los siguientes, se abre la herramienta XPS y en la ventana que aparece, se selecciona la opción `Base System Builder Wizard (recommended)` [1], [10] y [11].

Se aparece una nueva ventana donde se guarda el proyecto.

La siguiente ventana que aparece es como la de la siguiente figura.

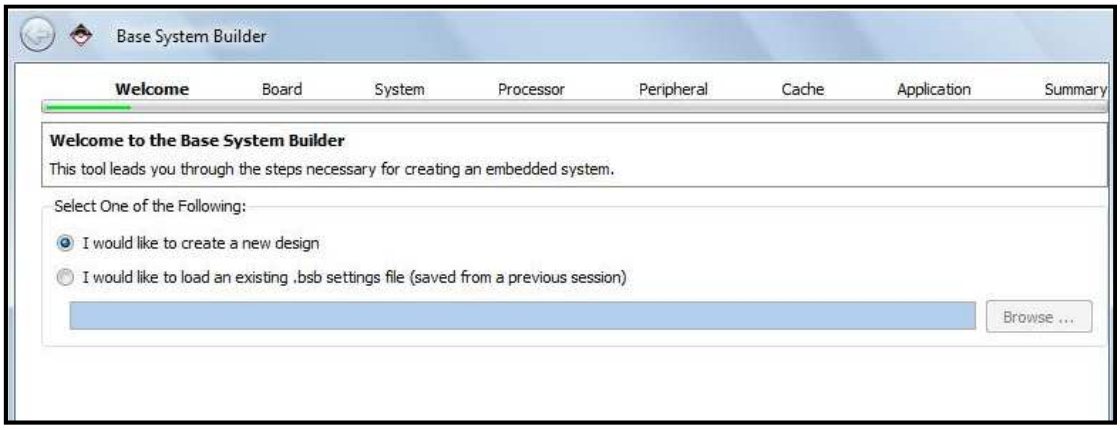


Figura 3.28: Ventana ubicación del proyecto en XPS

Se selecciona el punto `I would like to create a new design`, y se pincha en `Next`.

En la siguiente ventana se elige la tarjeta con la que se va a trabajar, en este caso, se rellenan los espacios como en la siguiente figura y se pulsa `Next`.

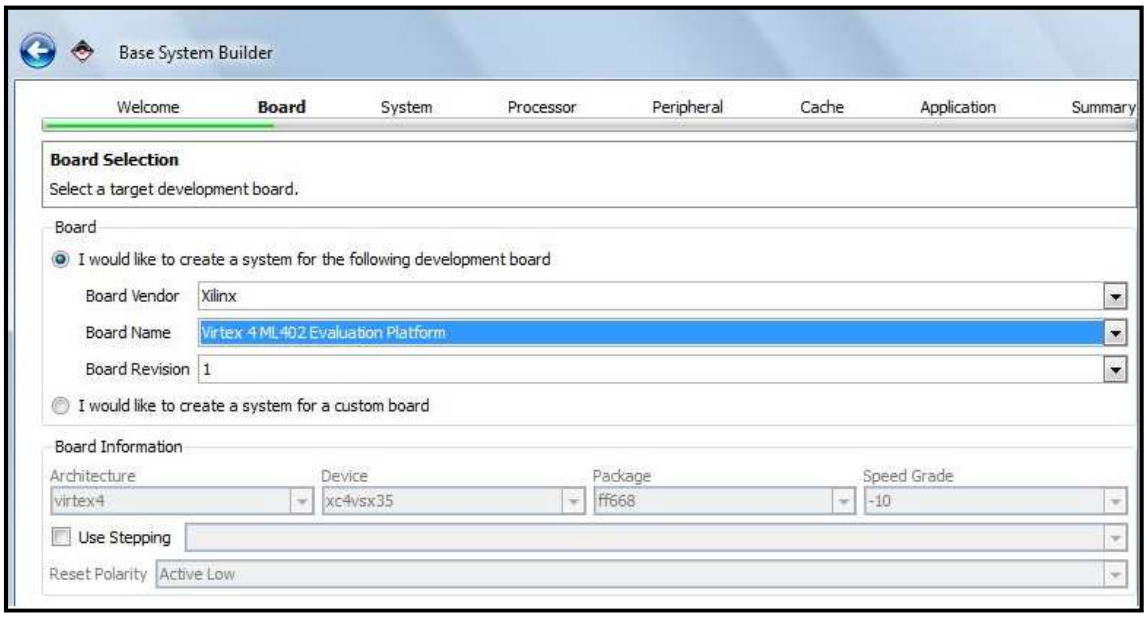


Figura 3.29: Ventana selección placa Xilinx en XPS

En la siguiente ventana se da la opción de elegir un sistema con un o dos microprocesadores, en este caso sólo se utiliza uno, por tanto, se elige la opción `Single Processor System` y se pincha en `Next`.

En la nueva ventana, se elige la frecuencia del reloj (100 MHz), el tipo de procesador (Microblaze) y el tamaño de la `Local Memory` (32 KB). Todo esto está representado en la siguiente figura.

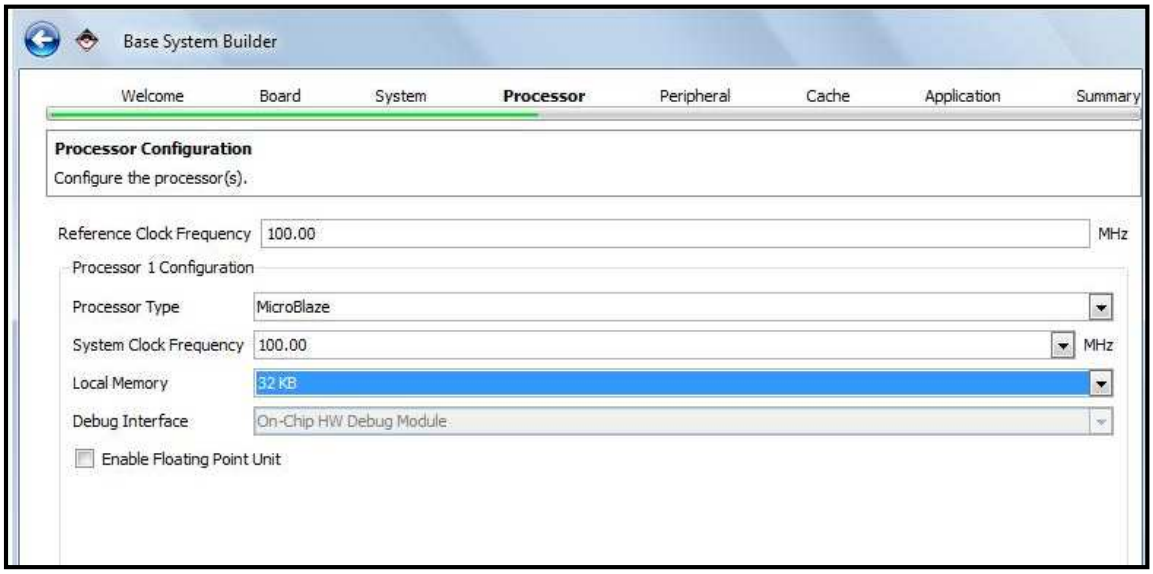


Figura 3.30: Ventana selección frecuencia en XPS

A continuación, aparece una ventana en la cual se elige los periféricos que se quieren para el sistema. Para este proyecto se necesitan los siguientes:

a. DDR_SDRAM
b. RS232_Uart → Baud Rate: 115200, Data Bits: 8, Parity: None, Use Interrupt: ☑
c. SysACE_CompactFlash → Use Interrupt: ☑
d. xps_timer_0 → Count Width: 32, Configure Mode: One Timer is present, Use Interrupt: ☑

Los demás periféricos se deseleccionan porque no se necesitan, quedando la ventana de la siguiente manera.

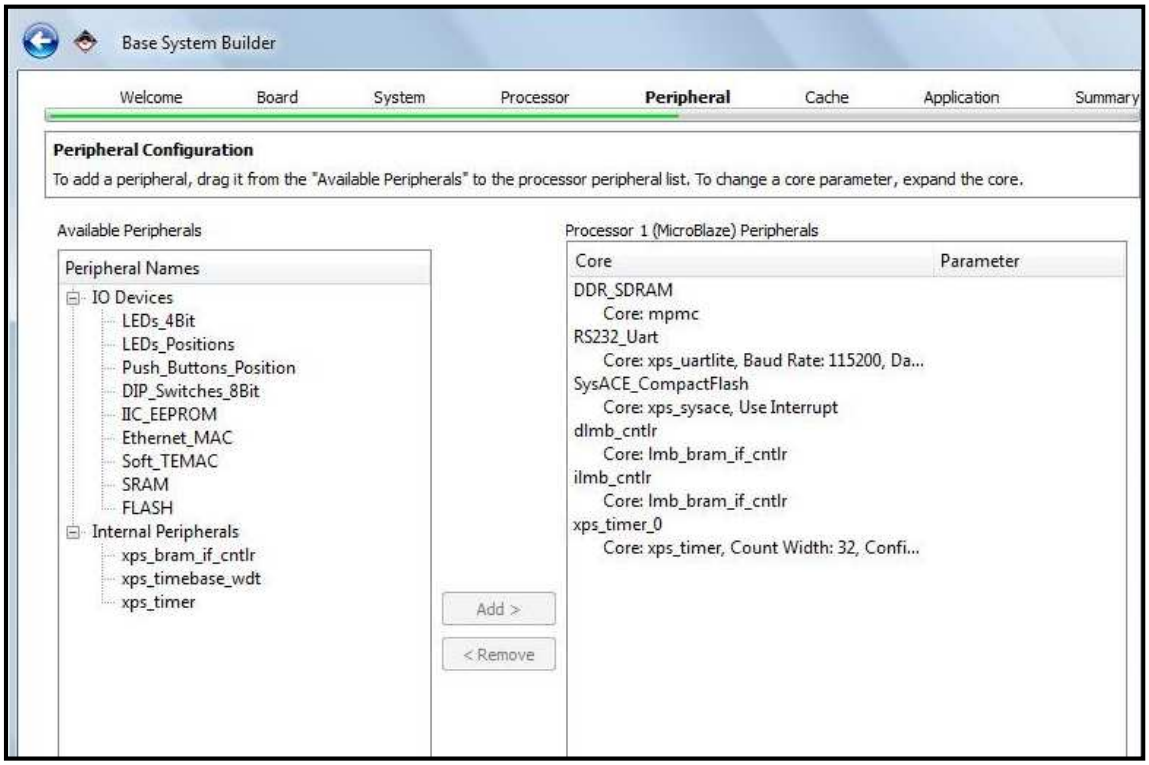


Figura 3.31: Ventana selección de periféricos en XPS

Se pulsa `Next`, y en la siguiente ventana, se selecciona `Instruction Cache`, con un tamaño de 32KB. Y también se selecciona `Data Cache`, con el mismo tamaño que la anterior. El espacio de memoria se obtiene de la memoria `DDR_SDRAM` en ambos casos. Una vez se selecciona todo lo anterior se pulsa `Next`.

En la siguiente ventana se deja todo como aparece por defecto, como se muestra en la siguiente figura.

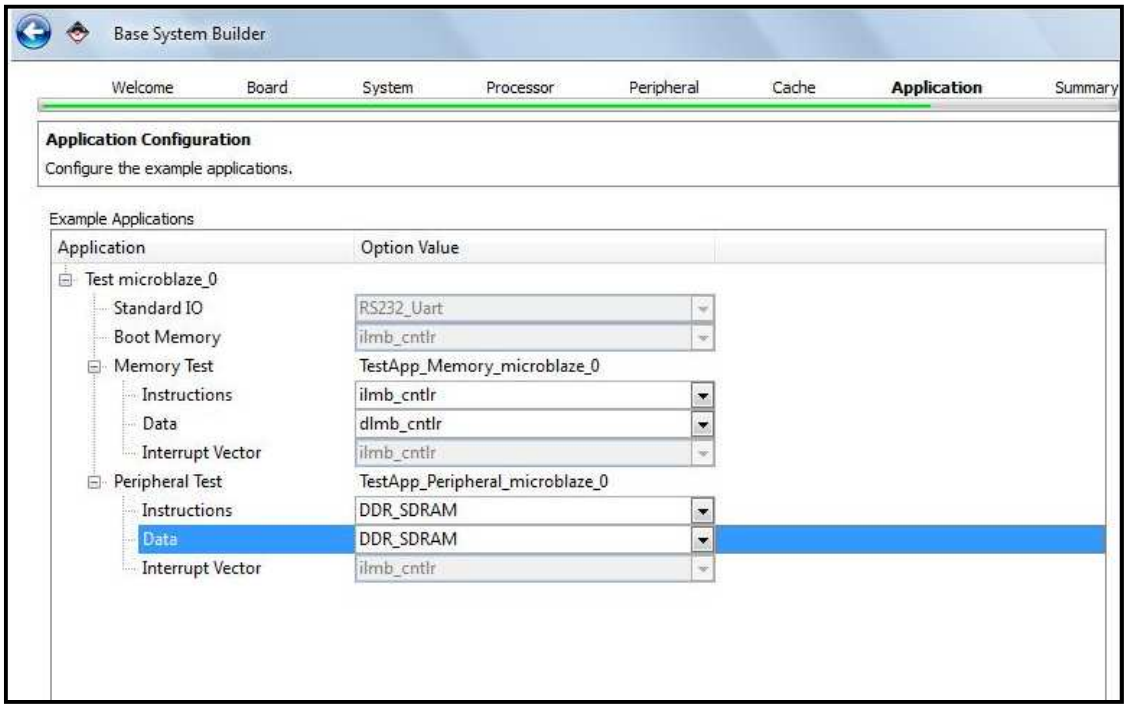


Figura 3.32: Ventana configuración de aplicaciones en XPS

Se pincha en `Next`.

Aparece una última ventana donde se resume las características del procesador. La ventana se muestra en la siguiente figura.

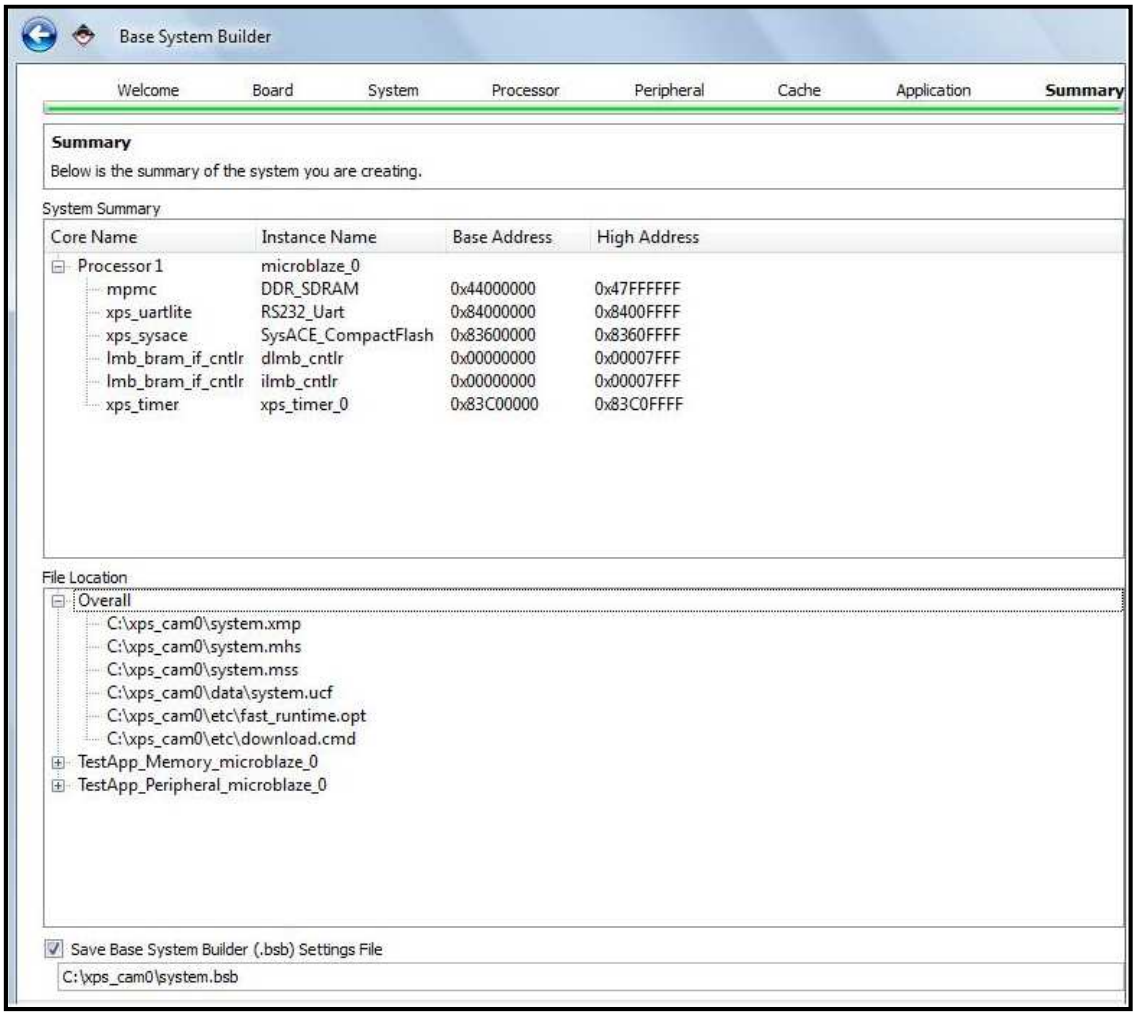


Figura 3.33: Ventana resumen de Microblaze en XPS

Y se pincha en `Finish`.

3.4.2 Integración de periféricos al Microblaze

- **Desarrollo de archivos MPD y PAO**

Estos archivos describen las características de los periféricos, y son necesarios porque la herramienta de Xilinx los demandará cuando llegue la hora de importar dichos periféricos.

Los archivos microprocesador Peripherals `description (MPD)`, están compuestos por los parámetros del periférico, así como de los puertos que lo componen. Un ejemplo a la hora de definir un parámetro y un puerto es el siguiente:

```
PARAMETER C_NUM_BANKS_MEM = 1, DT = INTEGER, IO_IS = C_NUM_BANKS_MEM, RANGE = (1:4)
```

```
PORT SPLB_Clk = “”, DIR = I, SIGIS = CLK, BUS = SPLB
```

Los códigos de este tipo de archivo de todos los periféricos están expuestos en el CD facilitado en este documento.

Los archivos `peripheral analyze order (PAO)`, son archivos compuestos por las librerías de las que está formado el periférico. Un ejemplo es el siguiente:

```
Lib emc_common_v2_00_a mem_steer vhdl
```

Esto significa que nuestro periférico necesita el código en VHDL, llamado `mem_steer`, que se encuentra en la librería `emc_common_v2_00_a`.

Al igual que en el otro tipo de archivos, estos se encuentran en el CD facilitado en este documento.

- **Proceso de importación de periféricos al proyecto.**

El primer periférico que se añade es el "PLB_Cypress_usb", el cual como se ha comentado anteriormente es la actualización del periférico "OPB_Cypress_usb", del proyecto de "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*".

Para ello se pincha en `Hardware → Create or Import Peripheral Wizard`.

En la ventana de bienvenida, se pulsa `Next`, y en la siguiente que aparece se selecciona la opción `Import existing Peripherals` ya que se tiene el periférico previamente diseñado y se pulsa `Next`.

En la siguiente ventana se selecciona `To an Xps Project`, porque se quiere guardar el periférico dentro del proyecto, y como en las anteriores se pulsa `Next`.

En la nueva ventana que aparece se pone el nombre del periférico, en este caso se llama "PLB_cypress_usb", y se pincha en Use versión, quedando el nombre del periférico como "plb_cypress_usb_v1_00_a".

Se pulsa `Next` en la ventana anterior, y aparece otra donde se selecciona `HDL source files`, lo que significa que el periférico se compone de archivos con código fuente en `HDL`, en este caso `VHDL`.

Se vuelve a pulsar `Next` en la ventana anterior, y en la siguiente se selecciona `VHDL` en la pestaña correspondiente `a HDL language used to implement your Peripherals.`

Se selecciona también `Use data (.mpd) collected during a previous invocation of this tool`, donde se pincha en `Browse` y se busca el archivo `(.mpd)` que se ha creado anteriormente para este periférico, y además de este, se selecciona `Use existing Peripheral Analysis Order File (.pao)`, donde se procede de la misma forma que con el archivo anterior. Una vez añadidos los archivos se pulsa `Next`.

En la nueva ventana aparece las librerías en las que está compuesto el periférico, y se pincha en `Next`, apareciendo una nueva ventana, donde se selecciona, `Select bus Interface(s) → PLBv46 Slave (SPLB) → Next.`

En la ventana que aparece tras el último `Next`, no se modifica, y en la siguiente se selecciona los parámetros como se muestra en la siguiente figura:

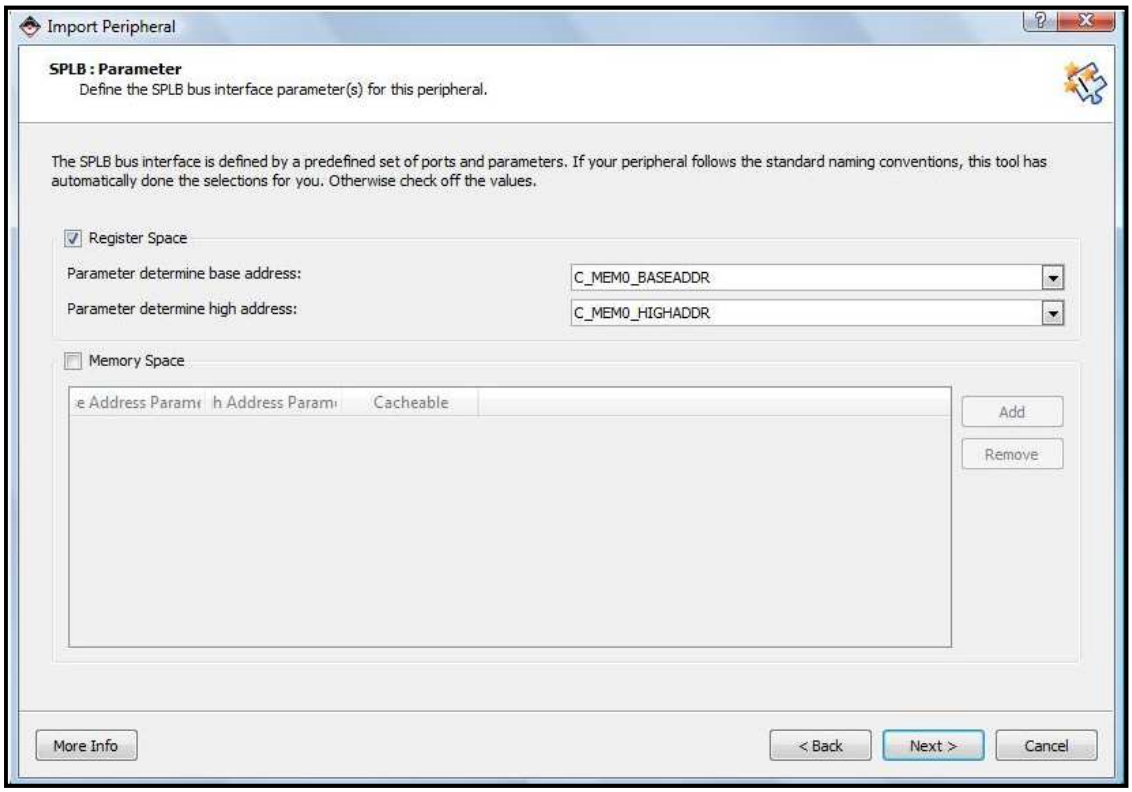


Figura 3.34: Ventana selección parámetros SPLB del periférico

En la siguiente ventana se deja como esta por defecto (asegúrense que la ventana coincide con la imagen que a continuación se muestra)

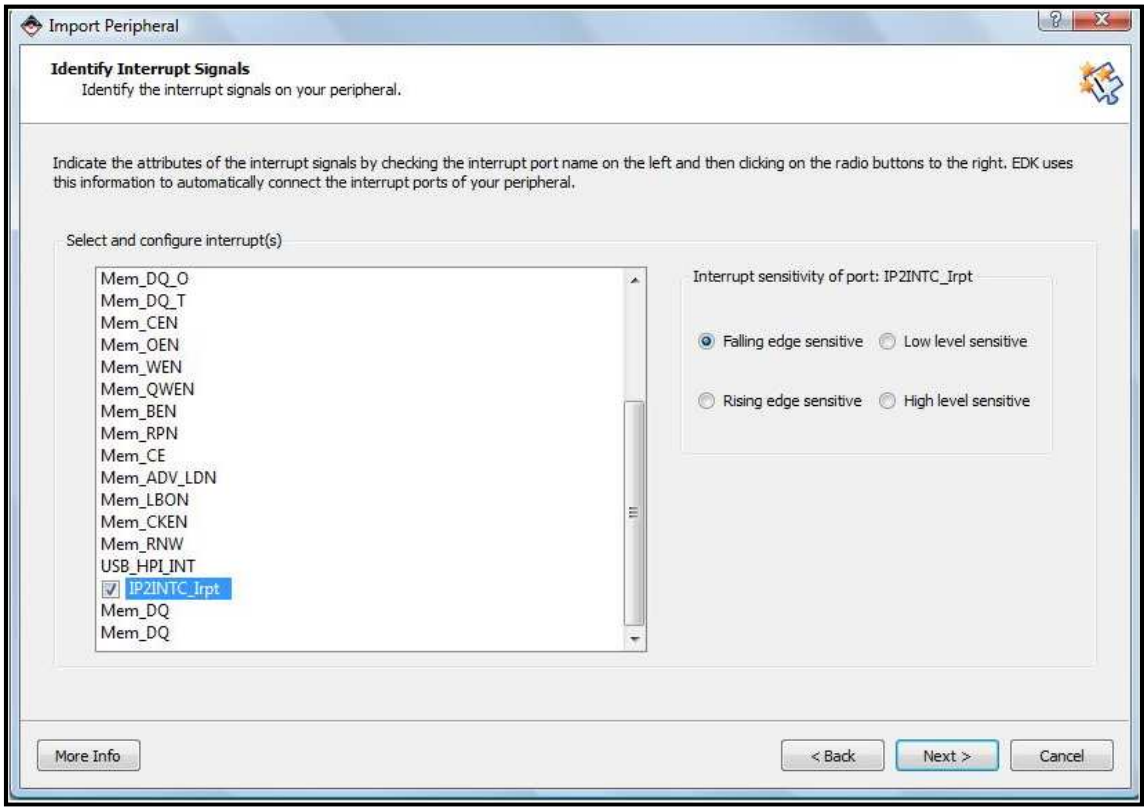


Figura 3.35: Ventana selección de interrupción

En las siguientes ventanas se pincha en `Next`, hasta llegar a la última ventana donde se pincha en Finish.

Una vez realizado todo lo anterior, se pincha en `IP Catalog` y en `USER` debe aparecer el nombre del periférico que se acaba de importar.

A continuación, se añade el periférico "sysace_usb_mux", facilitado en el proyecto "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*".

Para añadirlo se procede como el anterior periférico, teniendo en cuenta, que éste no se conecta al bus PLB, ni tampoco tiene ninguna interrupción.

Se añade también de la misma forma que la anterior los siguientes periféricos:

control_imagenes_ext → conectado al PLB y sin interrupciones.

plb_camara1 → conectado al PLB y sin interrupciones.

nimagen1 → sin conexión al PLB y sin interrupciones.

Memoria_externa → sin conexión al PLB y sin interrupciones.

Para este periférico, donde se selecciona `HDL source files`, también se selecciona la opción `Netlist files`. Más adelante la herramienta de importación de periféricos demanda un archivo `Netlist`.

El archivo que se añade es el obtenido al generar la memoria con el `CORE Generator`, y que tiene extensión .`ngc`.

Esto se realizará para solventar un error en la sintetización del sistema.

dcm48mhz → sin conexión al PLB y sin interrupciones.

- **Integración de los periféricos al sistema cámara estéreo**

Para introducir los periféricos en el sistema, se realiza "doble click" en cada uno de ellos, y en la ventana que emerge se pincha en YES.

Conexión de los periféricos al bus PLB

La primera conexión que se realiza es la del bus PLB a los periféricos que lo requieran. Estos periféricos son las plb_camaras, el PLB_cypress_usb y el control_imagenes_ext, para ello se procede de la siguiente manera. Se pincha en la pestaña `Bus Interfaces`, posteriormente, se pincha en "+" del periférico, donde se despliega una pestaña en la cual se selecciona `mb_plb`. Así para todos los periféricos que requieran ser conectados, quedando el sistema como se muestra en la siguiente figura:

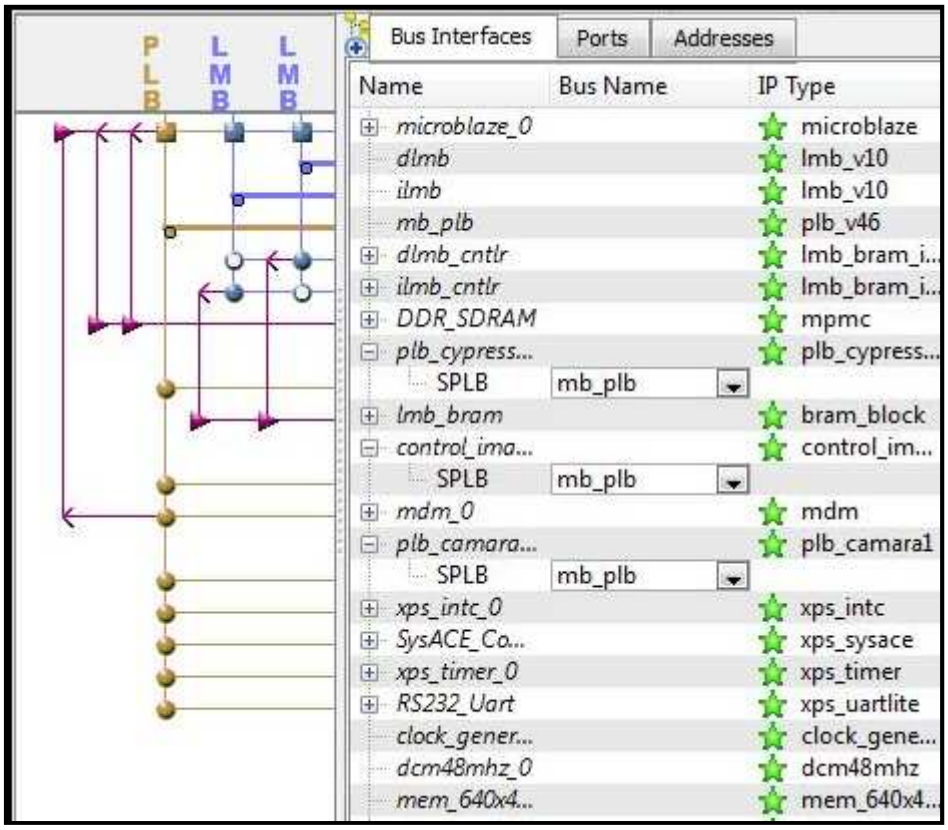
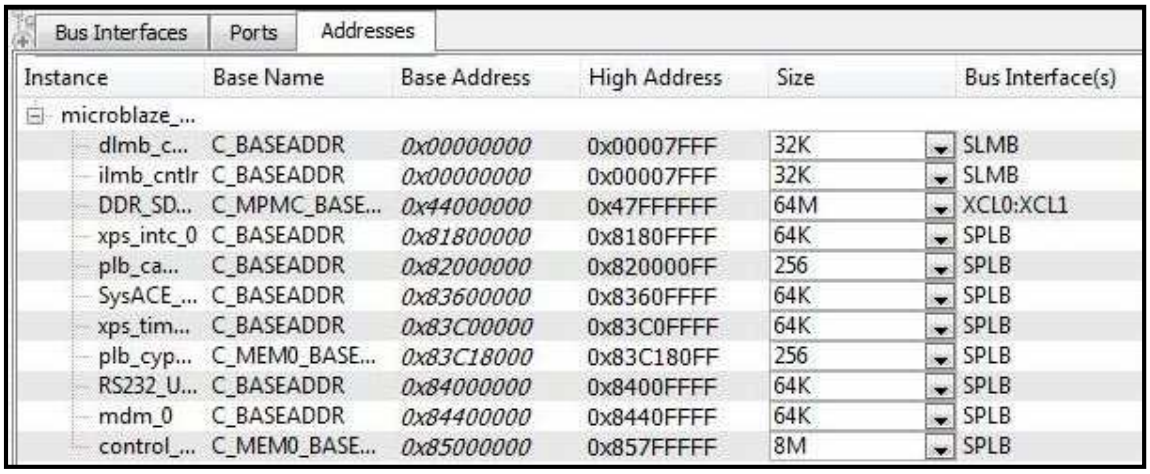


Figura 3.36: Periféricos conectados al Bus interfaz PLB

A continuación, se asigna a los periféricos una dirección del bus PLB para poder establecer comunicación con ellos a partir de la misma. Por tanto, se pincha en la pestaña `Addresses`, se selecciona `Unmapped` y se pincha en `Generate Addresses.`

Se pincha en la pestaña de tamaño del periférico "control_imagenes_ext", y se selecciona un tamaño de 8M, este tamaño depende del tamaño y número de las imágenes que se almacenan, y se hace "doble click" encima de la dirección de origen del periférico para cambiarla a 0x85000000.

Se muestra la asignación de las direcciones en la siguiente figura:



Bus Interfaces | Ports | Addresses

Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)
microblaze_...					
dlmb_c...	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB
ilmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB
DDR_SD...	C_MPMC_BASE...	0x44000000	0x47FFFFFF	64M	XCL0:XCL1
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF	64K	SPLB
plb_ca...	C_BASEADDR	0x82000000	0x820000FF	256	SPLB
SysACE_...	C_BASEADDR	0x83600000	0x8360FFFF	64K	SPLB
xps_tim...	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB
plb_cyp...	C_MEM0_BASE...	0x83C18000	0x83C180FF	256	SPLB
RS232_U...	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB
control_...	C_MEM0_BASE...	0x85000000	0x857FFFFF	8M	SPLB

Figura 3.37: asignación de direcciones del bus PLB a periféricos

La conexión y asignación de dirección de memoria para la "plb_camara2" y "plb_camara3", se realiza de la misma manera que para la "plb_camara1".

También se realiza la introducción del periférico `xps_GPIO` porque se necesita para el buen funcionamiento del sistema (explicado con más detalle en "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*), para ello se pincha en la pestaña `IP Catalog` → `General Purpose IO`, y se hace "doble click" en el periférico que se despliega del menú. Se pincha en YES, en la ventana que emerge y se realiza los pasos que se han seguido para los demás periféricos.

Conexión de los puertos de los periféricos

Para realizar la conexión de los puertos de los periféricos entre ellos, se pincha en la pestaña `Ports`, se despliega el menú del periférico pinchando en "+" del periférico. Uno a uno se les pone nombre a los puertos para conectarlos a otros.

Ejemplo:

El puerto `read1`, del periférico "control_imagenes_ext", se le asigna el nombre de `read1_i`. Para conectar este puerto con el puerto `read` del periférico "nimagen", se le asigna a este último el nombre de `read1_i`.

Esta es la manera de proceder para la conexión de los restantes periféricos.

Una manera rápida e igualmente eficaz, se consigue modificando el archivo `system.mhs`, archivo donde se define las conexiones de cada periférico.

Se procede a la conexión de los puertos de los periféricos entre sí. Se abre el archivo `system.mhs` y se introduce las siguientes líneas.

Para el plb_camara1:

```
PORT sys_clk = clk_100_0000MHzDCM0

PORT sys_rst = sys_rst_s

PORT rst_ram = rst_ram_ext

PORT cam_data = cam_data_ext

PORT cam_line_valid = cam_line_valid_ext

PORT cam_frame_valid = cam_frame_valid_ext

PORT cam_pixclk = cam_pixel_ext

PORT clkin_cam = clkin_cam_ext

PORT reset_cam = reset_cam_ext

PORT sclk_cam = sclk_cam_ext

PORT sdata_cam = sdata_cam_ext

PORT clk_48mhz = dcm48mhz_0_CLKFX_OUT

PORT data_out = data_out1

PORT act_mem = act_mem1_i

PORT f_col = f_col1_i

PORT done = done1_i

PORT fv = fv1_i

PORT read = read1_i

PORT f_img = f_img1_i

PORT oe_cam = oe_cam_ext
```

Para el control_imagenes_ext:

```
PORT clk48mhz = dcm48mhz_0_CLKFX_OUT


PORT act_mem1 = act_mem1_i
PORT f_col1 = f_col1_i
PORT done1 = done1_i
PORT fv1 = fv1_i
PORT data_pix1 = data_pix1_i
PORT f_img1 = f_img1_i
PORT read1 = read1_i
PORT r_w = r_w_i
PORT reset_in = sys_rst_s
```

Para el nimagen1:

```
PORT sys_clk = clk_100_0000MHzDCM0
PORT rst_ram_O = rst_ram_int
PORT dina = dina_i
PORT addra = addra_i
PORT ena = ena_i
PORT wea = wea_i
PORT douta = douta_i
PORT addrs = addrs_i
PORT data_in = data_out1
PORT data_out = data_pix1_i
PORT read = read1_i
```

```
PORT rst_ram_I = rst_ram_ext
PORT write = r_w_i
```

Para la memoria externa (mem_800x600_8bits):

```
PORT clka = clk_100_0000MHzDCM0
PORT rsta = rst_ram_int
PORT ena = ena_i
PORT wea = wea_i
PORT addra = addra_i
PORT dina = dina_i
PORT douta = douta_i
```

Para el dcm48mhz:

```
PORT CLKIN_IN = clk_100_0000MHzDCM0
PORT RST_IN = net_gnd
PORT CLKFX_OUT = dcm48mhz_0_CLKFX_OUT
```

Para el plb_cypress_usb:

```
PORT USB_HPI_INT = usb_int
PORT Mem_CE = usb_ce
PORT Mem_WEN = usb_wrn
PORT Mem_OEN = usb_rdn
PORT Mem_CEN = usb_csn_i
PORT Mem_DQ_T = usb_data_t
PORT Mem_DQ_O = usb_data_o
```

```
PORT Mem_DQ_I = usb_data_i
PORT IP2INTC_Irpt = plb_cypress_usb_0_IP2INTC_Irpt
PORT Mem_A = usb_addr
```

Para el sysace_usb_mux:

```
PORT clk = clk_100_0000MHzDCM0
PORT usb_csn = usb_csn
PORT sysace_mpa = sysace_mpa
PORT sysace_mpd_I = sysace_mpd_i
PORT sysace_mpd_O = sysace_mpd_o
PORT sysace_mpd_T = sysace_mpd_t
PORT sysace_mpoe = sysace_mpoe
PORT sysace_mpwe = sysace_mpwe
PORT sysace_usb_mpa = sysace_usb_mpa
PORT sysace_usb_mpd = sysace_usb_mpd
PORT sysace_usb_oen = sysace_usb_oen
PORT sysace_usb_wen = sysace_usb_wen
PORT usb_ce = usb_ce
PORT usb_wrn = usb_wrn
PORT usb_csn_i = usb_csn_i
PORT usb_rdn = usb_rdn
PORT usb_data_t = usb_data_t
PORT usb_data_o = usb_data_o
PORT usb_data_i = usb_data_i
PORT usb_addr = usb_addr
```

Para el xps_sysace:

```
PORT SysACE_CLK = sysace_clk

PORT SysACE_MPA = sysace_mpa

PORT SysACE_CEN = sysace_cen

PORT SysACE_OEN = sysace_mpoe

PORT SysACE_WEN = sysace_mpwe

PORT SysACE_MPIRQ = sysace_mpirq

PORT SysACE_IRQ = SysACE_CompactFlash_SysACE_IRQ

PORT SysACE_MPD_T = sysace_mpd_t

PORT SysACE_MPD_O = sysace_mpd_o

PORT SysACE_MPD_I = sysace_mpd_i
```

En el sistema se añaden los puertos externos para la conexión de la cámara estéreo, usb, etc. Para ello se introducen en la cabecera del archivo antes mencionado las siguientes líneas:

```
# Segnales Cypress_USB/SYSACE
 PORT sysace_usb_mpa = sysace_usb_mpa, DIR = O, VEC = [6:0]
 PORT sysace_usb_mpd = sysace_usb_mpd, DIR = IO, VEC = [15:0]
 PORT sysace_usb_oen = sysace_usb_oen, DIR = O
 PORT sysace_usb_wen = sysace_usb_wen, DIR = O
# Segnales SYSTEMACE
 PORT sysace_clk = sysace_clk, DIR = I
 PORT sysace_cen = sysace_cen, DIR = O
 PORT sysace_mpirq = sysace_mpirq, DIR = I
# Segnales CYPRESS_USB
 PORT usb_csn = usb_csn, DIR = O
 PORT usb_int = usb_int, DIR = I, SENSITIVITY = LEVEL_HIGH,
 SIGIS = INTERRUPT
# Segnales externas de la PLB_CAMARA1
 PORT sdata_cam_ext_pin = sdata_cam_ext, DIR = IO
 PORT sclk_cam_ext_pin = sclk_cam_ext, DIR = IO
 PORT cam_data_ext_pin = cam_data_ext, DIR = I, VEC = [9:0]
 PORT cam_line_valid_ext_pin = cam_line_valid_ext, DIR = I
 PORT cam_frame_valid_ext_pin = cam_frame_valid_ext, DIR = I
 PORT oe_cam_ext_pin = oe_cam_ext, DIR = O
 PORT cam_pixel_ext_pin = cam_pixel_ext, DIR = I
 PORT clkin_cam_ext_pin = clkin_cam_ext, DIR = O
 PORT reset_cam_ext_pin = reset_cam_ext, DIR = O
```

Figura 3.38: Puertos externos definidos en system.mhs

Eliminando las líneas siguientes:

```
PORT fpga_0_SysACE_CompactFlash_SysACE_MPA_pin =
fpga_0_SysACE_CompactFlash_SysACE_MPA_pin, DIR = O, VEC = [6:0]
PORT fpga_0_SysACE_CompactFlash_SysACE_CLK_pin =
fpga_0_SysACE_CompactFlash_SysACE_CLK_pin, DIR = I
PORT fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin =
fpga_0_SysACE_CompactFlash_SysACE_MPIRQ_pin, DIR = I
PORT fpga_0_SysACE_CompactFlash_SysACE_CEN_pin =
fpga_0_SysACE_CompactFlash_SysACE_CEN_pin, DIR = O
PORT fpga_0_SysACE_CompactFlash_SysACE_OEN_pin =
fpga_0_SysACE_CompactFlash_SysACE_OEN_pin, DIR = O
PORT fpga_0_SysACE_CompactFlash_SysACE_WEN_pin =
fpga_0_SysACE_CompactFlash_SysACE_WEN_pin, DIR = O
PORT fpga_0_SysACE_CompactFlash_SysACE_MPD_pin =
fpga_0_SysACE_CompactFlash_SysACE_MPD_pin, DIR = IO, VEC = [15:0]
```

Figura 3.39: Puertos externos eliminados del archivo system.mhs

Conexiones para interrupciones

Para la conexión de las interrupciones se pincha en la pestaña `Ports`, y se despliega el menú del periférico `xps_intc_0`, pinchando en "+".

Del menú que se despliega, se hace "doble click" en `Intr`. A continuación, aparece una ventana la cual se modifica hasta que quede como en la siguiente figura:

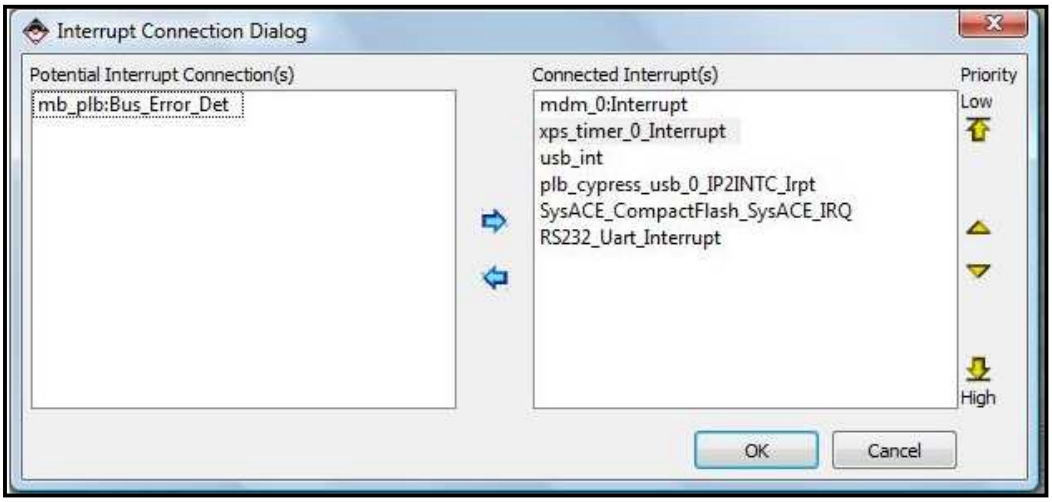


Figura 3.40: Conexión de interrupciones por prioridades

Asignación de puertos externos a localizaciones de la placa Virtex4.

Para realizar la asignación de los puertos externos del sistema a la localizaciones de los pines de la placa donde posteriormente se conecta el sensor de la cámara, se procede abriendo el archivo `system.ucf`, localizado en la carpeta data, dentro de la carpeta donde está guardado el proyecto.

Este archivo se modifica de la siguiente manera:

En primer lugar se realizan las asignaciones de los puertos externos del periférico "plb_cypress_usb", estas asignaciones son proporcionadas por el proyecto "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*".

En segundo lugar se procede a las asignaciones de los puertos externos del periférico "plb_camara1". Para ello se eligen los pines del `Single-Ended Expansion I/O Connectors (J6)` [12] y [13] de la placa como se muestra en la siguiente figura:

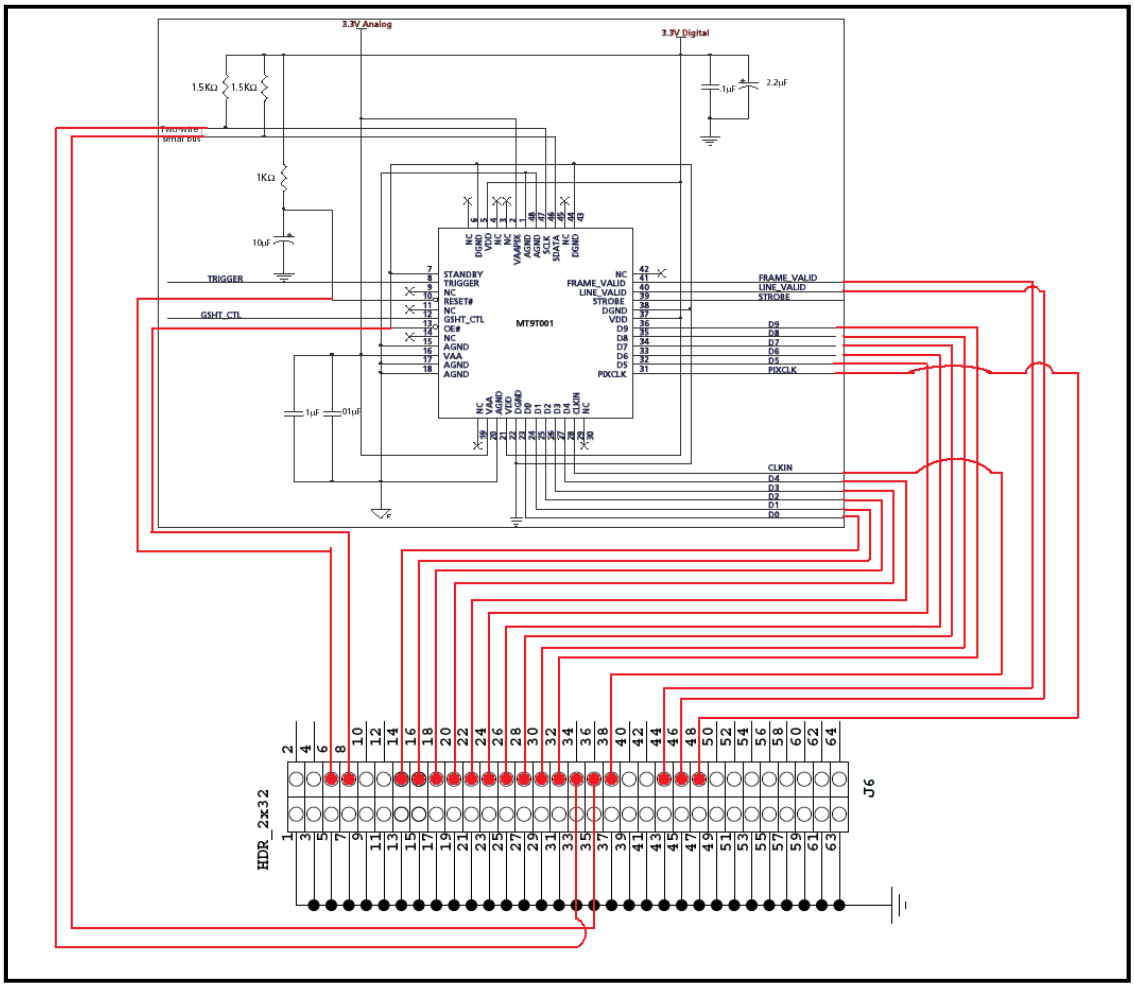


Figura 3.41: Conexión de pines de la cámara a los conectores de expansión de la placa

3.4.3 Compilación del sistema

Para la compilación del sistema se pincha en `Applications`, después se pincha con el botón derecho del ratón en Project: `TestApp_Memory_microblaze`, y en el menú que se despliega se selecciona `Mark to Initialize BRAMs`, quedando el menú como el de la siguiente figura:

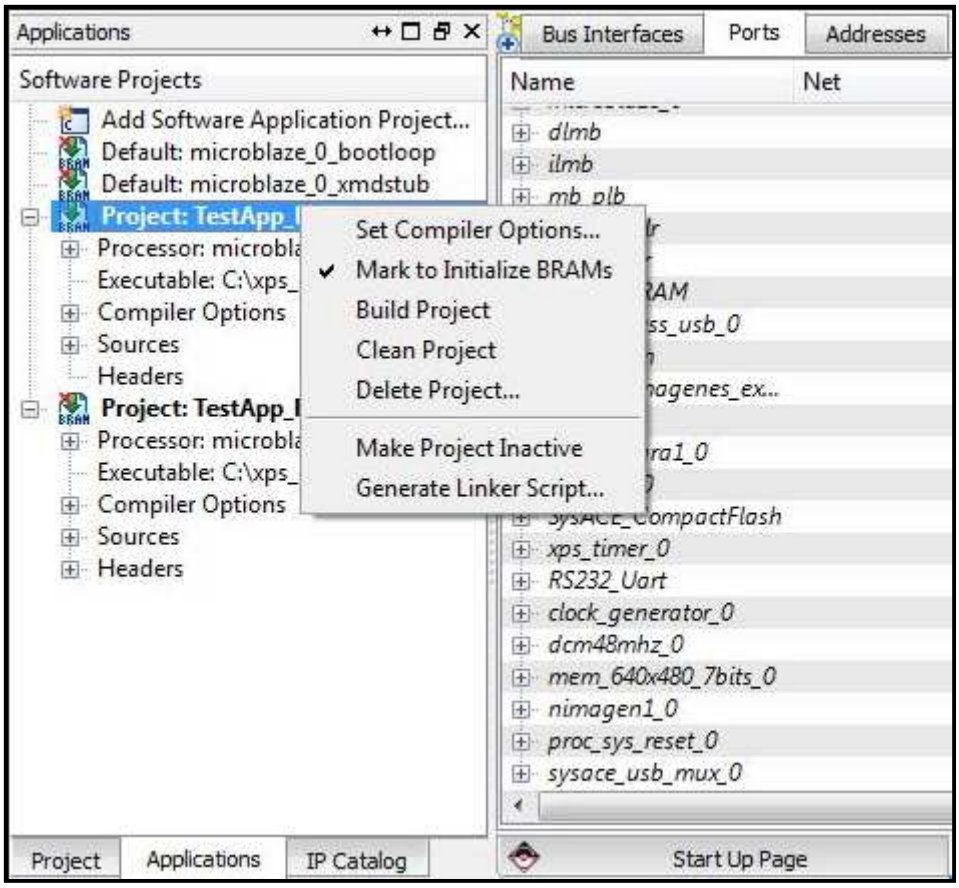


Figura 3.42: Activación del Mark to Initialize BRAMs para compilación

Se pincha en `Device Configuration` → `Update Bitstream`, y comienza el proceso de compilación, sintetización, etc. Se espera a que el proceso termine satisfactoriamente. Esto tarda bastantes minutos.

3.4.4 Inclusión del sistema operativo al Microblaze

El Sistema Operativo que se utiliza es el Petalinux, donde en el proyecto "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*", explica en detalle, tanto su funcionamiento, como su integración en el sistema. De todos modos en este proyecto se explica su integración en el sistema.

El software Petalinux se puede encontrar los archivos del proyecto antes mencionado, sin embargo, se obtiene una versión más actualizada del mismo, y se copia su carpeta repositorio en el lugar correspondiente como indica el proyecto "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*", se abre la herramienta XPS, y se pincha en `Software` → `Software Platform Settings`.

Se aparece una ventana, en la cual se debe elegir en `OS`: (parte de abajo de la ventana) Petalinux.

Se despliega el menú de `microblaze_0`, se pincha en la pestaña de `xmdstub_peripheral` y se elige `mdm_0`. La ventana debe quedar como se muestra en la siguiente figura:

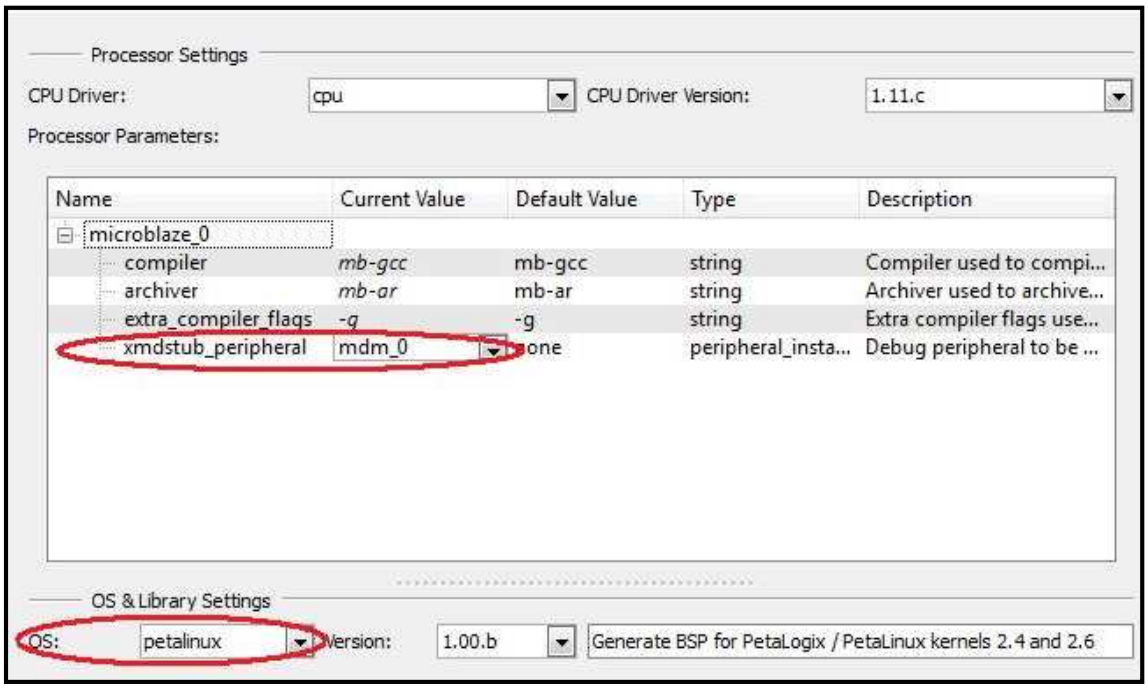


Figura 3.43: Ventana de selección de Petalinux

A continuación, se selecciona `OS` and `Lib Configuration` (parte izquierda de la ventana), se despliega el menú de Petalinux, y se realizan las siguientes modificaciones:

stdout → RS232_uart.

stdin → RS232_uart.

main_memory → DDR_SDRAM.

flash_memory_bank → -1.

lmb_memory → dlmb_cntrl.

Los demás parámetros se dejan con los valores por defecto.

La ventana debe quedar como se muestra en la siguiente figura:

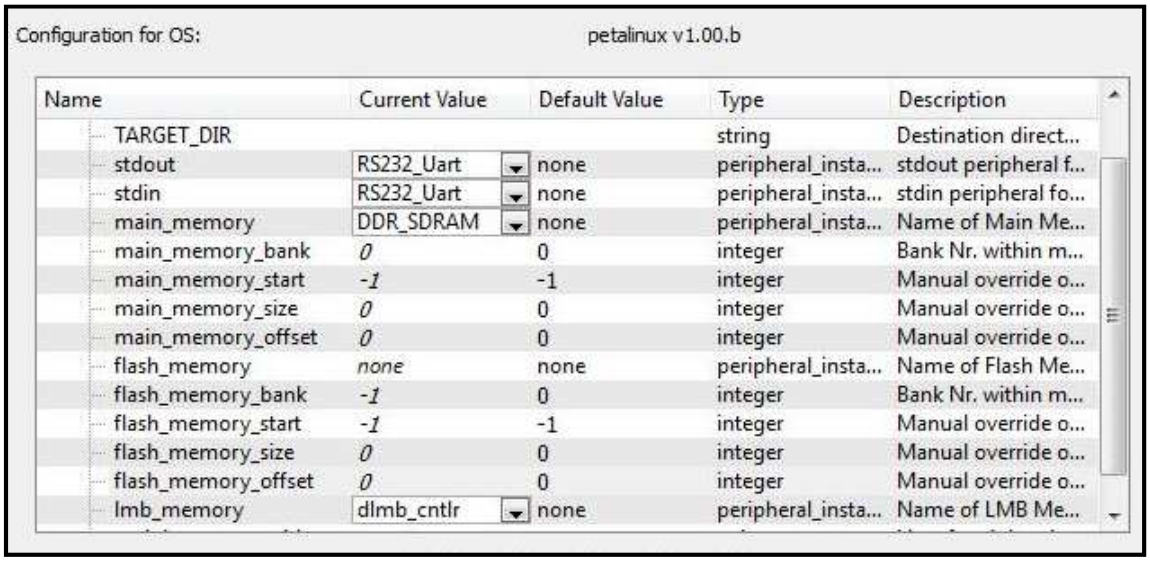


Figura 3.44: Ventana selección parámetros de Petalinux

A continuación se pincha en OK.

Después de todo lo anterior se pasa a la generación de las librerías y archivos, que ayudan a que Microblaze y Petalinux, sean compatibles, y funcionen correctamente. Para ello, se pincha en `Software → Generate Libraries and BSPs`. Con esto comienza el proceso de generación.

3.5 Diseño del software de control para la cámara estéreo

Para poder controlar la cámara estéreo desde el sistema operativo Petalinux que se introduce en el micro-controlador se desarrolla un pequeño programa de control en código C, para las funciones básicas de encendido, apagado y configuración.

Para ello se utilizan las funciones en C, ofrecidas por el micro-controlador Microblaze, y que ayudan a la comunicación con los periféricos a través del PLB.

Estas funciones son:

`XIo_In32 (dirección del periférico)` → con esta función se puede obtener el dato de 32 bits, almacenado en esa dirección.

Por ejemplo cuando se quiere leer el registro estado de la cámara2, de la cámara estéreo, se pone lo siguiente:

`XIo_In32 (0x82000200)`, donde 0x82000200 es la dirección asignada al registro estado de la cámara2, esta función devuelve el dato de 32 bits del registro estado.

`XIo_Out32 (dirección del periférico, dato)` → con esta función se puede escribir un dato de 32 bits, en la dirección del periférico.

Por ejemplo cuando se quiere parar la cámara1, de la cámara estéreo, se pone lo siguiente:

`XIo_Out32 (0x82000004, 0x00000004)`, donde 0x82000004 es la dirección asignada al registro control de la cámara1, y 0x00000004 es el dato que se quiere escribir en esa dirección y que indica al sensor que se pare.

Por tanto, con la ayuda de estas dos funciones se procede al desarrollo del programa.

3.5.1 Desarrollo del programa de control

El software está compuesto por las siguientes funciones:

Inicialización → Esta función para a todos los sensores, y los configura con sus valores por defecto.

Menú → Esta función permite seleccionar la actividad que se quiere realizar con la cámara estéreo, encenderla, apagarla, configurarla o salir del programa.

Configuración → En esta función se selecciona la resolución y el zoom de la imagen que se quiere captar.

Read_img → En esta función se puede seleccionar las imágenes que se quieren leer, (las imágenes captadas por la camara1, la camara2 y/o la camara3).

Esta función guarda las imágenes captadas por la cámara estéreo, en matrices.

Alm_img → En esta función se puede seleccionar las cámaras que se quieren activar. Posteriormente esta función activa las cámaras seleccionadas, y avisa cuando las imágenes están almacenadas.

Esta función se mantiene en ejecución hasta que el usuario quiera lo contrario, cuando esto ocurra se llama a la función “parada_sensores”, para desactivar los sensores.

Parada_sensores → Esta función tiene el cometido de enviar la orden de desactivación de las cámaras y esperar a que estas se paren.

Esta función es utilizada dentro de las funciones de “inicialización”, “read_img”, “alm_img” y se ejecuta también cuando el usuario quiera abandonar la aplicación.

En la figura siguiente se muestra el diagrama de flujo, para visualizar el funcionamiento de este programa.

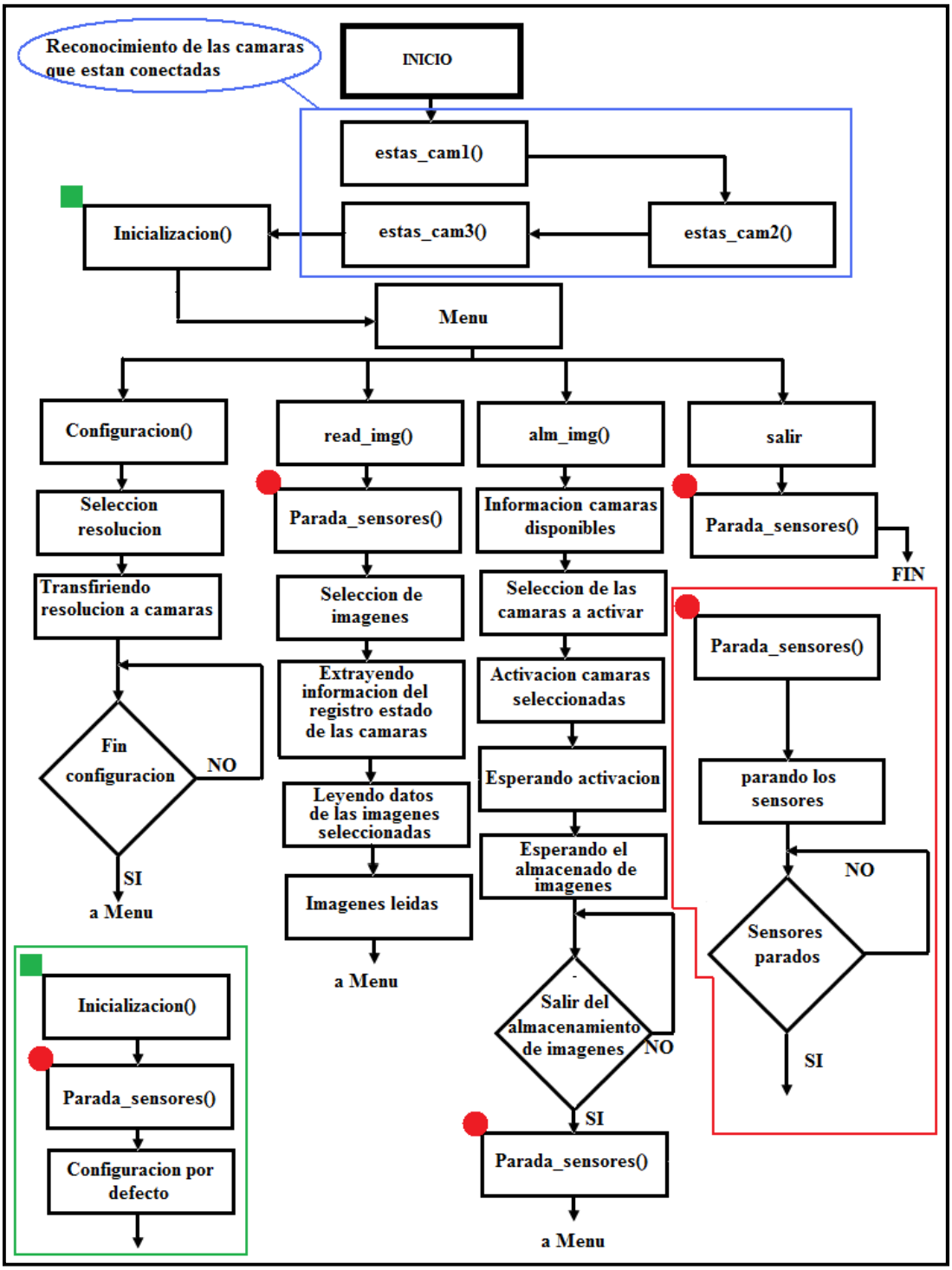


Figura 3.45: Diagrama de flujo del programa control cámara

3.5.2 Introducción del software en el Microblaze

Para incluir el programa anterior en el proyecto creado con la herramienta XPS de Xilinx se procede de la siguiente forma.

Primer paso: En el `work space` del XPS, se selecciona el menú `Applications`, y a continuación se pincha con el ratón en `"Add software application Project"`.

Segundo paso: En la ventana que se aparece, se da nombre al software, en este caso se llama `cntrl_cam`. Se pincha en OK, y a continuación sale el software en el `work space`, junto con el software `TestApp_Memory_microblaze_0` y `TestApp_Peripheral_microblaze_0`.

Tercer paso: Se pincha con el botón izquierdo del ratón en la opción `Source`, del software creado, a continuación se despliega un menú donde se elige la opción `Add Existing File` y se añade al software el código C del control de la cámara, que en este caso se ha llamado `'main_cam.c'`.

Cuarto paso: Se pincha con el botón derecho del ratón sobre `Project: Cntrl_cam`, aparece un menú en el cual se selecciona la opción `Build Project`.

Al realizar esta acción comienza el proceso de compilación.

3.6 Cargador de Petalinux

Esta parte del proyecto se obtiene integró del proyecto "*Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas*", en el se puede obtener el proceso en detalle, así como su funcionamiento, para añadir un programa, que guarda y ejecuta la imagen del Kernel de Petalinux desde la tarjeta Compact Flash.

Se realizan modificaciones en el código del programa llamado `cf_loader`, facilitado por el proyecto mencionado anteriormente. Estas modificaciones consisten en cambios en los nombramientos de algunos parámetros, que difieren de los actuales. El código modificado se encuentra en el CD facilitado en este documento.

Para crear el archivo `system.ace`, que forma parte del proceso del cargador de Petalinux, no se ha encontrado la forma de realizarlo de la misma forma según se expone en el proyecto de referencia.

Debido seguramente a que se utiliza una versión más actual de la herramienta XPS, de la que utiliza el proyecto de referencia.

Por tanto, se describe el modo para crear el archivo `system.ace`.

Se pincha en `Project` → `Launch Xilinx Bash Shell`, se aparece una ventana parecida al "`cmd`" de Windows.

En ella se escribe la siguiente línea de comandos cuya función es la misma que la realizada por los pasos descritos en el proyecto de referencia, para la creación del archivo system.ace [9].

```
/Documents/xps_camV4_3S$ xmd -tcl genace.tcl -target mdm –
hw              implementation/download.bit              –elf
cf_loader/executable.elf       -board       ml402       -ace
download/system.ace
```

Con esta línea de comandos se crea el archivo `system.ace`, que está ubicado dentro de la carpeta donde está guardado todo el proyecto.

Posteriormente se debe generar un documento `imagen.bin`, el cual, contiene la imagen del Kernel de Petalinux. Este proceso de generación esta detalladamente explicado en el proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"* concretamente en el capítulo 3.3 *Desarrollo Sotfware* [1].

Con esto se termina el diseño de la cámara estéreo, la cual se llama xps_camV4_3S.

Capítulo 4

Descripción del sistema

4.1 Introducción

En este capítulo se mostrará una descripción de los recursos utilizados por cada módulo del periférico de la cámara estéreo. Se destacarán los detalles de los componentes más utilizados y se llegará a la conclusión de que el diseño no consume demasiados recursos a excepción de algunos componentes como las RAMB16s que se necesitarán para la creación de las memorias externas.

Se realizará un análisis de los recursos utilizados por el periférico actualizado "PLB_cypress_usb". Se demostrará que este periférico tampoco consume demasiados recursos.

También se realizará una descripción de los recursos utilizados por todo el conjunto, es decir, de los utilizados por Microblaze y sus periféricos, incluyendo la cámara estéreo. Se observará que los recursos utilizados por el sistema completo son en su mayoría, utilizados por Microblaze y los periféricos necesarios, para su buen funcionamiento.

4.2 Recursos utilizados por el periférico cámara estéreo

Para una descripción de los recursos más detallada se realiza un análisis independiente, de los recursos utilizados por cada módulo que compone a este periférico.

4.2.1 PLB_camara

Se observa en la siguiente tabla el número y el tipo de celdas que está utilizando la PLB_camara.

Design Statistics:			
IOs	244		
Cell Usage :			
BELS	753	FlipFlops/Latches	333
GND	1	FD_1	2
INV	12	FDC	122
LUT1	25	FDCE	137
LUT2	79	FDE	9
LUT3	207	FDP	12
LUT3_D	1	FDPE	3
LUT4	243	FDR	26
LUT4_D	3	FDRS	9
LUT4_L	1	FDRSE	7
MUXCY	42	FDS	6
MUXF5	96	Clock Buffers	2
MUXF6	11	BUFG	2
MUXF7	3	DCM_ADVs	1
VCC	1	DCM_ADV	1
XORCY	28		

Tabla 4.1: Recursos utilizados por el periférico PLB_camara

Se destaca la diferencia del número de los `Flipflops FDC` y `FDCE`. Los `FDC` son registros de tipo – `D`, la mayoría de ellos en el componente “interfaz_PLBcam”. Los `FDCE` son registros del mismo tipo que los anteriores con la diferencia de que estos contienen un bit de `Enable`. Estos se utilizan en su mayoría en el componente “interfaz_PLBcam”.

También se destaca la cantidad de "Look-up tables" utilizadas, en concreto las LUTs de 3 y 4 entradas.

El componente DCM_ADV es utilizado por el módulo "DCM8" del componente "cámara" de este periférico.

En la siguiente tabla se observa los porcentajes de utilización de los recursos en relación a los disponibles en la placa Virtex 4 (ML402).

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	313 out of 15360	2%
Number of Slices Flip Flop	333 out of 30720	1%
Number of 4 input LUTs	571 out of 30720	1%
Number of IOs	244	
Number of bonded IOBs	0 out of 448	0%
Number of GCLKs	2 out of 32	6%
Number of DCM_ADVs	1 out of 8	12%

Tabla 4.2: Dispositivos utilizados por la PLB_camara

4.2.2 Control_imagenes_ext

Los recursos que utiliza este periférico perteneciente a la cámara estéreo, se muestran en la siguiente tabla.

Design Statistics:		LUT4_D	6
IOs	265	LUT4_L	4
		MUXCY	16
Cell Usage :		MUXF5	1
BELS	143	VCC	1
GND	1	XORCY	17
INV	1	FlipFlops/Latches	57
LUT1	16	FDC	21
LUT2	31	FDCE	3
LUT3	2	FDR	26
LUT3_D	9	FDRE	7
LUT3_L	2		
LUT4	36		

Tabla 4.3: Recursos utilizados por el periférico Control_imagenes_ext

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	62 out of 15360	0%
Number of Slices Flip Flop	57 out of 30720	0%
Number of 4 input LUTs	107 out of 30720	0%
Number of IOs	265	
Number of bonded IOBs	0 out of 448	0%

Tabla 4.4: Dispositivos utilizados por Control_imagenes_ext

Se observa que este periférico no consume muchos recursos en comparación con el periférico “PLB_camara”. En relación a lo disponible en la placa se observa que el consumo es inexistente.

4.2.3 DCM48mhz

Este periférico demanda el componente `DCM_ADV`, ya que este periférico se encarga de generar una señal de reloj de 48 MHz. La utilización de los demás componentes es inexistente como se observa en la siguiente figura.

Design Statistics	
IOs	4
Cell Usage:	
BELS	1
GND	1
Clock Buffers	1
BUFG	1
DCM_ADVs	1
DCM_ADV	1

Tabla 4.5: Recursos utilizados por el periférico dcm48mhz

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	0 out of 15360	0%
Number of IOs	4	
Number of bonded IOBs	0 out of 448	0%
Number of GCLKs	1 out of 32	3%
Number of DCM_ADVs	1 out of 8	12%

Tabla 4.6: Dispositivos utilizados por dcm48mhz

4.2.4 Nimagen

Como en los periféricos anteriores se ha conseguido que la demanda de recursos sea inexistente en relación con la disponible en la placa.

Design Statistics:		FlipFlops/Latches	35
IOs	77	FD	17
		FDE	16
Cell Usage :		FDS	1
BELS	198	FDSE	1
GND	1		
LUT2	40		
LUT3	28		
LUT4	39		
MULT_AND	2		
MUXCY	42		
MUXF5	2		
XORCY	44		

Tabla 4.7: Recursos utilizados por el periférico nimagen

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	61 out of 15360	0%
Number of Slices Flip Flop	35 out of 30720	0%
Number of 4 input LUTs	107 out of 30720	0%
Number of IOs	77	
Number of bonded IOBs	0 out of 448	0%

Tabla 4.8: Dispositivos utilizados por nimagen

Destacan la utilización de `LUTs` de 2 y 4 entradas, multiplexores y registros de tipo `–D`.

4.2.5 Memoria_externa

En este caso el periférico está formado por una memoria de 800x600 con 8 bits de datos. Se menciona esto porque evidentemente los recursos utilizados por este periférico dependen del tamaño de la memoria que se le imponga. El dispositivo clave que utiliza este periférico es el `RAMB16.` Se observa en la siguiente figura el número de dispositivos de este tipo que utiliza.

Design Statistics		LUT3	19
IOs	37	LUT4	9
		MUXF5	8
Cell Usage:		FlipFlops/Latches	5
BELS	38	FDE	5
GND	1	RAMS	38
LUT2	1	RAMB16	38

Tabla 4.9: Recursos utilizados por el periférico memoria externa

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	15 out of 15360	0%
Number of Slices Flip Flop	5 out of 30720	0%
Number of 4 input LUTs	29 out of 30720	0%
Number of IOs	37	
Number of bonded IOBs	0 out of 448	0%
Number of FIFO16/RAMB16s	38 out of 192	19%
Number used as RAMB16s	38	

Tabla 4.10: Dispositivos utilizados por la memoria externa

Hay que destacar el porcentaje de `RAMB16s` que utiliza el periférico en relación con la disponible en la placa. Este periférico sólo guarda una imagen, por tanto para guardar dos o tres, habría que multiplicar el número de dispositivos por dos o tres, respectivamente.

4.3 Recursos utilizados por el periférico PLB_cypress_usb

Se observa que destacan la utilización de `LUTs` de 4 entradas, y de `FlipFlops FDR`, `FlipFlops` de tipo `–D`, con `Reset` síncrono.

Destacar también que con relación a los dispositivos disponibles en la placa este periférico no consume.

Design Statistics:		MULT_AND	20
IOs	356	MUXCY	77
		MUXCY_L	6
Cell Usage :		MUXF5	7
BELS	469	VCC	1
GND	1	XORCY	58
INV	7	FlipFlops/Latches	443
LUT2	36	FD	1
LUT2_L	2	FDR	247
LUT3	76	FDRE	123
LUT3_D	1	FDRS	9
LUT3_L	4	FDRSE	9
LUT4	165	FDS	44
LUT4_D	5	FDSE	10
LUT4_L	3		

Tabla 4.11: Recursos utilizados por el periférico PLB_cypress_usb

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	306 out of 15360	1%
Number of Slices Flip Flop	379 out of 30720	1%
Number of 4 input LUTs	299 out of 30720	0%
Number of IOs	356	
Number of bonded IOBs	0 out of 448	0%
IOB Flip Flops	64	

Tabla 4.12: Dispositivos utilizados por PLB_cypress_usb

Por tanto en términos generales, los periféricos añadidos al sistema no generan grandes complicaciones de disponibilidad, a excepción de los dispositivos `RAMB16s`, utilizados por la memoria externa y que limitan el tamaño de la imagen que se quiere almacenar.

4.4 Recursos utilizados por la cámara estéreo y Microblaze

A continuación, se muestra en la siguiente figura los recursos utilizados por el sistema completo, destacando el porcentaje de utilización de los componentes de la `RAMB16s`, que están proporcionalmente relacionados con el número y tamaño de las memorias externas de las "PLB_camaras".

Los porcentajes medios como los 38% de los `Number of Slices` o 27% de los `LUTs` de 4 entradas, son en su mayoría recursos demandados por Microblaze y sus periféricos como el RS232, ya que los periféricos diseñados en este proyecto no consumen en su mayoría casi ningún recurso, como se ha visto anteriormente, a excepción, de las memorias externas.

Design Statistics:		FDPE	9
IOs	150	FDR	1493
		FDR_1	1
Cell Usage :		FDRE	1850
BELS	10030	FDRE_1	1
BUF	11	FDRS	100
GND	31	FDRSE	145
INV	192	FDS	174
LUT1	210	FDS_1	1
LUT2	1043	FDSE	141
LUT2_D	26	IDDR	32
LUT2_L	31	ODDR	73
LUT3	2511	RAMS	463
LUT3_D	34	RAM16X1D	290
LUT3_L	56	RAMB16	173
LUT4	3259	Shift Registers	190
LUT4_L	91	SRL16	125
LUT4_D	132	SRL16_1	2
MULT_AND	60	SRLC16E	8
MUXCY	682	Clock Buffers	14
MUXCY_L	248	BUFG	13
MUXF5	831	BUFGP	1
MUXF6	39	IO Buffers	149
MUXF7	11	IBUF	43
MUXF8	1	IBUFG	1

OR2	3	IOBUF	58
VCC	20	OBUF	46
XORCY	508	OBUFDS	1
FlipFlops/Latches	6799	DCM_ADVs	5
FD	1181	DCM_ADV	5
FD_1	10	DSPs	3
FDC	469	DSP48	3
FDC_1	5	Others	35
FDCE	444	BSCAN_VIRTEX4	1
FDE	600	IDELAY	32
FDE_1	5	IDELAYCTRL	2
FDP	65		

Tabla 4.13: Recursos utilizados por la cámara estéreo y Microblaze

Device utilization summary:		
Selected Device	4vsx35ff668-10	
Number of Slices	5923 out of 15360	38%
Number of Slices Flip Flop	6677 out of 30720	21%
Number of 4 input LUTs	8355 out of 30720	27%
Number used as logic	7585	
Number used as Shift registers	190	
Number used as RAMs	580	
Number of IOs	150	
Number of bonded IOBs	150 out of 448	33%
IOB Flip Flops	122	
Number of FIFO16/RAMB16s	173 out of 192	90%
Number used as RAMB16s	173	
Number of GCLKs	14 out of 32	43%
Number of DCM_ADVs	5 out of 8	62%
Number of DSP48s	3 out of 192	1%

Tabla 4.14: Dispositivos utilizados por la cámara estéreo y Microblaze

Capítulo 5

Pruebas

5.1 Introducción

En este capítulo se realizarán las simulaciones para observar si los códigos diseñados funcionan correctamente. Para realizar un análisis más detallado se realizará la simulación de los módulos más importantes por separado. En primer lugar, se realizará la simulación del módulo “interfaz_PLBcam” (módulo encargado del control de la comunicación de la cámara estéreo con el Microblaze a través del bus PLB) y se comprobará los procesos de escritura y lectura, para observar que las señales que actúan de árbitro en la comunicación se activan en los instantes y en el orden correctos. Más tarde se realizará la simulación del funcionamiento del módulo “PLB_cam”, para comprobar los procesos de escritura y lectura de sus registros internos. Se observará a su vez las operaciones de configuración, encendido, apagado, lectura de imágenes y con las indicaciones de fin de almacenamiento de imágenes.

El siguiente módulo se simulará el “control_imagenes_ext”, y se comprobará el almacenamiento ordenado de las imágenes y la lectura de los datos de las imágenes.

Por último, se realizará la simulación de los módulos “nimagen” y “mem_externa”, y se observará el buen funcionamiento de los procesos de conversión, escritura y lectura de datos.

Para acabar, se realizará la comprobación del software mediante la observación de la correcta ejecución del Sistema Operativo y la comunicación satisfactoria entre la cámara estéreo y Microblaze, a través del bus PLB.

5.2 Simulaciones hardware de la cámara estéreo

5.2.1 Interfaz_PLBcam

- Proceso de Escritura

Al módulo "interfaz_PLBcam", se le presentan las señales `PLB_ABus`, `PLB_PAValid`, `PLB_rnw` y `PLB_wrDbus`.

En esta simulación, la señal `PLB_ABus` contiene el valor x82000004 → registro control de cámara 1; y `PLB_wrDbus` que contiene el valor x00000002 → bit 1 del registro control a '1' lógico, indicando encendido de la cámara.

`PLB_rnw` está a '0' lógico, pues es el proceso de escritura.

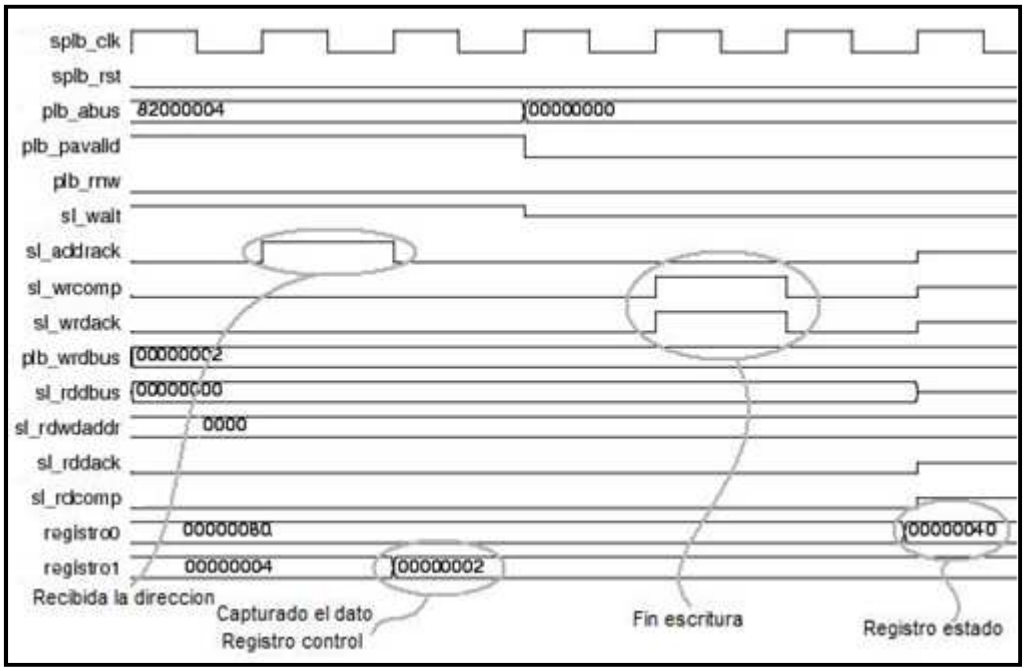


Figura 5.1: Simulación proceso de escritura

Se ha observado que las señales del bus, están activadas de forma correcta, comparándolas con las mostradas en la figura 3.7 del capítulo 3, del proceso de escritura. Estas señales son `Sl_wrdack`, `Sl_wait`, `Sl_addrack` y `Sl_wrComp`.

Terminado el proceso de escritura se ha observado, que el registro control (registro 1) de la cámara 1, ha pasado a tener el valor de x00000002 y el registro estado (registro 0) ha pasado a tener un valor de x00000040, indicando que la cámara esta activada. Este registro estado, posteriormente es leído por el micro-controlador.

- Proceso de Lectura

Como el anterior proceso al módulo “interfaz_PLBcam” se le presentan las señales `PLB_ABus`, `PLB_PAValid` y `PLB_rnw`.

Ahora `PLB_ABus` tiene un valor de x82000000 → registro estado de cámara 1.

`PLB_rnw` está a ‘1’ lógico, pues el proceso de lectura.

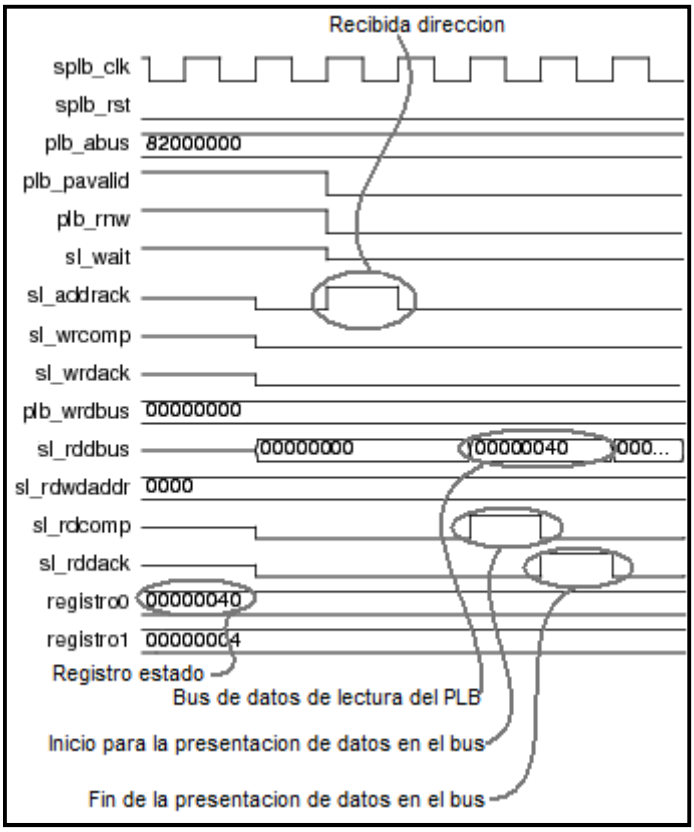


Figura 5.2: Simulación proceso de lectura

Se ha observado que las señales del bus, están activadas de forma correcta, comparándolas con las mostradas en la figura 3.6 del capítulo 3, del proceso de lectura.

Estas señales son `Sl_addrack`, `Sl_wait`, `Sl_rdDbus`, `Sl_rdComp` y `sl_rdDAck`.

Durante el proceso se ha observado que el valor del registro 0, se carga en el bus a través de la señal `Sl_rdDbus`.

- Funcionamiento de dos cámaras simultáneamente

En este apartado se quiere demostrar que cuando el micro-controlador quiere mandar una orden a la cámara 1, sólo tiene que indicar la dirección del registro control de la cámara 1 (x82000004), y la cámara 2 no hará caso a esta indicación ya que su dirección no se corresponde con la suya que es x82000104.

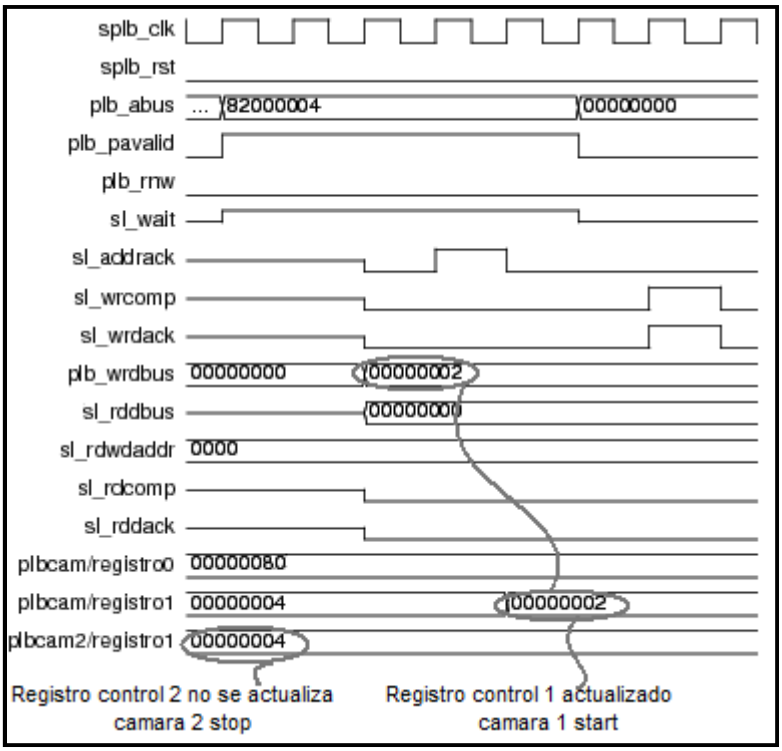


Figura 5.3: Simulación funcionamiento simultaneo de cámaras

Se ha observado en la figura que el valor del registro control de la cámara 1, actualiza su valor, mientras que el de la cámara 2 mantiene el valor anterior.

Indicar que las señales que controlan el bus PLB, por parte de la cámara 2, en este instante están conectadas en modo de alta impedancia 'Z' (en la simulación, en el funcionamiento normal permanecerán a '0' lógico), para no interferir con las señales de la cámara 1.

5.2.2 Funcionamiento de la PLB_cam

Lectura y Escritura de registros de la cámara

- Configuración:

Inicio de la configuración: Se ha observado como a la cámara a través de los registros de control, resolución y zoom, inicia el proceso de configuración, poniendo la señal `i2c_config` a '1'.

También se ha observado que el bus i2c, envía los datos al sensor correctamente.

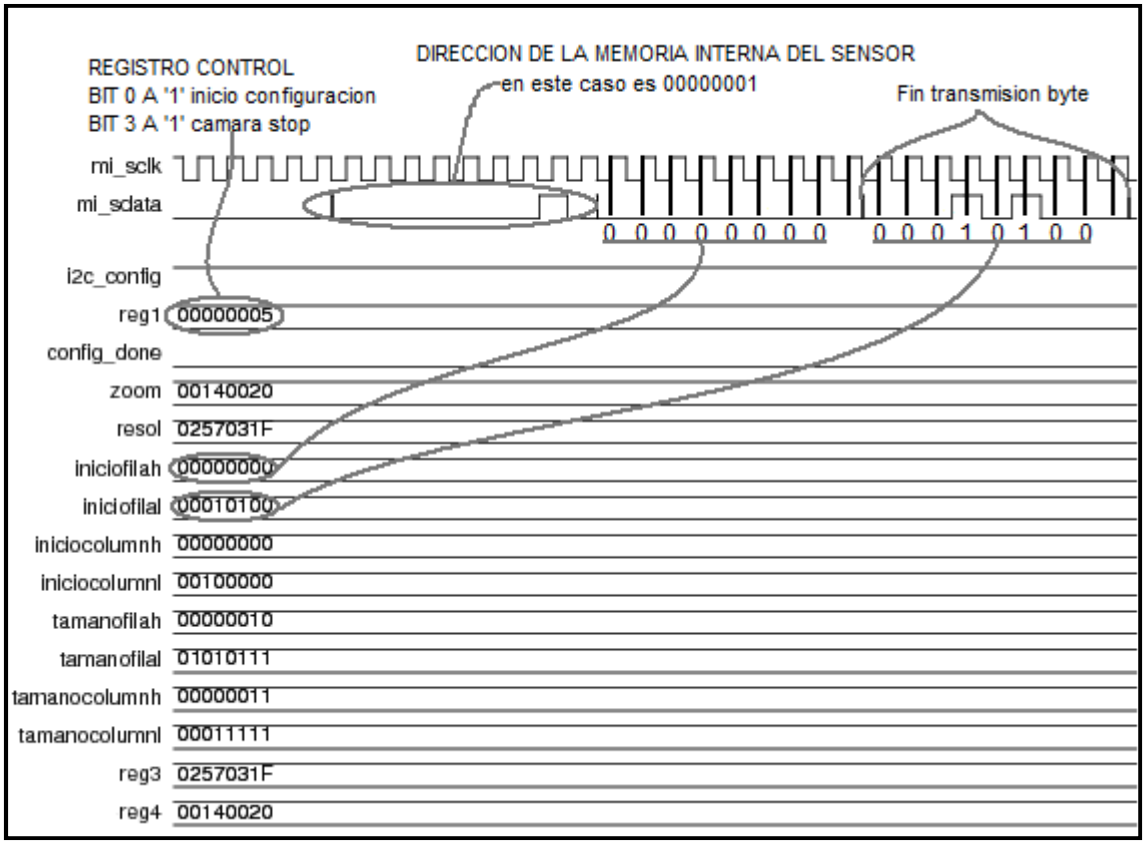


Figura 5.4: Simulación inicio configuración sensor

Fin de la configuración: Se ha observado que cuando la cámara ha terminado de configurar al sensor, pone la señal `config_done` a ‘1’.

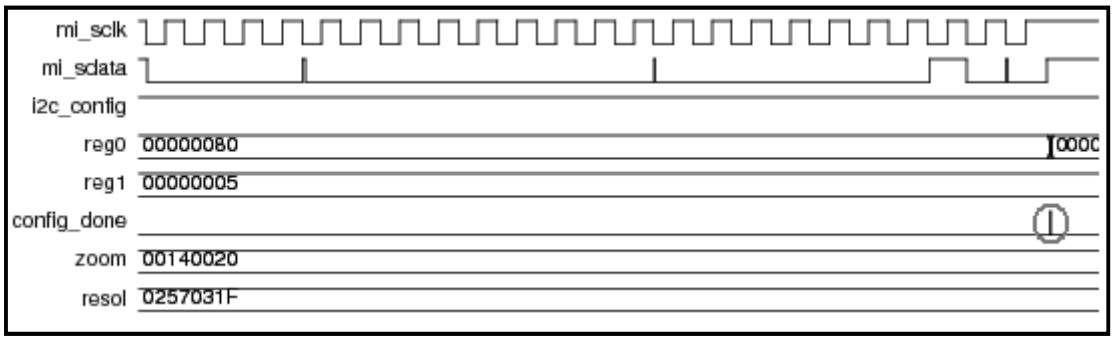


Figura 5.5: Simulación fin configuración sensor

Se realiza un zoom en la parte donde la señal `config_done` esta a ’1’, para observar que el registro de estado de la cámara se actualiza, poniendo el bit 4 a ‘1’.

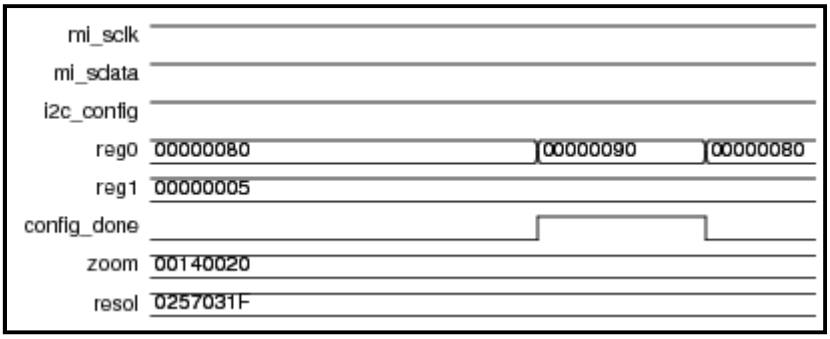


Figura 5.6: Zoom de simulación fin configuración sensor

- Encendido

En la figura se ha observado como la cámara recibe del micro-controlador la orden de `"start"` de la cámara por medio del registro control (registro 1), y como ésta responde a sus órdenes activándose (se ha observado que el reloj del sensor `cam_pixclk` se pone en funcionamiento), para la captación de los datos.

También se ha observado que una vez activada, la cámara escribe en su registro de estado un '1' en el bit 6, indicándole al micro-controlador que el sensor esta activado.

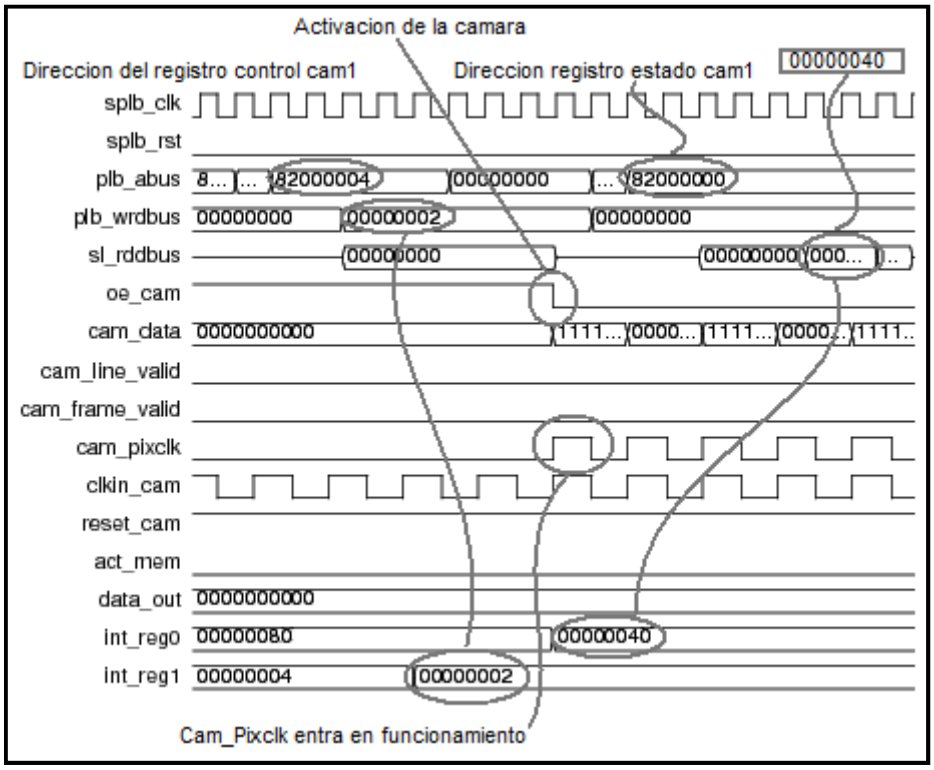


Figura 5.7: Simulación encendido sensor

- Apagado

Para realizar un análisis más detallado de este proceso se muestran dos imágenes.

En la primera se ha observado que el micro-controlador, indica a la cámara que se desactive a través de su registro de control, en ese instante la cámara esta en medio del proceso de captación, (se ha observado que la señal `cam_frame_valid` esta a '1' lógico), por ello la cámara permanecerá en espera hasta que la imagen se complete. Por otro lado, el micro-controlador, esta muestreando el registro estado de la cámara para saber cuando la cámara está en estado `"stop".`

En la segunda imagen se ha observado que una vez terminada la captación de la imagen, la "PLB_camara", desactiva la cámara, y lo indica en su registro estado, el cual, es leído por el micro-controlador.

Imagen 1

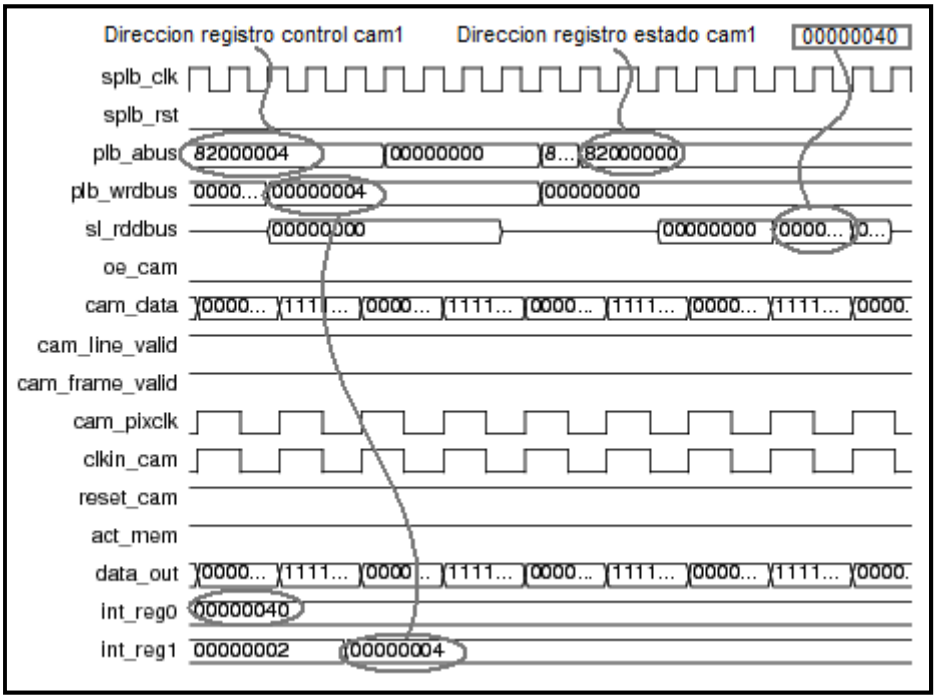


Figura 5.8: Imagen 1 de simulación apagado de sensor

Imagen 2

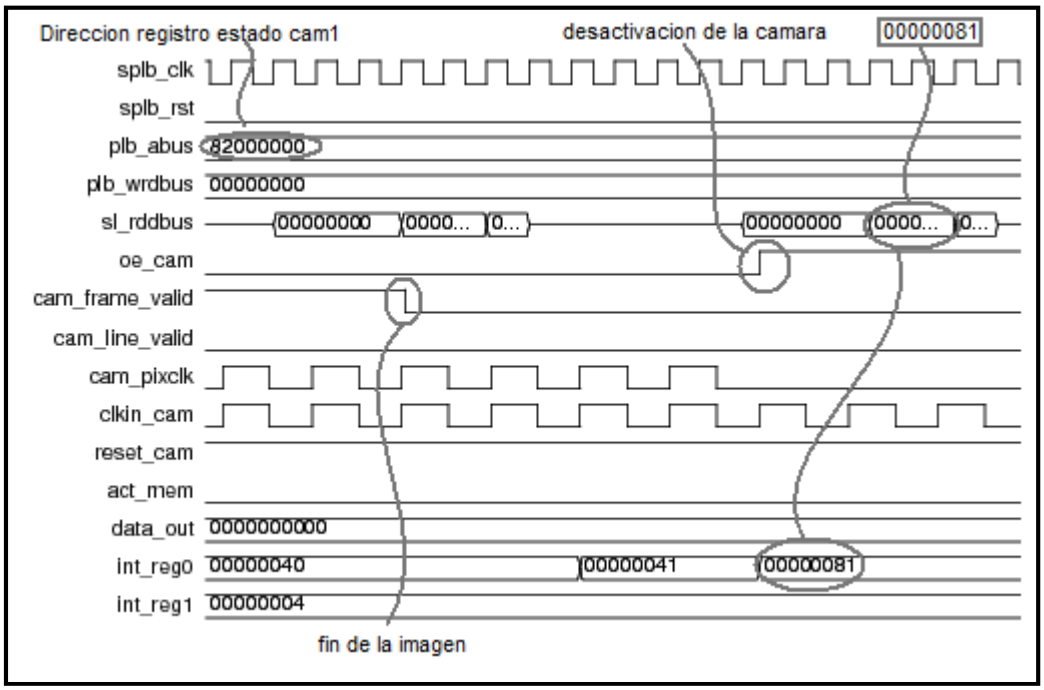


Figura 5.9: Imagen 2 de simulación apagado de sensor

Final del almacenamiento de imagen

En este caso se ha demostrado que cuando la cámara termina de almacenar una imagen, se lo indica al micro-controlador, activando el bit 0 de su registro estado.

También se ha observado que cuando hay varias cámaras conectadas, en este caso dos, cuando terminan de almacenar sus imágenes, éstas activan simultáneamente el bit 1 de sus registros de estado, indicándole al micro-controlador que todas las imágenes están almacenadas.

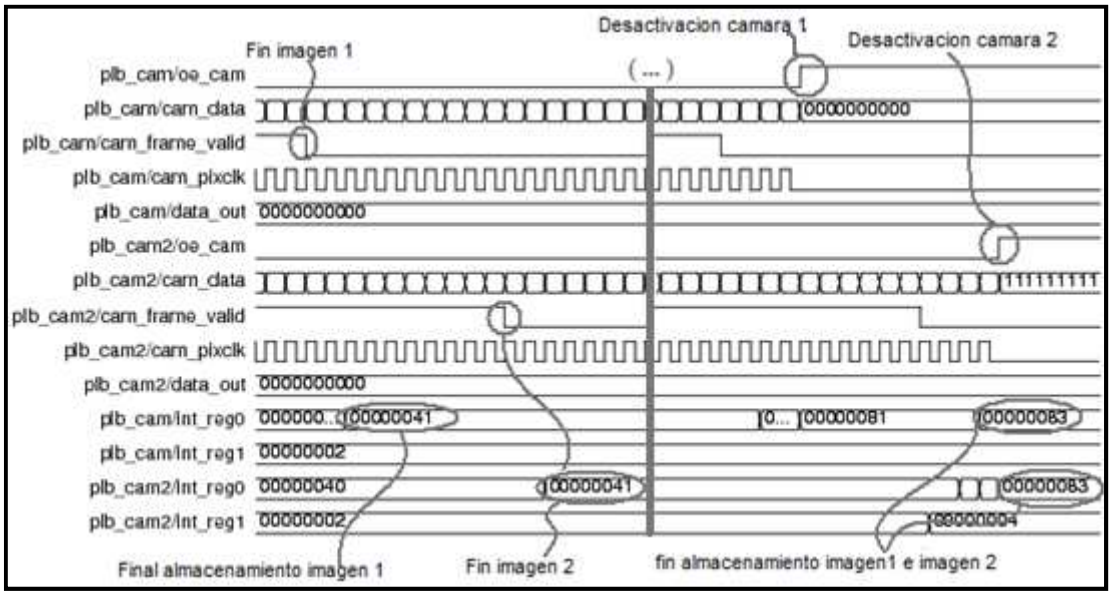


Figura 5.10: Simulación fin de almacenamiento de imágenes

Lectura de la imagen almacenada

Se ha observado en esta figura, que el micro-controlador envía la orden de leer la imagen almacenada activando a '1' el bit 5 del registro de control, la cámara a su vez recibe esta orden, activando la señal de lectura, esta señal activa la función de lectura del módulo "control_imagenes_ext".

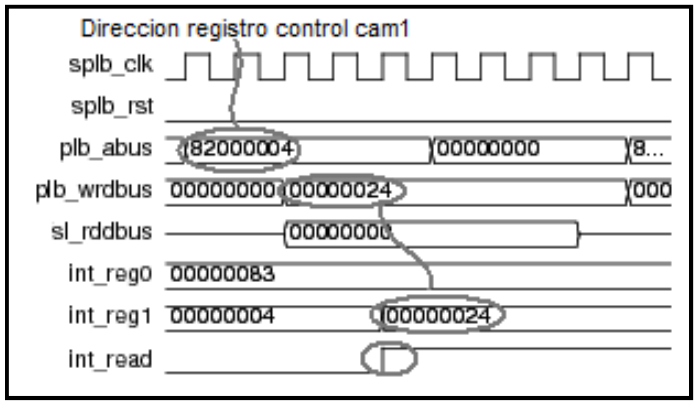


Figura 5.11: Simulación de lectura de las imágenes almacenadas

5.2.3 Funcionamiento Control_imagenes_ext

Almacenamiento ordenado de las imágenes

Se ha observado en la figura siguiente, como este módulo recibe las señales de control de todos los sensores conectados al sistema, y ésta va presentando las direcciones de las imágenes en orden para guardarlas en sus respectivas memorias.

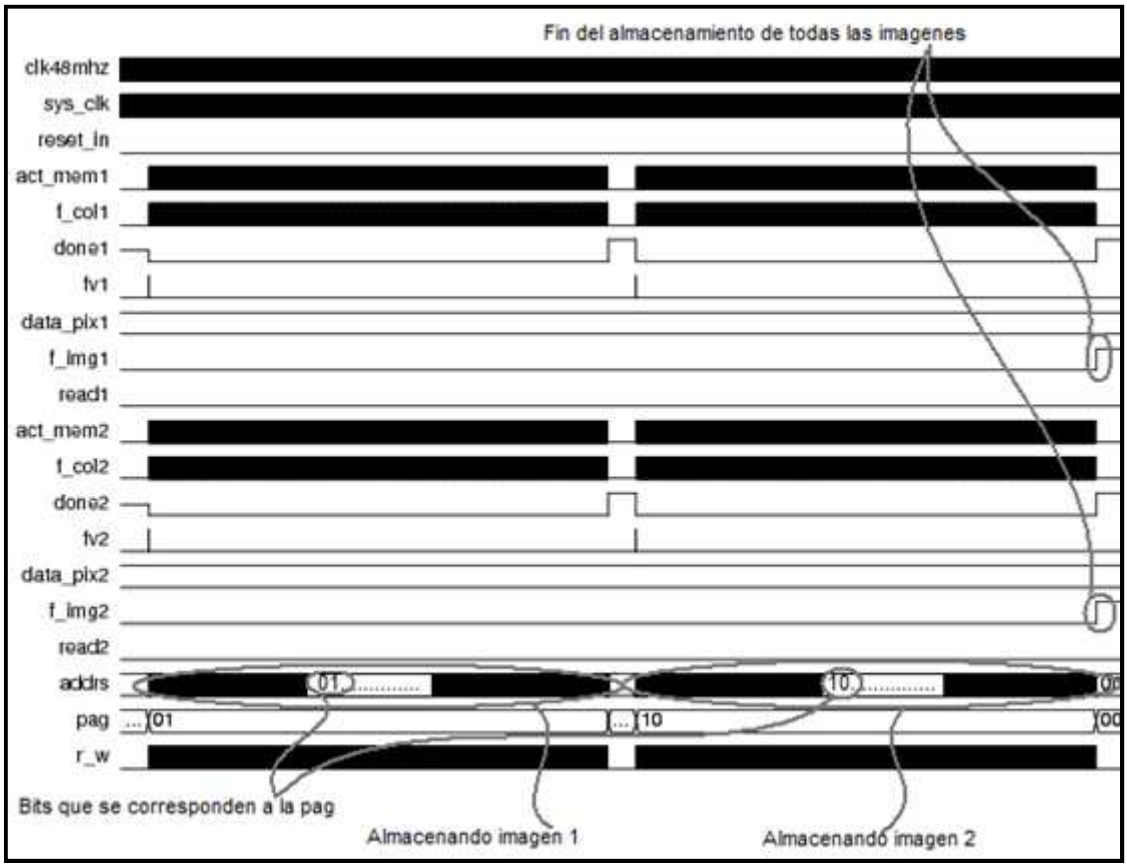


Figura 5.12: Simulación de almacenamiento ordenado de imágenes

Se ha observado también que en el bus de direcciones de este módulo, llamado `addrs`, sus dos últimos bits indican la página, es decir, en qué memoria se almacena la imagen.

En este caso solo están conectados al sistema dos sensores, la página '01' se corresponde con el almacenamiento de la imagen 1 y la página '10' se corresponde con el almacenamiento de la imagen 2.

Lectura de los datos de la imagen

Se ha observado que el módulo recibe la señal `read1` y `read2` de la "PLB_cam1" y "PLB_cam2", respectivamente.

Se ha observado que este módulo, extrae de la memoria los datos de las imágenes almacenadas anteriormente.

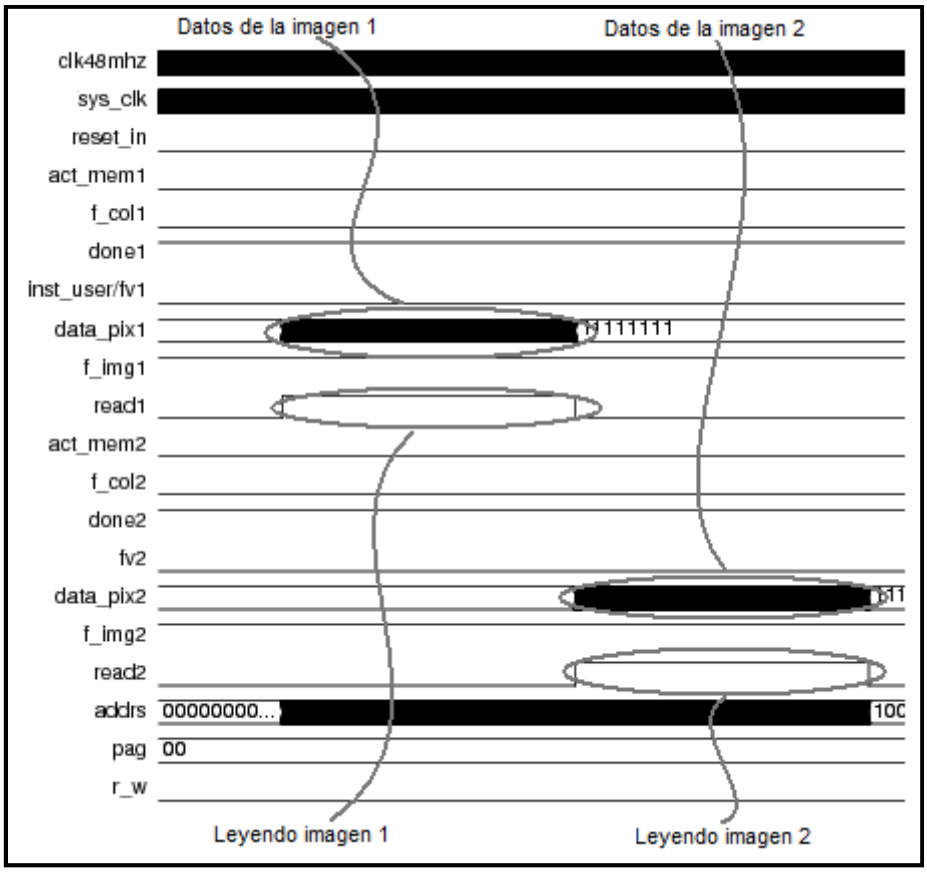


Figura 5.13: Simulación lectura de los datos de imagen

5.2.4 NimagenN y mem_800x600_8BITS

Escritura de la imagen

Se ha observado la conversión de los datos provenientes de la cámara compuestos por 10 bits a 8 bits para el almacenamiento en la RAM externa.

Se ha observado que cuando las direcciones que provienen del módulo "control_imagen_ext", no se corresponden con las del módulo "nimagen2", este está totalmente desactivado, al igual que el módulo "mem_800x600_8bits", que almacena las imágenes de la cámara 2, en cambio el módulo "nimagen1" y "mem_800x600_8bits" si están activados.

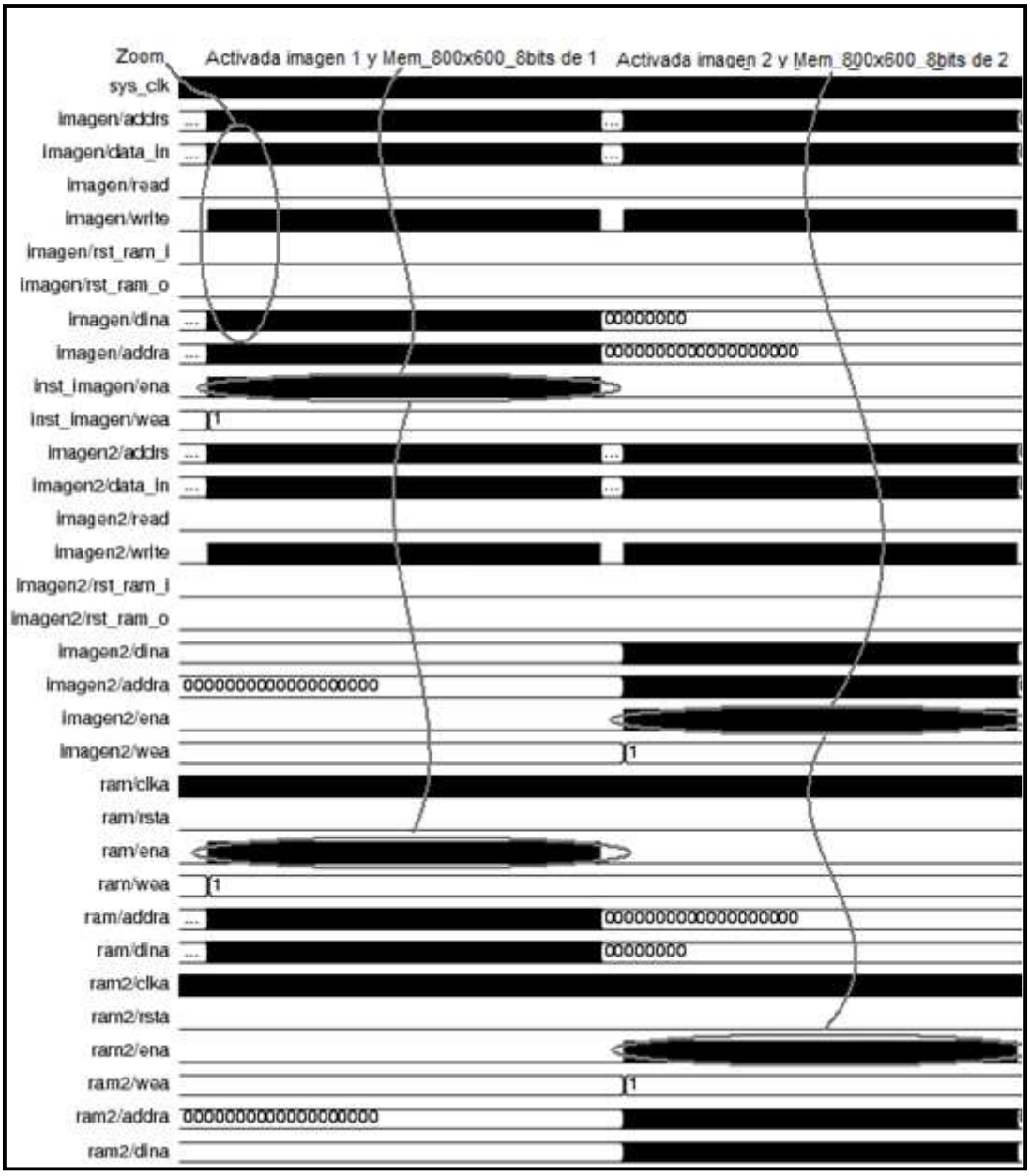


Figura 5.14: Simulación escritura de la imagen

Zoom

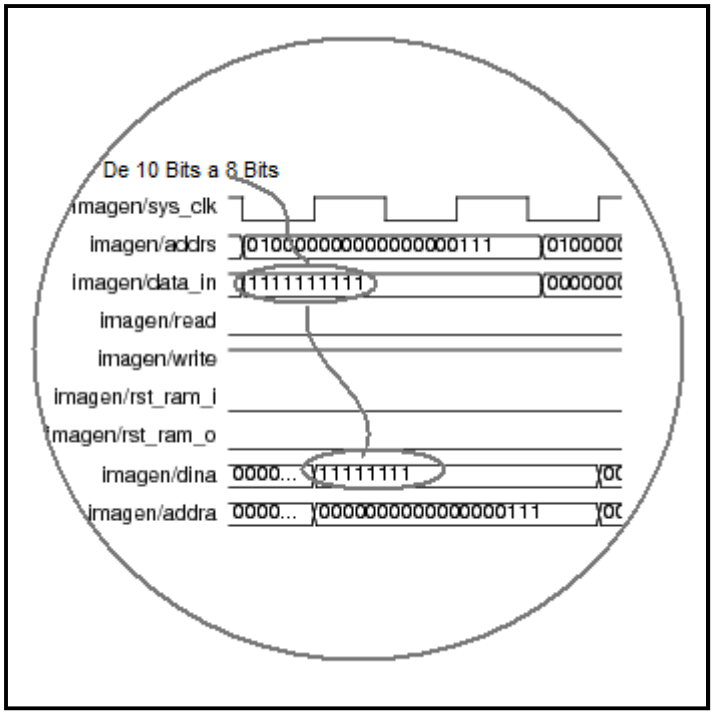


Figura 5.15: Zoom simulación escritura de la imagen

Lectura de la imagen

Se ha observado que en el módulo nimagen entra las direcciones de memoria que se quieren leer y esta devuelve los valores de 8 bits correspondientes a esa dirección.

Al igual que en el proceso de escritura, el módulo “nimagen 1” y “mem_800x600_8bits” de la cámara 1, se activan para las direcciones de memoria que terminen en ‘01’y lo mismo para el módulo “nimagen2” y “mem_800x600_8bits”, pero terminando las direcciones en ‘10’.

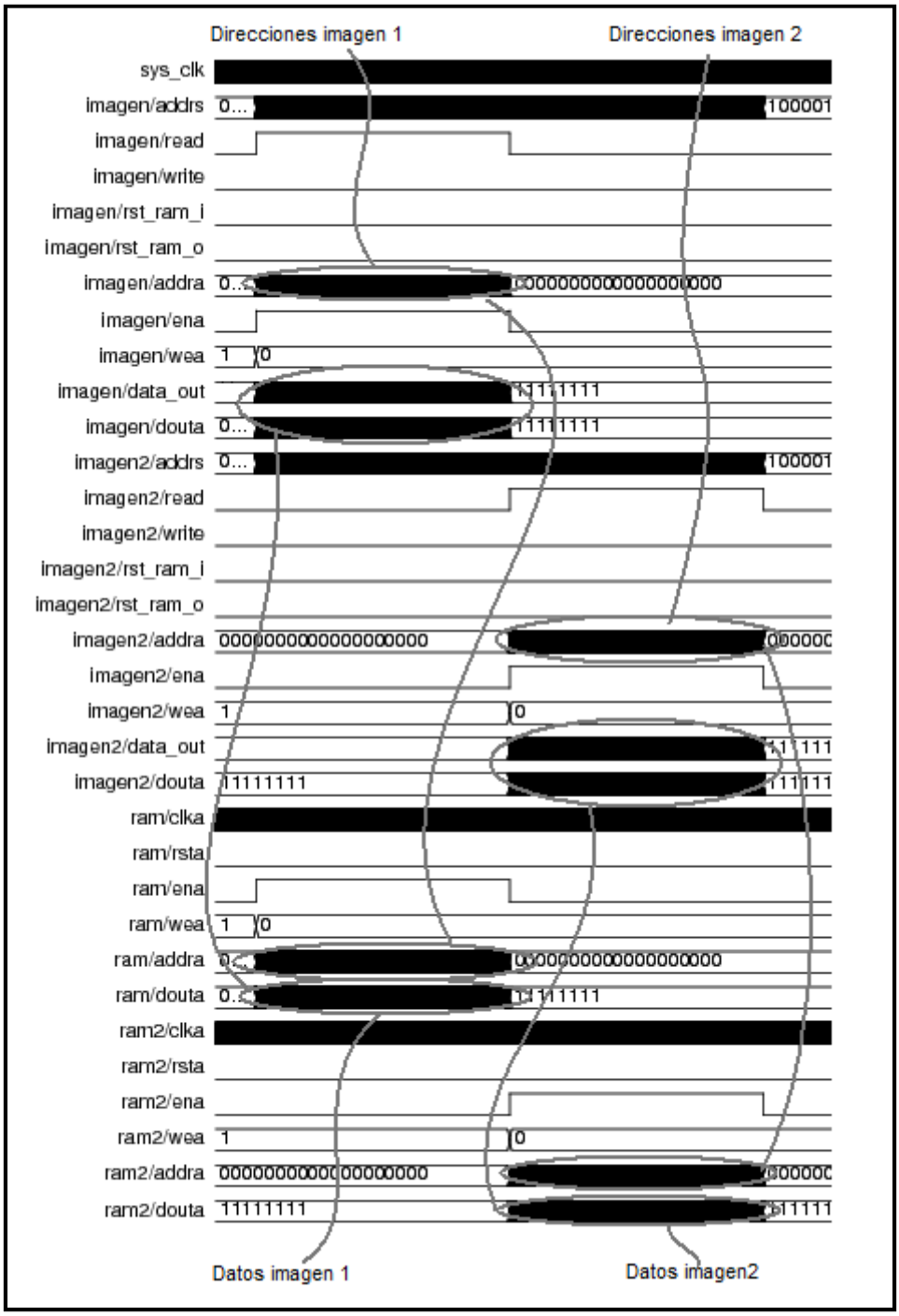


Figura 5.16: Simulación lectura de la imagen

Con esto se comprueba que todo el funcionamiento de la cámara estéreo es el esperado.

5.3 Pruebas del software

Estas pruebas consisten en observar que el sistema operativo Petalinux se ejecuta de forma correcta. Además se comprueba que existe comunicación entre el sistema y la cámara estéreo añadida a él.

5.3.1 Funcionamiento de Petalinux

Como se ha comentado en apartados anteriores, se procede a introducir los archivos `system.ace` e `imagen.bin` creados durante la realización de este proyecto en la tarjeta Compact Flash. La tarjeta se introduce en la placa ML402, y ayudados por un cable RS232 (`Null Modem`) se conecta la placa a un PC. Con la ayuda del `Hyperterminal`, configurado de forma correcta, se enciende la placa obteniéndose el siguiente resultado:

```
-- Entering m1() --
Starting MemoryTest for DDR_SDRAM_64Mx32:
  Running 32-bit test... PASSED!
  Running 16-bit test...PASSED!
  Running 8-bit test...PASSED!
Test Camera Stereo
 Data Status Register Sensor 1: 80000080
 Data Status Register Sensor 2: 80000080
 Data Status Register Sensor 3: 80000080
 Camera stereo found!
 Communication is OK!
 Sensor 1 connected!!
 Sensor 2 connected!!
```

```
 Sensor 3 connected!!
 Sensor 1 Stop!!
 Sensor 2 Stop!!
 Sensor 3 Stop!!
 End Test Camera Stereo
load_boot_image()
Lock granted
ResetCF Ok
IdentifyCF Ok


LBA: 00000020
SecPerClust: 4
NumReserved: 1
NumRootEnt: 512
SecPerFAT: 61
RootDirSec: 32
Aimag
IMAGE
FileSize: 2174980  FirstCluster: 919
IMAGE.BIN found!
.....
Jumping to kernel startup0... (44000000)
Linux version 2.4.32-uc0 (deako@Fedora10) (gcc version 3.4.1 ( PetaLinux 0.20 Bu
ild -rc1 050607 )) #97 Tue Aug 31 12:20:41 CEST 2010
On node 0 totalpages: 16384
```

```
zone(0): 16384 pages.
zone(1): 0 pages.
zone(2): 0 pages.
CPU: MICROBLAZE
Kernel command line:


Console: xmbserial on UARTLite
Calibrating delay loop... 49.86 BogoMIPS
Memory: 64MB = 64MB total
Memory: 62544KB available (953K code, 1325K data, 48K init)
Dentry cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Microblaze UARTlite serial driver version 1.00
ttyS0 at 0x84000000 (irq = 4) is a Microblaze UARTlite
Starting kswapd
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x0
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
Initializing SystemAce driver
```

```
Partition check:
 xsysacea: p1
System ACE at 0x83600000 mapped to 0x83600000, irq=3, 31360KB
SCSI subsystem driver Revision: 1.00
uclinux[mtd]: RAM probe address=0x4412dcbc size=0x10c000
uclinux[mtd]: root filesystem index=0
usb.c: registered new driver hub
usb.c: registered new driver hiddev
usb.c: registered new driver hid
hid-core.c: v1.8.1 Andreas Gal, Vojtech Pavlik <vojtech@suse.cz>
hid-core.c: USB HID support drivers
Initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
USB Mass Storage support registered.
cy7c67200_300_hcd.c:1750: Configuring for Xilinx ML40x
usb.c: new USB bus registered, assigned bus num
usb.c: kmalloc IF 443f74dc, numif 1
usb.c: new device strings: Mfr=0, Product=2, SerialNumber=1
usb.c: USB device number 1 default language ID 0x0
Product: USB CY7C67200/300 Root Hub
SerialNumber: 0
hub.c: USB hub found
hub.c: 1 port detected
hub.c: standalone hub
hub.c: ganged power switching
```

```
hub.c: global over-current protection
hub.c: Port indicators are not supported
hub.c: power on to power good time: 100ms
hub.c: hub controller current requirement: 0mA
hub.c: port removable status: R
hub.c: local power sourc
hub.c: no over-current condition exists
hub.c: enabling power on all ports
usb.c: hub driver claimed interface 443f74dc
usb.c: kusbd: /sbin/hotplug add 1
usb.c: kusbd policy returned 0xfffffffe
usb.c: new USB bus registered, assigned bus number 2
usb.c: kmalloc IF 443f75dc, numif 1
usb.c: new device strings: Mfr=0, Product=2, SerialNumber=1
usb.c: USB device number 1 default language ID 0x0
Product: USB CY7C67200/300 Root Hub
SerialNumber: 1
hub.c: USB hub found
hub.c: 1 port detected
hub.c: standalone hub
hub.c: ganged power switching
hub.c: global over-current protection
hub.c: Port indicators are not supported
hub.c: power on to power good time: 100ms
hub.c: hub controller current requirement: 0mA
```

```
hub.c: port removable status: R
hub.c: local power source is good
hub.c: no over-current condition exists
hub.c: enabling power on all ports
usb.c: hub driver claimed interface 443f75dc
usb.c: kusbd: /sbin/hotplug add 1
usb.c: kusbd policy returned 0xfffffffe
VFS: Mounted root (cramfs filesystem) readonly.
Freeing init memory: 48K
Mounting proc:
Mounting var:
Populating /var:
Mounting Compact Flash:
hub.c: port 1, portstatus 100, change 1, 12 Mb/s
hub.c: port 1 connection change
hub.c: port 1, portstatus 100, change 1, 12 Mb/s
hub.c: port 1, portstatus 100, change 1, 12 Mb/s
hub.c: port 1 connection change
hub.c: port 1, portstatus 100, change 1, 12 Mb/s
Running local start scripts.
Setting hostname:
(none) login:
```

Figura 5.17: Login de Petalinux

Como se ha observado de la figura anterior el resultado es igual al del proyecto *"Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas"*, utilizado como referencia.

Se introduce el `login` en este caso `root`, y también la `password` la cual también es `root`. Una vez introducido esto, se inicia el sistema operativo donde se puede navegar por él mediante las líneas de comandos.

5.3.2 Comprobación de la comunicación entre el sistema y la cámara estéreo

Ahora se realiza la comprobación de la comunicación entre el sistema y la cámara estéreo. Para ello se introduce en la función *main()* del código fuente *cf_loader.c* nombrado anteriormente las siguientes líneas de código.

```
// Probando camara

xil_printf ("Test Camera Stereo \r\n");

XIo_Out32(0x82000004,0x80000000);

XIo_Out32(0x82000104,0x80000000);

XIo_Out32(0x82000204,0x80000000);

c1reg0 = XIo_In32(0x82000000);

xil_printf(" Data Status Register Sensor 1: %08x \r\n",c1reg0);

c1reg0_31 = (c1reg0 & 0x80000000);// mascara para obtener el bit 31 del registro

c1reg0_7 = (c1reg0 & 0x00000080);

c1reg0_6 = (c1reg0 & 0x00000040);

c2reg0 = XIo_In32(0x82000100);

xil_printf(" Data Status Register Sensor 2: %08x \r\n",c2reg0);

c2reg0_31 = (c2reg0 & 0x80000000);// mascara para obtener el bit 31 del registro

c2reg0_7 = (c2reg0 & 0x00000080);

c2reg0_6 = (c2reg0 & 0x00000040);

c3reg0 = XIo_In32(0x82000200);

xil_printf(" Data Status Register Sensor 3: %08x \r\n",c3reg0);

c3reg0_31 = (c3reg0 & 0x80000000);// mascara para obtener el bit 31 del registro

c3reg0_7 = (c3reg0 & 0x00000080);

c3reg0_6 = (c3reg0 & 0x00000040);
```

```
 if (c1reg0_31 == 0x80000000 || c2reg0_31 == 0x80000000 ||        c3reg0_31 == 0x80000000){
   xil_printf(" Camera stereo found! \r\n");
   xil_printf(" Communication is OK! \r\n");}//end if
 else{
   xil_printf(" Camera stereo NOT found! \r\n");
   }//end else
 if (c1reg0_31 == 0x80000000){
   xil_printf(" Sensor 1 connected!! \r\n");}
 else{
   xil_printf(" Sensor 1 NOT connected!! \r\n");}
 if (c2reg0_31 == 0x80000000){
   xil_printf(" Sensor 2 connected!! \r\n");}
 else{
   xil_printf(" Sensor 2 NOT connected!! \r\n");}
 if (c3reg0_31 == 0x80000000){
   xil_printf(" Sensor 3 connected!! \r\n");}
 else{
   xil_printf(" Sensor 3 NOT connected!! \r\n");}
 if (c1reg0_31 == 0x80000000){
   if (c1reg0_7  == 0x00000080 && c1reg0_6 == 0x00000000){
     xil_printf(" Sensor 1 Stop!! \r\n");}
   else{
     xil_printf(" ERROR Sensor 1!! \r\n");}
          }
         if (c2reg0_31 == 0x80000000){
```

```
        if (c2reg0_7  == 0x00000080 && c2reg0_6 == 0x00000000){
     xil_printf(" Sensor 2 Stop!! \r\n");}
   else{
    xil_printf(" ERROR Sensor 2!! \r\n");}
       }
if (c3reg0_31 == 0x80000000){
        if (c3reg0_7  == 0x00000080 && c3reg0_6 ==
      0x00000000){
     xil_printf(" Sensor 3 Stop!! \r\n");}
   else{
     xil_printf(" ERROR Sensor 3!! \r\n");}
       }
        xil_printf(" End Test Camera Stereo \r\n");
```

Figura 5.18: Código para el Test Camera Stereo

Al introducir estas líneas de código, se ha observado al arrancar el sistema si se produce comunicación o no, porque desde el registro control de cada sensor que compone la cámara estéreo, se escribe la orden de reconocimiento, que consiste en enviar un bit y si el sensor la recibe correctamente, envía otro a través de su registro estado. Todo este proceso se realiza gracias a las funciones de escritura y lectura de las direcciones de Microblaze, las cuales son las siguientes:

```
XIo_Out32(0x82000004,0x80000000);//escritura

XIo_In32(0x82000000);//lectura
```

Figura 5.19: Funciones de escritura y lectura de memoria del Microblaze

Al encender la placa se ha observado que la comunicación entre Microblaze y la cámara estéreo se realiza de manera correcta. Las líneas que demuestran esto son las siguientes:

```
Test Camera Stereo
 Data Status Register Sensor 1: 80000080
 Data Status Register Sensor 2: 80000080
 Data Status Register Sensor 3: 80000080
 Camera stereo found!
 Communication is OK!
 Sensor 1 connected!!
 Sensor 2 connected!!
 Sensor 3 connected!!
 Sensor 1 Stop!!
 Sensor 2 Stop!!
 Sensor 3 Stop!!
 End Test Camera Stereo
```

Figura 5.20: Resultado Test Camera Stereo

Además se ha observado que los registros de estado también indican que los sensores de la cámara estéreo están en estado `“Stop”.`

Capítulo 6

Conclusiones y trabajos futuros

6.1 Conclusiones

En este proyecto se ha realizado el diseño de una cámara estéreo, que almacena dos o tres imágenes, para poder formar una imagen en tres dimensiones. Esta cámara estéreo, forma parte de un proyecto global cuyo objetivo es construir una cámara estéreo capaz de realizar un reconocimiento biométrico facial en 3D. Por ello se ha intentado que el control de la cámara estéreo utilice lo menos posible los recursos del micro-controlador para que puedan ser utilizados en futuros proyectos en el diseño del software que permite el reconocimiento biométrico. Esto se ha conseguido gracias al diseño de una cámara a la que se le han introducido unos registros internos, para que el micro-controlador realice únicamente las tareas de lectura y escritura sobre estos, obteniendo, sin embargo, el control completo de las funcionalidades de la cámara estéreo.

Para que el micro-controlador pueda escribir o leer los registros internos de la cámara estéreo, debe establecerse una comunicación, esto se ha conseguido con éxito como puede verse en las simulaciones del capítulo de pruebas, mediante el diseño de un módulo que actúa de interfaz entre el micro-controlador (Microblaze) y la cámara estéreo, la comunicación se establece mediante el protocolo PLB. También se puede observar la correcta comunicación al ejecutarse el Sistema Operativo, también expuesto en el capítulo de pruebas.

Una vez conseguido el objetivo más importante, que es la comunicación de Microblaze con la cámara estéreo para controlar los sensores de imagen creados por el proyecto *"Diseño y control de una cámara de video digital"* y tener acceso a las imágenes almacenadas, se realiza una serie de operaciones para la comodidad del manejo de la cámara estéreo.

Una de estas operaciones es la de adaptar los sensores de imágenes para que puedan ser configurados desde el exterior sin tener que modificar el código fuente del control del sensor. Esto se ha conseguido con éxito, gracias a la creación de registros que configuran el zoom, la resolución, etc. El funcionamiento de este proceso se puede observar en el capítulo de pruebas.

También se ha realizado el aumento del tamaño de la resolución de las imágenes captadas y el aumento de la calidad de la imagen, ya que los sensores captaban imágenes en blanco y negro, y ahora captan imágenes con 256 colores. Todo esto se ha conseguido gracias a la introducción de memoria externa, disponible en la placa Virtex4, hay que mencionar que se ha utilizado esta placa porque es la que está disponible en los laboratorios, pero existen placas en el mercado como la Virtex6 que tienen disponible mucha capacidad de memoria, pudiendo así obtener una resolución más alta.

Mencionado lo anterior se puede decir que se han cumplido todos los objetivos marcados al inicio de este proyecto.

6.2 Trabajos futuros

En este punto se añade algunos trabajos que pueden completar el diseño de la cámara estéreo en el futuro.

Nuestra cámara estéreo almacena imágenes en una memoria externa disponible en la placa lo que causa que el tamaño de la imagen dependa del tipo de placa utilizada. Por ello se podría realizar un diseño de un periférico que permitiese la conexión con una tarjeta MicroSD para almacenar las imágenes y así no depender del tipo de placa disponible en el laboratorio.

También en trabajos futuros se puede realizar un software de control de la cámara estéreo más avanzado e incluso realizar una interfaz grafica que haga más fácil el manejo de la cámara, ya que en esta versión todo se maneja mediante la línea de comandos.

Por último, se podrá añadir a la cámara el software capaz de analizar las imágenes captadas por la cámara estéreo, con el objetivo de obtener la información necesaria para el reconocimiento biométrico.

Referencias

[1] "Diseño de un sistema Linux empotrado para el procesamiento de señales biométricas" de Rodríguez Canales, Raúl. Julio 2008

[2] "Diseño y control de una cámara de video digital" de Ruiz Salcedo, Carlos. Junio 2008

[3] Estereoscopía. "Wikipedia enciclopedia libre". Disponible en la *http://es.wikipedia.org/wiki/Estereoscop%C3%ADa.* Febrero 2010

[4] Técnica de la estereoscopía. "Visión computacional. Luis Enrique Sucar". Disponible en la

http://www.google.es/url?sa=t&source=web&ct=res&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fccc.inaoep.mx%2F~esucar%2FVision%2Fvis07v05-3d.ppt&rct=j&q=Se+basa+en+buscar+una+peque%C3%B1a+regi%C3%B3n+%28template%29+de+una+imagen+en+la+otra%3A&ei=n_PjS4CTPJ_UmgPJprSCBg&usg=AFQjCNEfzBWxzmPwm7o8bNv42mYtDx2Icg. Mayo 2004

[5] Hiperestereoscopía. "La técnica de la estereoscopía". Disponible en la http://usuarios.arsystel.com/luismarques/documentacion/txt/04220_hiperestereoscopia.htm . Abril 2006

[6] Historia estereoscopía. "Introducción a la estereoscopía". Disponible en la *http://dmi.uib.es/~abasolo/cursorealidad/paco/Estereoscopia.html#introduccion*. Mayo 2004

[7] Processor Local Bus "128-Bit Processor Local Bus Architecture Specifications Version 4.7. IBM". Disponible en la

https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/3BBB27E5BCC165BA87256A2B0064FFB4/$file/PlbBus_as_01_pub.pdf. 2007

[8] Bus comunicación. "Tutorial Xilinx Microblaze. Aguayo E, González I. y Boemo E. Escuela Politécnica Superior, Universidad Autónoma de Madrid, España". Disponible en la *http://arantxa.ii.uam.es/~ivan/microblaze-jcra04.pdf*. Octubre 2006

[9] Board ML40x "*ML40x Getting Started Tutorial for ML401/ML402/ML403/ML405 Evaluation Platforms*". Disponible en la

http://www.xilinx.com/support/documentation/boards_and_kits/ug083.pdf . Septiembre 2007.

[10] User Guide ML40x "*ML40x EDK Processor Reference Design User Guide*". Disponible en la

http://www.xilinx.com/support/documentation/boards_and_kits/ug082.pdf. Septiembre 2007.

[11] User Guide Platform "*ML401/ML402/ML403 Evaluation Platform User Guide*". Disponible en la

http://www.xilinx.com/support/documentation/boards_and_kits/ug080.pdf. Septiembre 2007.

[12] Schematics "*ML401/ML402/ML403 Schematics*". Disponible en la *http://www.xilinx.com/support/documentation/boards_and_kits/ml401_2_3_schematics.pdf*. Septiembre 2007.

[13] Summary of Virtex-4 Family Features. "*Virtex-4 Family Overview*". Disponible en la *http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf*. Septiembre 2007.

[14] Presupuesto "Virtex-4 ML402 SX XtremeDSP Evaluation Platform". Disponible en la *http://www.xilinx.com/products/boards_kits/virtex4.htm*. Febrero 2010

Presupuesto y Diagrama de Gantt

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

PRESUPUESTO DEL PROYECTO

1 - Autor: Amadeo Ángel Velasco García

2 - Departamento: Departamento de tecnología electrónica

3 - Descripción del Proyecto:

Titulo: Diseño de una cámara estéreo con microprocesador embebido para aplicaciones biométricas.

Duración: 12 Meses

Tasa de costes indirectos: 15%

4- Presupuesto total del Proyecto: 32140.33€

5- Desglose presupuesto:

PERSONAL

Apellidos y Nombre	NIF	Categoría	Dedicación[1] (hombres /mes)	Coste hombre/mes	Coste	Firma de Conformi-dad
Velasco García, Amadeo Ángel	45678901-X	Ingeniero	9	2.694.39€	24249.51€	
Pérez Yagüe, Paloma	45678902-X	Secretaria	3	1077.77€	3233.27€	
				Total	**27482.78€**	

[1] 1 Hombre mes = 131.25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas).

EQUIPOS

Descripción	Coste (con IVA = 18%)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable[2]
Virtex-4 ML402 SX XtremeDSP Evaluation Platform [14]	1099.20€ (1410.10$)	100	9	60	139.73€
PC Compaq Presario CQ60	706.82€	100	9	60	89.85€
Licencia Software "Platform Studio and Embedded Development Kit"	1854.57€ (2358.82$)	100	9	60	235.75€
				Total	**465.33€**

[2] Fórmula de cálculo de la Amortización:

$$\frac{\mathbf{A}}{\boldsymbol{B}} \; \boldsymbol{x \; C \; x \; D}$$

A = n º de meses desde la fecha de facturación en que el equipo es utilizado.

B = periodo de depreciación (60meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
-	-	0,00€

OTROS COSTES DIRECTOS DEL PROYECTO

Descripción	Empresa	Coste imputable
-	-	0,00€

6 – Resumen de costes:

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	27482.78€
Amortización	465.33€
Subcontratación de tareas	0.00€
Costes de funcionamiento	0.00€
Costes Indirectos	4192.22€
Total	**32140.33€**

DIAGRAMA DE GANTT

El tiempo empleado para la realización del proyecto ha sido de 12 meses con una dedicación de 8 horas diarias.

TAREA	INICIO	FIN	DURACION	2009					2010									
				A	S	O	N	D	E	F	M	A	M	J	J	A	S	O
Estudio previo	07/09 /2009	11/11 /2009	48															
Diseño Sistema	13/11/ 2009	26/03 /2010	92															
Desarrollo y Pruebas	01/04/ 2010	01/06 /2010	43															
Redacción Memoria	03/06/ 2010	03/09/ 2010	66															

Glosario

2D (2 Dimensiones)

3D (3 Dimensiones)

CCD (charge-coupled device)

DCR (Device Control Register)

FSL (Fast Simple Link)

HMD (Head Mounted Display)

LCD (Liquid Crystal Display glasses)

LCS (Liquid Crystal Shutter glasses)

LMB (Local Memory Bus)

LUT (Look Up Table)

MHz (Megaherzios)

MPD (Microprocesador Peripherals Description)

OPB (On-chip Peripheral Bus)

PAO (Peripheral Analyze Order)

PLB (Processor Local Bus)

RAM (Random-Access Memory)

RAMB (Random-Access Memory Block)

RGB (Red Green Blue)

ROM (read-only memory)

SIFT (Scale-invariant feature transform)

USB *(*Universal Serial Bus*)*

VGA *(*Video Graphics Array*)*

VHDL (Very High Description Language)