Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Ingeniería de Computadores

CAPTURA DE IMÁGENES Y VISUALIZACIÓN DE VIDEO SOBRE PLACA ALTERA DE2

David Infante Sainz

Junio 2016

informatika fakultatea
facultad de informática

Agradecimientos

Me gustaría aprovechar esta oportunidad para agradecer a mi familia todo el apoyo recibido durante la carrera.
También, recordar a todos mis amigos y a los compañeros de clase de la carrera con los que tanto tiempo he pasado.
Por último, dar las gracias a mi director y coordinador de proyecto por su ayuda y disponibilidad.

Resumen

El proyecto presentado pretende desarrollar un sistema que recoja imágenes de una **cámara** y las muestre en una **pantalla LCD** con la intención de dejar una plataforma lista para que los usuarios puedan disponer de él e incluso se puedan hacer mejoras partiendo de dicho sistema.
Con el objetivo de desarrollar este sistema, se partirá de dos diseños ejemplo ya desarrollados, aplicables para la gestión del funcionamiento de cada dispositivo. El diseño ejemplo **DE2_LTM_Ephoto**, nos permite visualizar en una **pantalla LCD** las imágenes almacenadas en una memoria Flash. En el sistema **DE2_D5M**, las imágenes capturadas por la **cámara** se muestran en una pantalla VGA. Estos ejemplos se explicarán de manera más detallada a lo largo de la memoria.
El diseño se ha realizado en **vhdl**, un lenguaje visto en la asignatura de Diseño y construcción de Sistemas Digitales para evitar problemas a la hora del desarrollo. Por lo tanto, se tendrá que traducir los dos diseños ejemplos anteriormente comentados ya que están en lenguaje **verilog**. Para su desarrollo se ha utilizado la herramienta **Quartus II**.
El sistema a desarrollar se implementa sobre una **placa Altera DE2**, donde se podrán controlar los dos dispositivos, tanto la **cámara** como la **pantalla LCD** mediante dos puertos a los que se pueden conectar.
A la hora de desarrollar el sistema final, se utilizarán los dos sistemas ejemplo ya que implementan todas las funcionalidades, pero cada una por separado, por lo que se tendrá que hacer una serie de modificaciones para adaptar los dos dispositivos. Dichas modificaciones se detallarán a lo largo de la memoria.

Índice

Lista de Figuras

FIGURAS

1

Introducción y definiciones

1.1. Justificación del Proyecto

La principal decisión por la que se ha realizado este proyecto es para poder aplicar lo aprendido en la asignatura de Diseño y construcción de Sistemas Digitales ya que es una de las asignaturas donde más he disfrutado a la hora de desarrollar los conocimientos aprendidos.
Una vez decidido qué tipo de proyecto realizar, después de hablar con el director y profesor de la asignatura mencionada, se llegó a la conclusión de desarrollar una plataforma base mediante el leguaje **vhdl** en el que fuese compatible la **cámara** con la **pantalla LCD.**

1.2. Objetivos del proyecto

En este apartado se exponen los diversos objetivos que se quieren conseguir al término de este proyecto:
El principal objetivo de este proyecto consiste en la construcción de un sistema en **vhdl** en la que se pueda visualizar las imágenes que se capturan de la cámara (**TRDB_D5M**), en la pantalla (**TRDB_LTM**) a través de la **placa altera DE2**, para que nos permita la configuración y control de las funcionalidades de ambos dispositivos de tal forma que pueda servir como punto de partida para sistemas más sofisticados de captura y procesado de imágenes.
Se comenzará por estudiar y comprender los dos sistemas ejemplos **DE2_LTM_Ephoto** (para la pantalla) y **DE2_D5M** (para la cámara). Estos sistemas están en **verilog** por lo que podrán surgir algunos problemas de comprensión.

Se tendrá que conseguir juntar los dos ejemplos modificándolos de manera que se consiga desarrollar un sistema que funcione con los dos dispositivos; la cámara y la pantalla LCD.

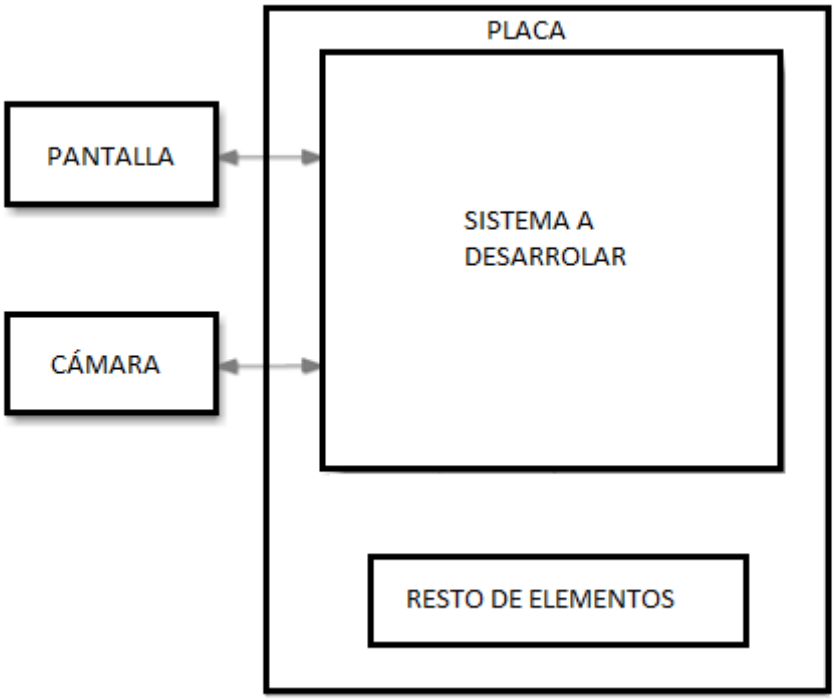


Figura 1: Sistema para la Pantalla LCD y la Cámara

Partiendo de esta base, se tendrá que analizar el funcionamiento de los dispositivos que van a estar integrados en nuestro proyecto. Se comenzará ejecutando cada uno de los sistemas ejemplo para ver cómo interactúan cada uno de los sistemas con los dispositivos mediante la **placa altera DE2**. Como he mencionado anteriormente, para comprender a la perfección cada uno de los ejemplos, tendremos que aprender una base del lenguaje **verilog** para entender la sintaxis y poder convertir los sistemas ejemplo a **vhdl**. También se tendrá que aprender el funcionamiento del programa **Quartus II** para desarrollar el sistema final. Este paso será más sencillo ya que se trabajó con esta herramienta durante la asignatura de Diseño y Construcción de Sistemas Digitales.

Además, se tendrá que seleccionar que partes de los diseños ejemplos van a servir para el diseño final, y que partes se tendrán que modificar para lograr visualizar las imágenes capturadas por la **cámara** en la **pantalla LCD.**

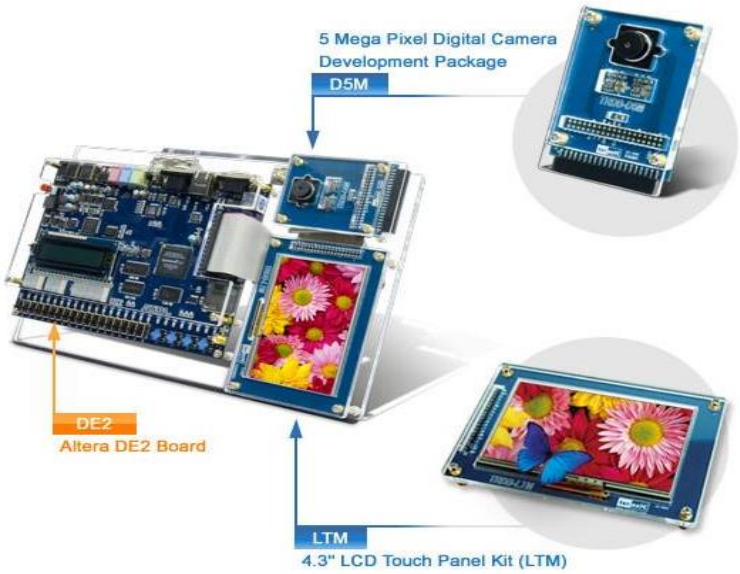


Figura 2: Conexión de dispositivos (cámara y pantalla LCD) con la placa DE2.

1.3. Fases del proyecto

1.3.1 Fase de planificación

Para la creación de este proyecto se ha visto conveniente el desarrollo de las siguientes fases:
Fase de estudio de tecnologías, fase de análisis, fase de diseño, fase de implementación, fase de pruebas y fase de documentación.
Una vez capturados y especificados los requisitos se puede comenzar a desarrollar el proyecto.
Partiendo de la base de que para desarrollar el sistema definitivo hay que partir desde los dos diseños (**DE2_D5M** y **DE2_LTM_Ephoto**) uno por cada dispositivo (cámara **TRDB_D5M** y pantalla **TRDB_LTM**), hay que tener en cuenta primeramente que el diseño (**DE2_D5M**) que funciona con **la cámara TRDB_D5M**, está pensado para que trabaje con una pantalla VGA, y que el

diseño (**DE2_LTM_Ephoto**) que funciona con el **LCD TRDB_LTM**, está pensado para recoger imágenes desde una memoria Flash.
Se hará una breve descripción de estos dos elementos que será importante a la hora de planificar como podemos juntar ambos elementos eliminando las partes no necesarias para conseguir lograr el objetivo final. Se estudiarán sus configuraciones internas y el control de flujo de datos de cada uno de los dispositivos para después crear un sistema ejemplo que se utilizará como base para crear el nuevo sistema en **vhdl**.
Antes de realizar nuestro nuevo sistema hay que analizar los dos sistemas ejemplo. De estos sistemas habrá partes que interactuarán con la captura de imágenes desde la memoria Flash o con la muestra en la pantalla VGA, que serán eliminados en el nuevo proyecto a desarrollar, ya que no tienen ninguna utilidad en el proyecto que se quiere realizar.
Parte importante del desarrollo del sistema estará en concretar que funciones de cada uno de los dispositivos se necesitarán para poder modificarlos e incluso añadir cosas para lograr que la cámara sea compatible con la pantalla.
Se partirá de un diseño base una vez estudiadas sus características principales que podrán ir variando según los resultados para lograr el sistema definitivo.

1.3.2 Fase estudio de tecnologías

Debido a la experiencia desarrollada durante la carrera se ha decidido desarrollar un sistema mediante **Quartus II**, por lo que se realizará un estudio del lenguaje **verilog** para poder entender los dos ejemplos con que partiremos y de esta manera poder traducirlos al lenguaje **vhdl** seleccionando que módulos serán necesarios para el desarrollo del sistema.

1.3.3 Fase de análisis

A partir de la planificación y estudio de las tecnologías que se van a utilizar, se realizará un análisis detallado de la aplicación, obteniendo los casos de uso definitivos con el objetivo de completar la aplicación para la fase de diseño.
Mediante esta fase se identificarán de manera definitiva las funciones que se necesitarán de los dos casos ejemplo para desarrollar el sistema final.

1.3.4 Fase de diseño e implementación

En la fase de diseño se diseñará la arquitectura de la aplicación a partir de los resultados de la fase de análisis.

Esta será la fase de mayor envergadura del proyecto ya que se tendrá que poner a prueba lo aprendido tanto en la asignatura como por mi cuenta para poder realizar una traducción de **verilog** a **vhdl** correctamente. Una vez funcionen los dos casos ejemplos en **vhdl**, se tendrá que decidir que módulos y funciones serán los necesarios para el tipo de plataforma que se quiere desarrollar.

Y como parte final y la más complicada, saber complementar y combinar las funciones de los dos casos ejemplo para conseguir que funcione el sistema en perfectas condiciones mostrando en la **pantalla LCD** las imágenes capturadas por la **cámara.**

Prácticamente siguiendo las pautas del diseño se logrará que la aplicación tome una forma casi completa a falta de realizar las pruebas necesarias para validar su correcto funcionamiento

1.3.5 Fase de pruebas

Una vez finalizada la implementación se deberá probar que cumple todos los requisitos planteados.

Para evaluar la aplicación se plantearán un conjunto de pruebas que el sistema deberá superar.

El documento de plan de pruebas especificará las distintas pruebas realizadas, describiendo detalladamente los resultados de cada una.

En el caso de encontrar algún error, repercutirá en modificaciones en etapas anteriores.

1.3.6 Fase de documentación

A pesar de ser la última fase del proyecto, es conveniente elaborar dicha documentación según se van realizando las demás fases.

La documentación se realizará inicialmente mediante una base de anotaciones realizadas durante la realización del proyecto que se modificarán en la última fase del proyecto.

2

Fase de Planificación

2.1. Estimación para la realización de las tareas

Con el fin de lograr los objetivos marcados en la elaboración del proyecto, hay una serie de tareas que se deben de cumplir en los plazos que se verán en el siguiente gráfico.
Se tendrá que tener claro que tareas hay que realizar y en qué orden se han de hacer. Algunas de las tareas se harán de forma individual ya que para comenzar con la siguiente fase se necesita tener terminada dicha tarea. Otras por el contrario podrán ser realizadas de forma simultánea.
Esto ocurrirá prácticamente una vez iniciada la fase de pruebas ya que un error causará la modificación de las anteriores tareas.

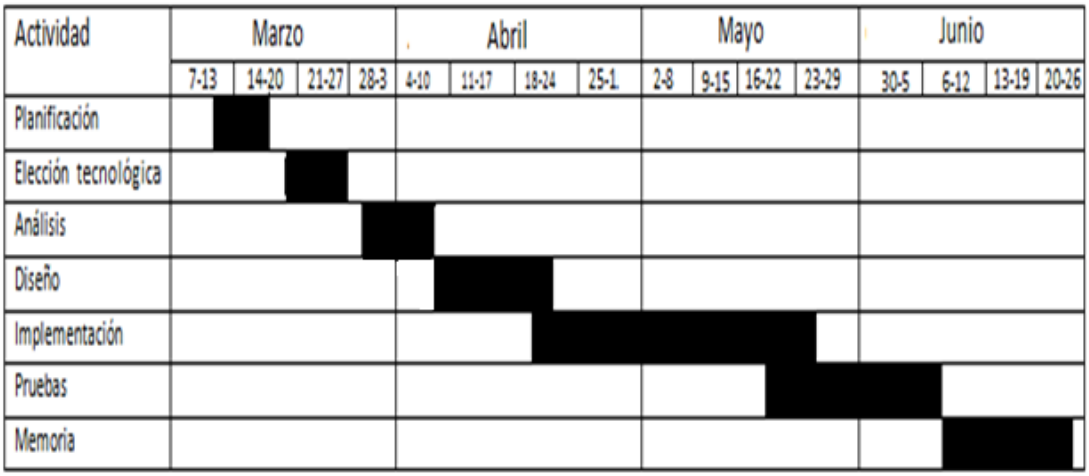


Figura 3: Diagrama de Gantt de desarrollo del proyecto.

2.2. Plan de Contingencia

Se ha decidido crear un plan de contingencia con el fin de superar los obstáculos que me pueda encontrar para lograr que el sistema se desarrolle tal y como se había decidido.
Uno de los riesgos más importantes es la pérdida de información tanto del desarrollo del software como de la documentación. Para evitarlo se ha decidido tomar una serie de costumbres para guardar el desarrollo del proyecto hasta el momento, en el disco duro y en un usb por seguridad.
A continuación, se comentan alguno de los riesgos:

-Objetivos no decididos

Probabilidad del evento: Alta
Impacto del riesgo: Alto
Descripción: Debido a la numerosa información disponible sobre los dos elementos que queremos utilizar en nuestro proyecto, algunos objetivos podrán cambiar según se va desarrollando el sistema.
Contramedida: Establecer unos objetivos concretos y alcanzables.

-Experiencia insuficiente

Probabilidad del evento: Media
Impacto del riesgo: Alto
Descripción: Los conocimientos enseñados en la Facultad pueden ser insuficientes para algunos temas a tratar en el proyecto.
Contramedida: Aprender las tecnologías a emplear y evitar grandes riesgos en la medida de lo posible.
Dado que tanto los Sistemas de Ejemplo aplicados a los dispositivos, como el Sistema a desarrollar, tienen como lenguaje de programación **verilog**. Primeramente, se aprenderá a realizar un diseño de un Sistema en dicho lenguaje.

-Pérdida de información

Probabilidad del evento: Alta
Impacto del riesgo: Alto
Descripción: Siempre que se trabaja con Sistemas de Información aparece el riesgo de perder información.
Contramedida: Evitar poner en riesgo el trabajo realizado hasta el momento realizando copias de seguridad cada poco tiempo.

3

Fase de Estudios de Tecnologías

Los dos dispositivos (cámara **TRDB_D5M** y pantalla **LCD TRDB_LTM**) que se desean conectar, están conectados a través de una **FPGA DE2 Cyclone II EP2C35F672C6N** por sendos puertos de expansión de 40 pines.
Desde los dos sistemas ejemplos se desarrollará un sistema integral, que consistirá en el control de la captura de imágenes de la **cámara D5M** y las muestre en el **LCD.**
Antes de comenzar con el desarrollo del sistema, el sistema inicializará los registros de configuración internos de los dispositivos.
En el desarrollo de este sistema, se utilizará la herramienta de software **Quartus II**. Herramienta que sirve para el diseño e implementación en las placas de **ALTERA** de proyectos hardware tanto en **vhdl**, **verilog** o herramientas **CAD**.

3.1. QUARTUS II

Es una herramienta de diseño de **ALTERA** para el análisis y síntesis de diseños realizados en **HDL**. **Quartus II** permite compilar dichos diseños, realizar análisis temporales mediante simulaciones o verificar las señales internas de la **placa DE2** en tiempo real e implementar los diseños en la placa para ejecutarlas internamente.
Quartus II permite al desarrollador o desarrolladora compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador.
Con el programa de diseño **Quartus II** los diseñadores pueden usar los dispositivos HardCopy Stratix de manera que puede prever y verificar su rendimiento, el cual resulta en promedio un 50 por ciento más rápido que su

FPGA equivalente. Además, en el flujo de diseño del HardCopy Stratix, **Quartus II** incluye una serie de utilidades que reducen el tiempo de diseño. Como contraste adicional el bajo precio del **Quartus II** en comparación con otras herramientas de diseño de ASIC.
La Edición Web es una versión gratuita de **Quartus II** que puede ser descargada o enviada gratuitamente por correo. Esta edición permite la compilación y la programación de un número limitado de dispositivos **Altera**. La familia de **FPGAs** de bajo coste **Cyclone**, está soportada por esta edición, por lo que los pequeños desarrolladores y desarrolladoras no tendrán problemas por el coste del desarrollo de software.
Se requiere un registro de licencia para utilizar la Edición Web de **Quartus II**, la cual es gratuita y puede ser renovada ilimitadamente o de pago.

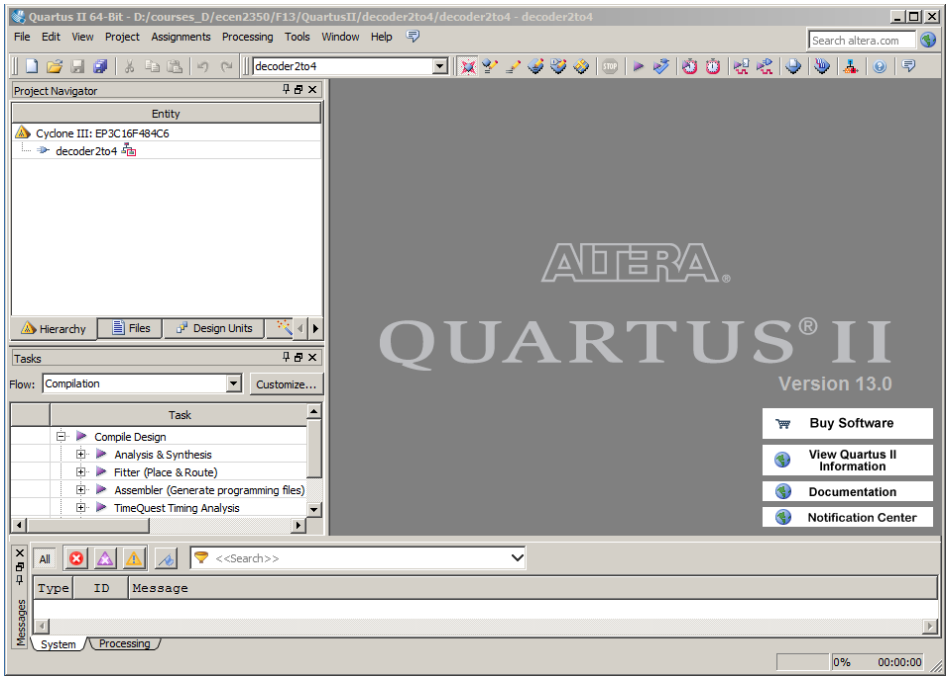


Figura 4: Plataforma Quartus II.

3.2. PLACA ALTERA DE2

Altera Corporation (NASDAQ: ALTR) es un fabricante líder de dispositivos lógicos programables. Es uno de los valores que integran los índices bursátiles NASDAQ-100 y S&P 500.
Altera desarrolla algunas características que están orientadas hacia capacidad de sistemas en chips programables (SOPC). Algunos de los

ejemplos más recientes incluyen memoria embebida, procesadores embebidos, y transceptores de alta velocidad.
Los procesadores **soft-core Nios II** y **Nios** de **Altera** y los dispositivos HardCopy II y HardCopy están extendiendo el alcance de **Altera** en el mercado, y coloca a esta empresa en el mundo de los procesadores embebidos y ASICs estructuradas respectivamente. Entre sus principales competidores están: Xilinx,Lattice
Altera ofrece también el software **Quartus II**, dirigido al diseño y simulación de circuitos lógicos.
Aunque su software soporta extensivamente **VHDL** y **Verilog** como principales lenguajes, **Altera** es el desarrollador de lenguaje de descripción de hardware conocido como AHDL.
Es una placa especialmente pensada para la gestión de dispositivos conectados a ella, en el control de la imagen y el sonido mediante diseños hardware. Para ello los diseños hardware realizados se pueden implementar en la **FPGA Cyclone II 2C35**. Desde estos diseños implementados controlaremos los componentes de la placa para gestionarlos según el propósito que queremos realizar.
De esta forma todos los componentes importantes de la placa se conectan a los pines de la **FPGA**, permitiendo al usuario controlar todos los aspectos de las operaciones de la placa.

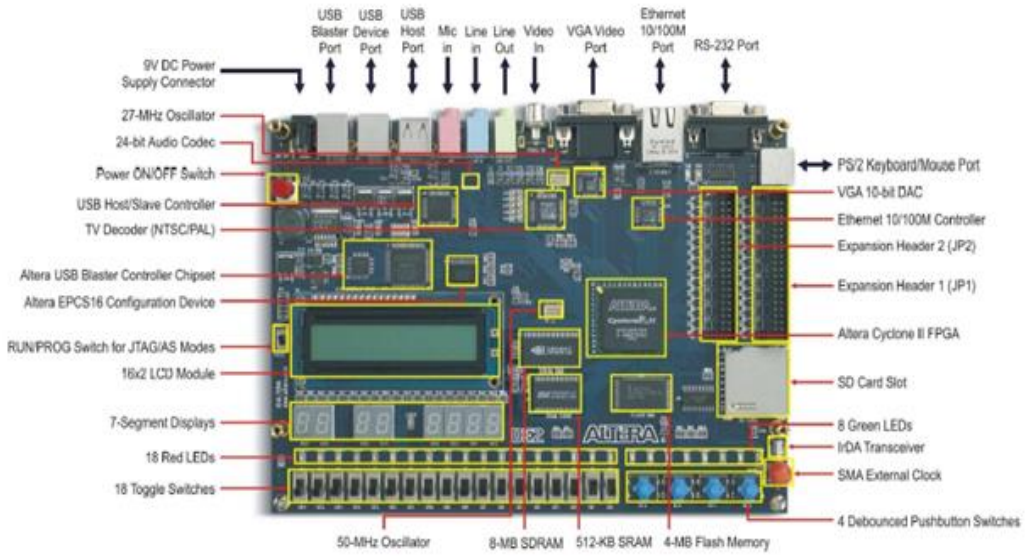


Figura 5: Placa de ALTERA DE2.

Para desarrollos más avanzados, se suele utilizar la SRAM, el chip de memoria Flash o la SDRAM cuando conlleve un almacenamiento importante de datos cómo por ejemplo los datos de los pixeles de la imagen. Para experimentos que requieran un procesador e interfaces de entrada/salida, se suele utilizar el procesador **NIOS II de ALTERA** y los interfaces estándar de RS232 y PS/2. Para experimentos que impliquen señales de sonido o video, hay conectores estándar para micrófonos, líneas de entrada, líneas de salida (CODECs de 24 bits de audio), entradas de video (Decoders de TV), y VGA (10-bits DAC); estas herramientas se pueden utilizar para crear aplicaciones de audio y de video de alta calidad. Finalmente, es posible conectar otra placa u otro dispositivo definido a través de los dos puertos de expansión.

3.3. CAMARA (TRDB_D5M)

Esta cámara es uno de los dos elementos que vamos a utilizar.
La **cámara D5M** realizará las capturas de las imágenes para mostrarlas en el **LCD**, a través de la **placa FPGA DE2**.

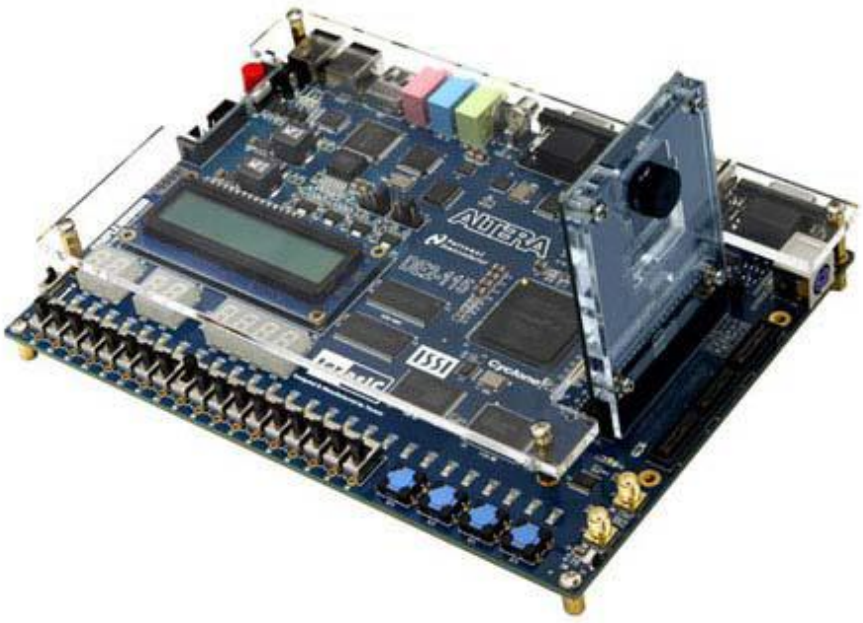


Figura 6: Cámara TRDB-D5M.

Este dispositivo está conectado a la placa **FPGA DE2** mediante un puerto de 40 pines. Mediante estas conexiones podremos controlar aspectos como el sincronismo y funcionamiento de la cámara, así como el estado de la cámara. Las conexiones con la **placa DE2** se pueden ver en la siguiente imagen.

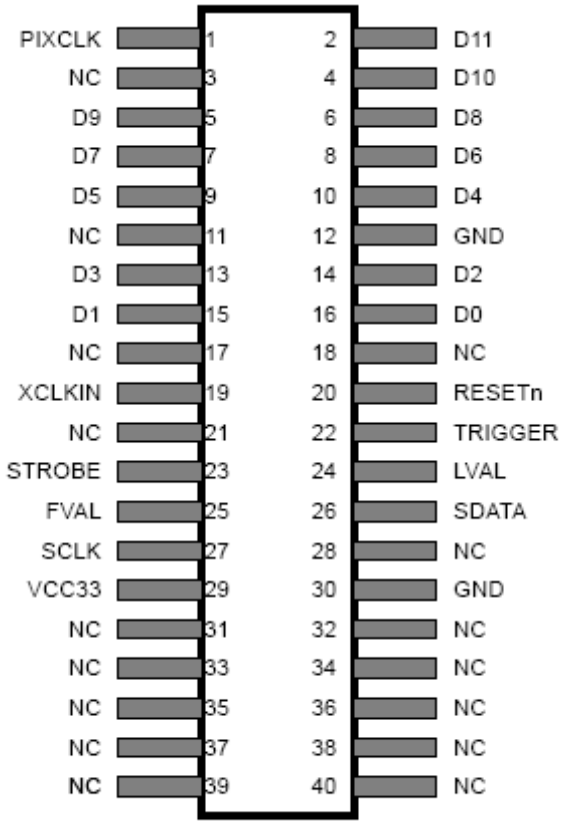


Figura 7: Conexiones de la cámara con el puerto de 40 pines.

Aspectos del funcionamiento interno de la cámara THDB-D5M:

1.- Configuración de los registros

Tiene unos registros internos configurables desde el exterior para el control de su funcionamiento y captura de las imágenes.

En estos registros internos se puede controlar, la ganancia de luminosidad, la velocidad de captura de la imagen, resolución de la imagen, la exposición, la sincronización, etc…

Cada uno de estos registros configura una función determinada de la cámara **THDB-D5M**, que puede variar dependiendo del valor o parámetro que se asigne.

Ejemplo:

Registro R0x004 tamaño de columna (Column size).
Registro R0x003 tamaño de fila (Row size).
Registro R0x022 Output de salida (Row_Bin y Row_Skip).
Registro R0x023 Output de salida (Column_Bin y Column_Skip).

Más adelante veremos funciones para la resolución de la imagen recogida por la cámara, modos de lectura, etc.…
La escritura de esta información en cada registro de la cámara o su lectura, se realiza vía serie.
A través del módulo principal del sistema que se ha diseñado reinicializa la configuración de los registros y gestiona las salidas de datos de los pixeles de la cámara a otros dispositivos.

2.- Sincronismo en el envío de información de los pixeles

La secuencia de la señal captada por la cámara se divide en imágenes o en frames, y cada uno de estos frames se divide en filas. Por defecto, el sensor produce una resolución por imagen de 1944 filas y 2592 columnas.
Esta resolución es configurable en los registros internos de la cámara como se va a ver más adelante.
El dato de cada pixel lo envía la cámara al exterior por cada ciclo de reloj **PIXCLK**. La frecuencia de este reloj también se configurará en los registros internos de la cámara. Por cada pixel se envía hacia el exterior 12 bits para el color rojo R [11:0], 12 bits para el color verde G [11:0] y 12 bits para el color azul B [11:0].

3.4. PANTALLA LCD (TRDB_LTM)

La **pantalla LTM** es el otro elemento que se utilizará, al cual llegarán las imágenes que captura la **cámara D5M**, a través de la **placa DE2.**
También tiene registros internos de control configurables para seleccionar su funcionamiento y la muestra de imágenes en pantalla al igual que la cámara. Se podrá modular el color, el brillo, así como convertir las coordenada X/Y analógicas de la pantalla táctil a un valor digital a través del convertidor AD7843 AD.

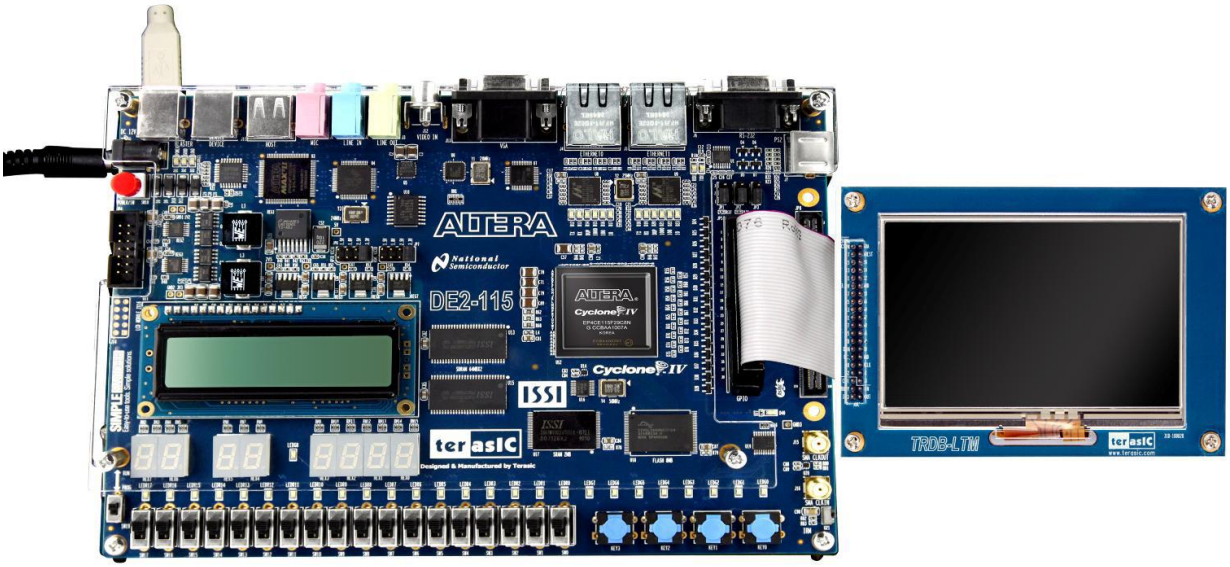


Figura 8: Pantalla TRDB-LTM

Esta pantalla está conectada a la placa **FPGA DE2** mediante un puerto de 40 pines.
Mediante estas conexiones se podrá controlar el sincronismo y el funcionamiento del dispositivo. Mediante las salidas se informará a nuestro sistema de su estado.
También se podrá modificar su resolución hasta un valor de 800x480.

Pin Numbers	Name	Direction	Description
1	ADC_PENIRQ_n	output	ADC pen Interrupt
2	ADC_DOUT	output	ADC serial interface data out
3	ADC_BUSY	output	ADC serial interface busy
4	ADC_DIN	input	ADC serial interface data in
5	ADC_DCLK	input	ADC/LCD serial interface clock
6	B3	Input	LCD blue data bus bit 3
7	B2	Input	LCD blue data bus bit 2
8	B1	Input	LCD blue data bus bit 1
9	B0	Input	LCD blue data bus bit 0
10	NCLK	Input	LCD clock signal
11	NC	N/A	N/A
12	GND	N/A	Ground
13	DEN	Input	LCD RGB data enable
14	HD	Input	LCD Horizontal sync input
15	VD	Input	LCD Vertical sync input
16	B4	Input	LCD blue data bus bit 4
17	B5	Input	LCD blue data bus bit 5
18	B6	Input	LCD blue data bus bit 6
19	B7	Input	LCD blue data bus bit 7
20	G0	Input	LCD green data bus bit 0
21	G1	Input	LCD green data bus bit 1
22	G2	Input	LCD green data bus bit 2
23	G3	Input	LCD green data bus bit 3
24	G4	Input	LCD green data bus bit 4
25	G5	Input	LCD green data bus bit 5
26	G6	Input	LCD green data bus bit 6
27	G7	Input	LCD green data bus bit 7
28	R0	Input	LCD red data bus bit 0
29	VCC33	N/A	Power 3.3V
30	GND	N/A	Ground
31	R1	Input	LCD red data bus bit 1
32	R2	Input	LCD red data bus bit 2
33	R3	Input	LCD red data bus bit 3
34	R4	Input	LCD red data bus bit 4
35	R5	Input	LCD red data bus bit 5
36	R6	Input	LCD red data bus bit 6
37	R7	Input	LCD red data bus bit 7
38	GREST	Input	Global reset, low active
39	SCEN	Input	LCD 3-wire serial interface enable/ADC chip enable
40	SDA	Input/Output	LCD 3-wire serial interface data

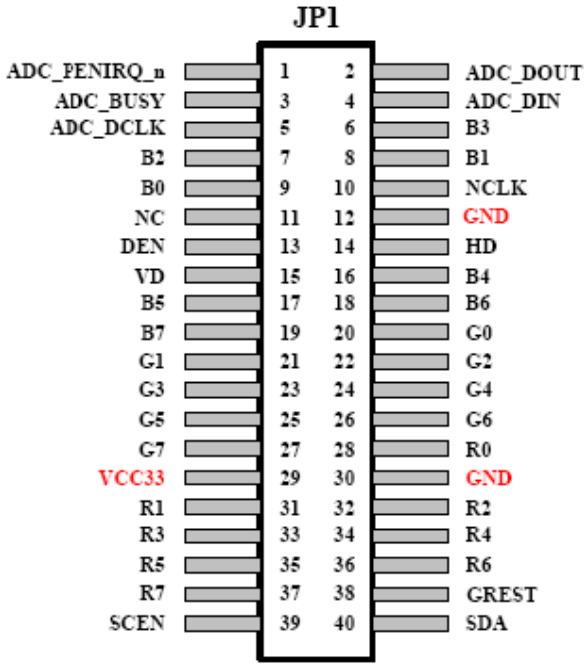


Figura 9: Conexiones de la pantalla con el puerto de 40 pines.

Antes de que la pantalla comience a funcionar se tendrá que reinicializar sus registros para que funcione adecuadamente según lo que queremos conseguir. Estos registros son los que configuran el funcionamiento interno del dispositivo.

Aspectos del funcionamiento interno de la cámara THDB-D5M:

1.- Configuración de los registros

Para configurar los registros internos del **TRDB_LTM**, los datos se envían vía serie desde el puerto de expansión de 40 pines de la **FPGA DE2** al LCD.

La información que se envía es de un total de 16 bits y se empiezan a enviar en el momento que la entrada del dispositivo **TRDB_LTM SCEN** se pone a cero.

El dato no se acepta si hay menos o más de 16 ciclos para una transacción.

2.- Proceso de envío de datos RGB a pantalla

El envío de datos de los pixeles dependerá de la resolución.

El ancho de pantalla lo controla la entrada **HD** del **TRDB_LTM**. Entre cada uno de los flancos de bajada del **HD**, hay una serie de pixeles de una fila que se muestran en pantalla que marcará el ancho de la imagen.

Cada dato **RGB** de cada pixel que se envía está compuesto por 8 bits por cada color (rojo (bits R0-7), verde (bits G0-7), azul (bits B0-7)). De esta forma cada dato está compuesto por 24 bits de información. Esta información irá variando cada ciclo de reloj **NCLK**.

Mientras la entrada **SCEN** esté activada a nivel alto se enviarán los pixeles que estarán habilitados para mostrarse en pantalla. Mientras esto ocurra la información de los pixeles se mostrarán en pantalla. El tiempo que hay entre que **HD** se pone a nivel bajo y la entrada **SCEN** se pone a nivel alto, es el borde de la pantalla que hay hasta llegar a la imagen.

4

Fase de Análisis

4.1. Aprendizaje lenguaje Verilog

Dado que los dos Sistemas de Ejemplo aplicados a los dispositivos tienen como lenguaje de programación **verilog**, y como dichos ejemplos serán la base de nuestro Sistema a desarrollar, como primer paso se comenzará por estudiar los conceptos básicos del lenguaje y se aprenderá a realizar un diseño de un Sistema en dicho lenguaje.

4.2. Análisis de la Documentación

En esta sección se realizará una lectura y análisis de la documentación asociada a cada uno de los dispositivos (**D5M** y **LTM**), para estudiar y entender su funcionamiento.
Los dos dispositivos mencionados cuentan con un Sistema ejemplo de aplicación, que se debe de estudiar, porque habrá partes que se podrán aprovechar y partes en las que con pequeñas modificaciones se logre desarrollar un Sistema para los dos dispositivos.
Hay que destacar los módulos que se encargan de la inicialización del Sistema y de los registros de cada uno de los dispositivos, así como los módulos que se encargan de realizar el tratamiento de flujo de datos entre los dispositivos y la **placa DE2**, para lograr acondicionar estos módulos a las necesidades del Sistema a desarrollar.

4.2.1. Sistema ejemplo DE2_D5M para cámara D5M

La cámara es el primer dispositivo que se utilizará, el cual contiene parte del sistema que utilizaremos para desarrollar el proyecto. Su principal función es la captura de las imágenes por la **cámara D5M** para enviarlas a una pantalla VGA.
En la siguiente figura se puede observar la interacción del Sistema **DE2_D5M** con los dispositivos.

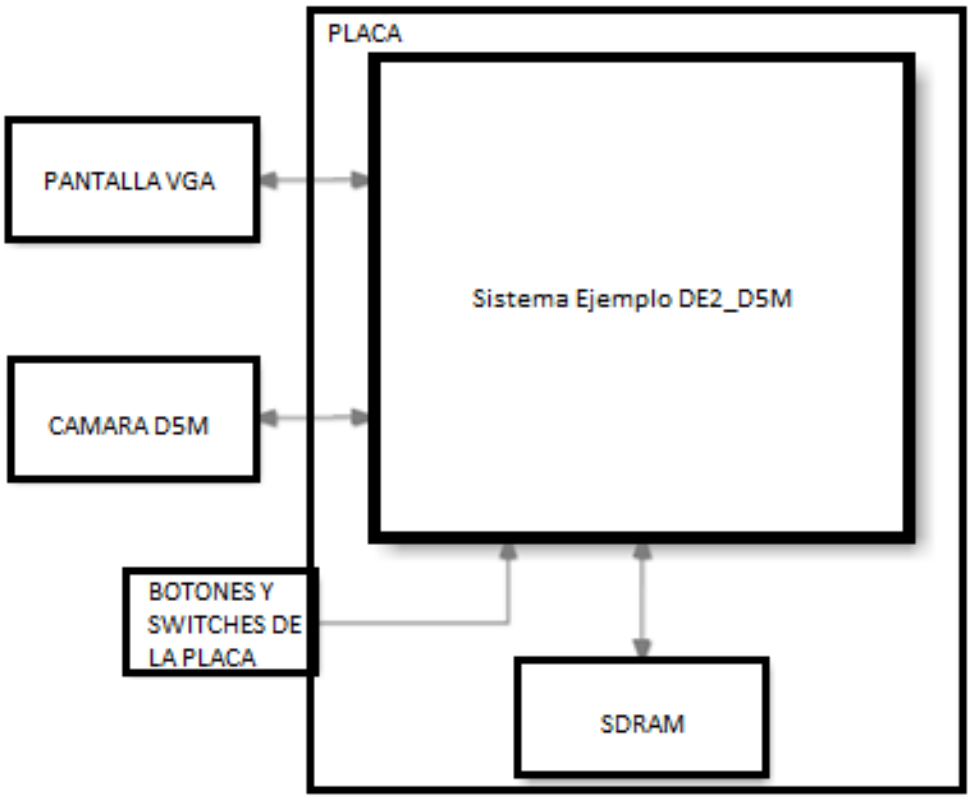
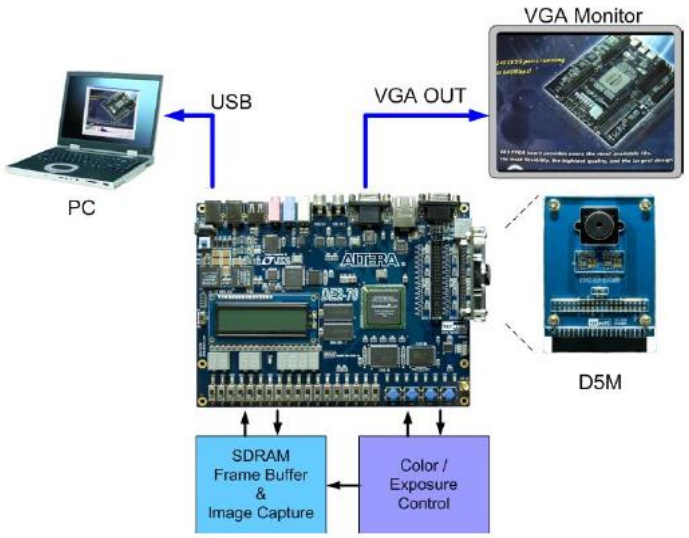


Figura 10: Esquema del Sistema (DE2_D5M) para la cámara

Dentro de este sistema hay unos módulos que se encargan de la inicialización de ciertos registros referentes a la configuración interna de la **cámara D5M**. Antes de que el Sistema se ponga en marcha, hay que reiniciar sus registros de funcionamiento pulsando **KEY [0].** Esta acción reinicia la configuración de los registros internos de la cámara con un valor para que funcione la cámara adecuadamente a lo esperado.

También encontraremos unos módulos encargados de la captura de los datos de las imágenes de la cámara. Captura los datos de los tres colores (RGB) de cada pixel de la cámara en el momento que se le pulsa **KEY [3].** Para recoger la señal asociada a la imagen se utilizan algunos pines del puerto **GPIO_1 [1:13]** en el que está conectada la cámara.

Del mismo modo tendremos módulos para la conversión de estos datos a **datos RGB** para almacenarlos en la **memoria SDRAM**. Los almacena en la **memoria SDRAM** separadamente con cada color (rojo, verde y azul). Los datos recogidos se irán procesando a través de unos módulos internos que nos adaptan su formato al de la pantalla VGA para que se vea de forma correcta en pantalla.
Estos módulos enviarán dichos **datos RGB** a la pantalla VGA para que se muestren las imágenes en la pantalla.
En el momento de mostrar las imágenes capturadas en la pantalla VGA, se realiza la lectura de la **memoria SDRAM** para enviarlos a ella.
Encontraremos también un módulo que se encarga de la gestión en el almacenamiento y lectura de datos en la **memoria SDRAM**.
También tendremos la posibilidad de variar la luminosidad de la imagen mediante **SW [0]** y pulsando constantemente **KEY [1].**
Para finalizar y como detalle final se podrá congelar la imagen un momento dado mediante **KEY [2].**



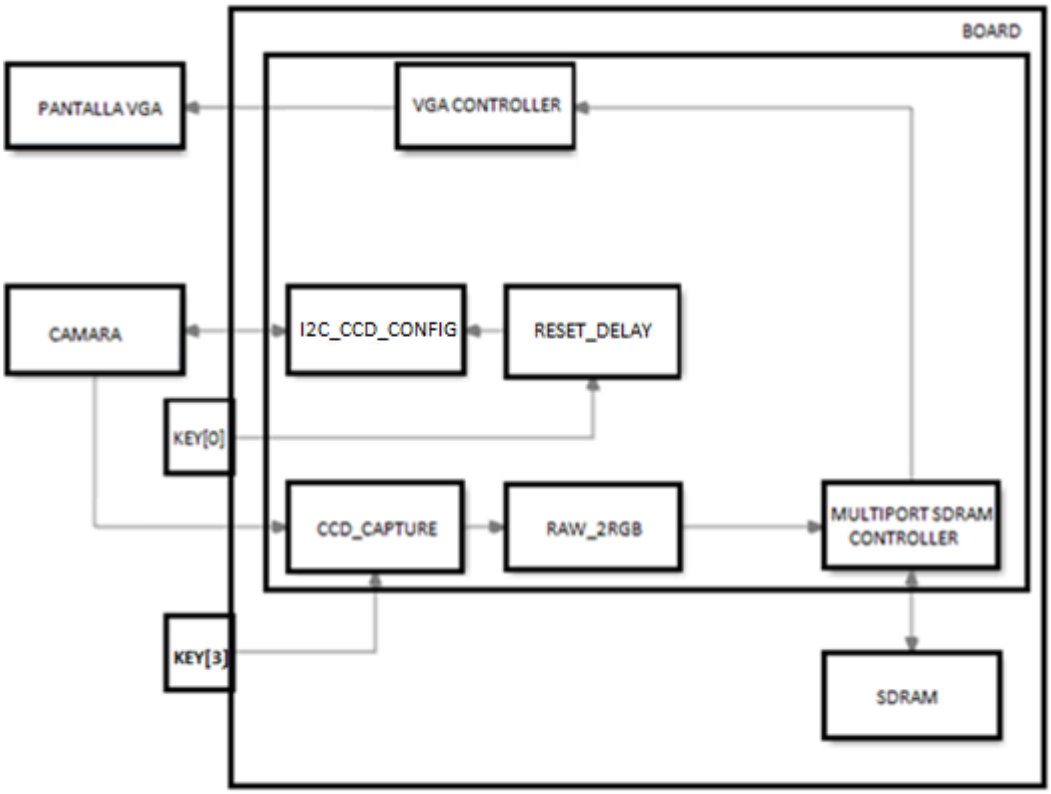


Figura 11: Interconexión de la placa DE2 con la pantalla VGA y la cámara D5M.

Haremos una breve descripción de cada módulo utilizado en el sistema ejemplo para entrar más en detalle en la siguiente fase del proyecto (Fase de Diseño e implementación):

- La inicialización de los registros se realiza en el módulo **I2C_CCD_Config**. Con estos valores se configura el funcionamiento interno y el control en el flujo de datos de las imágenes que van hacia el exterior. Esta inicialización se realizará en el momento que se pulse el botón **KEY [0],** que desde el módulo **Reset_Delay**, resetearemos este módulo y otros módulos de este Sistema.

- Mediante el módulo **CCD_Capture** se realizará la captura de los datos RGB de los pixeles de las imágenes desde la cámara de forma continua en el momento que pulsemos **KEY [3].**

- A continuación, se envían los datos capturados al módulo **RAW2RGB**, donde en este módulo se convierten a formato **RGB**.

- Gracias al módulo controlador multipuerto **Sdram_Control_4Port,** podremos gestionar las lecturas/escrituras solicitadas. Los datos con formato **RGB**, se almacenarán en la **SDRAM**, de manera que el

formato de sus bits se adapta de tal forma al formato propuesto por la pantalla VGA. La **SDRAM** en este caso funciona como memoria para la cámara.

- Posteriormente el módulo **VGA_Controller** gestiona el envío de los datos almacenados en la SDRAM a la pantalla VGA de forma continua, para mostrar finalmente las imágenes capturadas en pantalla en forma de video.

4.2.2. Sistema ejemplo DE2_LTM_Ephoto para LCD

El otro dispositivo ya mencionado anteriormente es la **pantalla LCD.** Utilizaremos el sistema ejemplo de este dispositivo para el desarrollo del proyecto. Se encarga primordialmente, de capturar las imágenes desde una memoria Flash y enviarlas a la **pantalla LTM**. Aparte también se tiene la posibilidad de poder recorrer táctilmente la superficie de la pantalla, mientras en el **Display de 7 segmentos** aparece la posición (X, Y) en la que se encuentra el dedo. En el momento que se toca la parte superior izquierda de la pantalla, se cambia la imagen que se muestra en ella por otra imagen guardada en la memoria Flash.

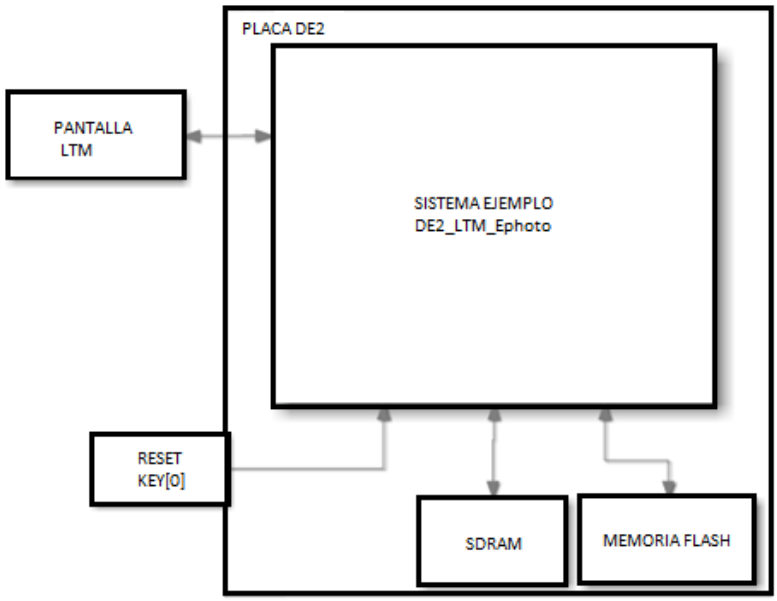


Figura 12: Esquema del Sistema (DE2_LTM_Ephoto) para la pantalla TRDB-LTM.

En este sistema también habrá un módulo encargado de inicializar los dispositivos configurando los registros internos de la **pantalla LTM** que se encargan de controlarlo internamente.
Hay un módulo encargado para la captura de las imágenes de la memoria Flash y el almacenamiento en la **memoria SDRAM**. La **memoria SDRAM** que almacena los datos de los pixeles de la imagen capturada, debe ser controlada por un módulo que gestiona la escritura y lectura de datos en ella. También existe un módulo que envía **datos RGB** desde la **memoria SDRAM** a la **pantalla LTM**, para la visualización de la imagen.
Por último, la detección táctil de la pantalla y el cambio de imagen mostrada en pantalla son controlados por sendos módulos pensados para ello.
Previamente a todo esto hay que cargar las imágenes que tenemos en el ordenador a la memoria Flash, mediante el programa **DE2_Control_Panel**. A continuación, se desactiva el uso que está haciendo dicho programa con la conexión USB, y se carga el Sistema ejemplo **DE2_LTM_Ephoto**. Con lo que ya se pondría este Sistema en marcha.
Cuando el sistema funciona normalmente, se capturan los datos de cada uno de los pixeles asociados a la imagen almacenada en la memoria flash. Estos datos se almacenan en la **memoria SDRAM**, de manera que se envía a esta memoria por separado tres datos RGB de cada pixel correspondientes a los colores rojo, verde y azul, y así posteriormente enviar estos datos almacenados al **LCD TRDB_LTM**, para poder visualizar la imagen almacenada en la memoria flash.
Finalmente existe la posibilidad de recorrer la superficie de la pantalla de manera táctil, y poder saber en el **Display de 7 segmentos**, que posición (X, Y) de la superficie da la pantalla se está palpando. Dependiendo de si se palpa la parte superior izquierda, existe la posibilidad de poder cambiar la imagen que se muestra en la pantalla.

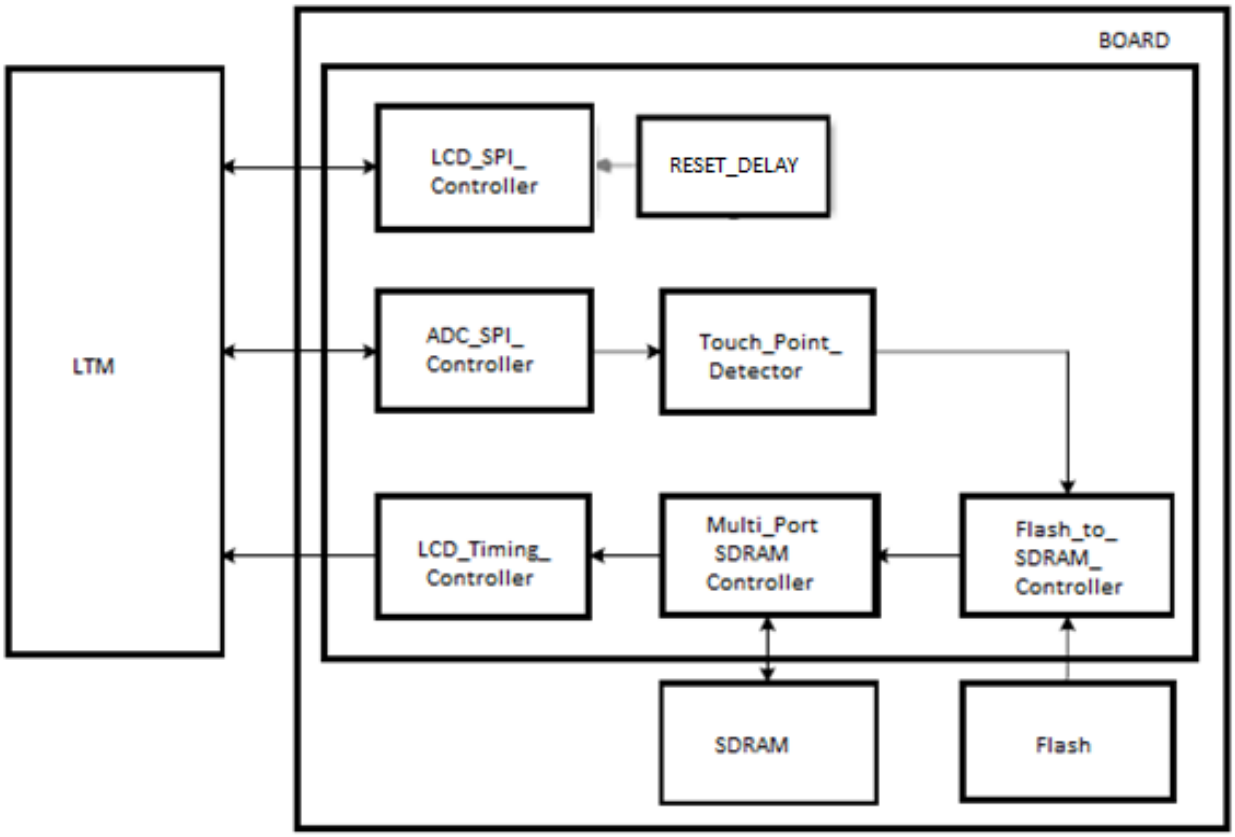


Figura 13: Interconexión de la placa DE2 con la pantalla LTM.

Haremos una breve descripción de cada módulo utilizado en el sistema ejemplo para entrar más en detalle en la siguiente fase del proyecto (Fase de Diseño e implementación):

- Habrá un módulo para la captura de datos de la imagen de la memoria Flash **(flash_to_sdram_controller).**

- Otro módulo se encargará del control de la **memoria SDRAM (Sdram_Control_4Port),** y otro módulo se encargará del control en el envío de datos a la **pantalla LTM**, para la muestra de la imagen en ella **(lcd_timing_controller).**

- El módulo que se encarga de la inicialización de los registros internos de la pantalla LTM es **lcd_spi_cotroller**. También habrá un módulo de reseteo que es el que dé la orden de reinicio del sistema, con sus módulos y los parámetros de configuración de los registros del LTM **(Reset_Delay).**

- Por último, el módulo que se encarga de rescatar la posición (X, Y) de la pantalla que se está palpando es **adc_spi_controller**. El que se encarga de mostrar está posición en el **Display de 7 segmentos** es **SEG7_LUT_8**, y dependiendo de si la posición de la pantalla que nos encontremos palpando es la parte superior izquierda o no, se mostrará la siguiente imagen almacenada en la memoria Flash, enviando una orden al módulo **flash_to_sdram_controller**, para que nos muestre la siguiente imagen almacenada.

5

Fase de Diseño e implementación

5.1. Diseño de la estructura del nuevo sistema

El principal objetivo en esta fase consistirá en realizar la estructura correcta del Sistema a desarrollar, utilizando los módulos necesarios de los sistemas ejemplo vistos en la anterior fase.
Antes de comenzar a desarrollar el sistema tenemos que saber que tiene que ser compatible con ambos dispositivos.
Se pretende capturar las imágenes de la **cámara THDB-D5M** y visualizarlas en la **pantalla THDB-LTM**. También se podrá modificar el brillo de la imagen pulsando en la pantalla y podremos congelar la imagen tanto desde la placa pulsando **KEY [2]** o pulsando en la pantalla táctil en el lugar que indicaremos más adelante.
También tendremos que tener en cuenta los parámetros que se tendrán que modificar para que la imagen capturada se vea en la pantalla correctamente. Para ello se tendrá que conseguir la misma resolución compatible para ambos dispositivos y una sincronización entre ambos dispositivos.
Partiendo de esta base y viendo todos los **PDFs** explicativos de ambos dispositivos se utilizará 800x480 como resolución estándar.
Esta resolución es la que hay que tener en cuenta a la hora de configurar los parámetros.
Entre estos aspectos están los de la sincronización de los diferentes elementos del Sistema. Con estos valores se configuran sus registros internos.
Tal y como hemos visto en la anterior fase del proyecto, el primer paso es el de reinicializar la configuración de los registros internos de la **cámara D5M** y la **pantalla LTM**. Con ello serán necesarios los módulos que gestionen el reinicio de estos dispositivos. Los módulos que se encargan del reinicio de los

dispositivos, son **lcd_spi_controller** para la **pantalla LTM** e **I2C_CCD_Config,** para la **cámara D5M.**
El módulo que provoca el reseteo del Sistema es **Resert_Delay.** Este módulo se ha recogido del Sistema de ejemplo **DE2_D5M.**
También se debe de incorporar el módulo **CCD_Capture** para poder capturar las imágenes de la cámara de forma continua en el momento que se pulse **KEY [3].** Es necesario procesar los datos capturados, con la intención de que salgan tres datos RGB en 3 registros, uno por cada color, por lo que necesitaremos el módulo **RAW2RGB** para realizar este procesamiento.
Se necesitará un módulo controlador de **memoria SDRAM** que nos almacene y nos lea los **datos RGB** en dicha memoria. Para ello se partirá como base el módulo controlador de la SDRAM (**Sdram_Control_4Port**) del ejemplo de aplicación para la **cámara DE2_D5M**. A partir de este módulo se modificarán los registros de entrada y salida, los formatos de los bits de estos registros y parámetros para que sean compatibles con los dos dispositivos.
Por último, se necesitará el módulo **lcd_timing_controller** para que gestione y sincronice el envío de los datos de los pixeles de las imágenes a la **pantalla LTM.**
Una vez seleccionados todos los módulos necesarios para el nuevo Sistema vemos que los módulos que interactúan con la memoria flash y la pantalla VGA no van a ser necesarios.
Estos módulos son: **flash_to_sdram_controller**, módulo que captura una imagen que está almacenada en la memoria flash; **VGA_Controller**, módulo que se encarga del envío de datos de las imágenes almacenadas en la SDRAM hacia la pantalla VGA.
A continuación, se muestra un esquema base de cómo se distribuye la estructura del Sistema a desarrollar:

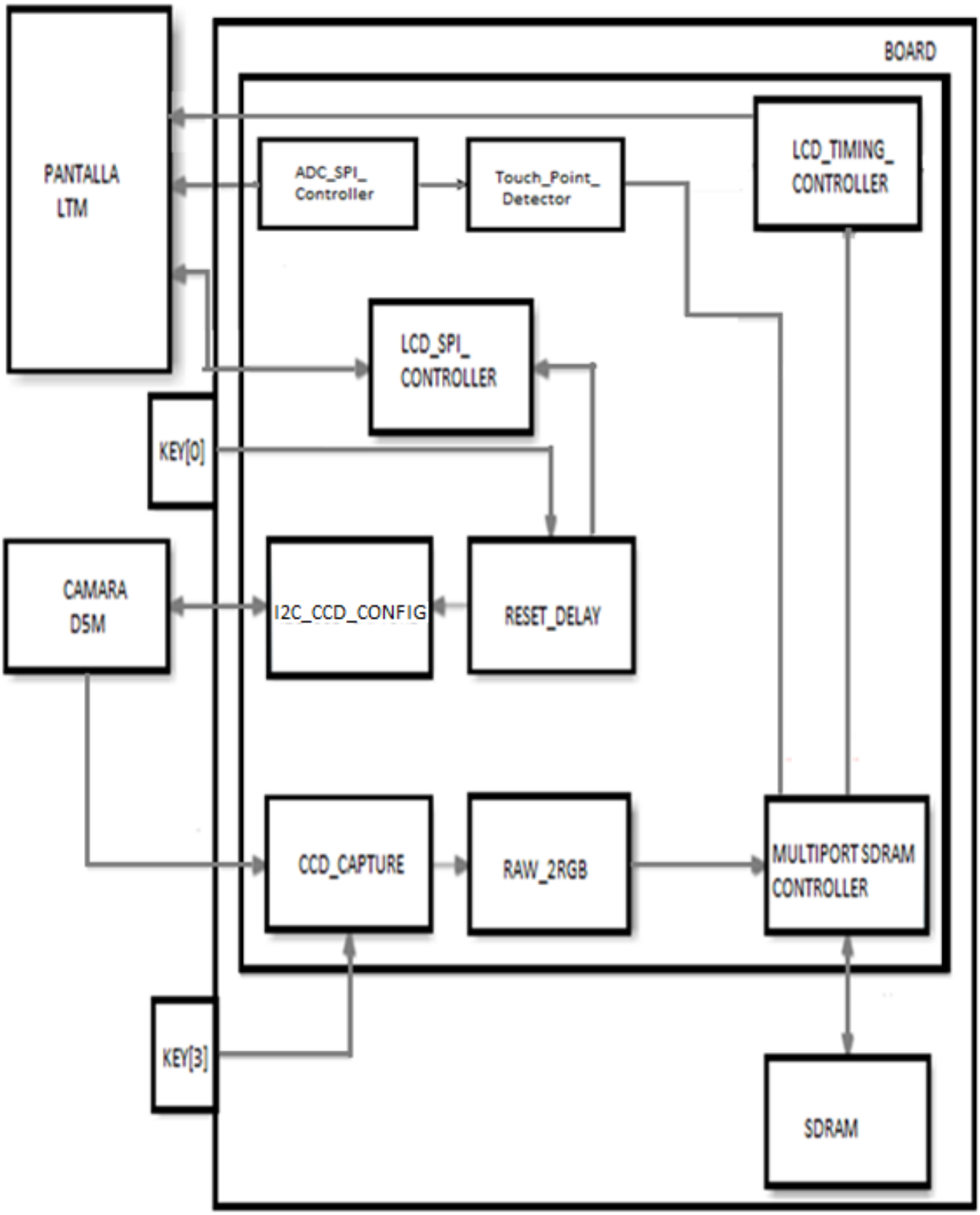


Figura 14: Interconexión de los módulos del Sistema a desarrollar.

5.2. Módulos añadidos en el sistema

En este apartado se detallarán uno a uno los módulos que se van a incluir en el proyecto:

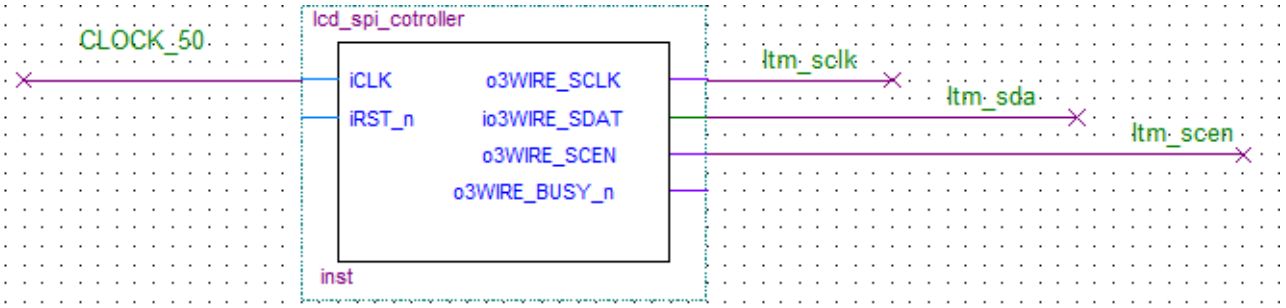
Lista de módulos por cada dispositivo:

- **Cámara D5M:**
    - **I2C_CCD_Config:** Permite reiniciar los registros internos del dispositivo, para configurar el funcionamiento (resolución, intensidad del color, brillo).
    - **CCD_Capture:** Captura de forma continuada los datos(pixeles) recogidos por la cámara.
    - **RAW2RGB:** Módulo que transforma los datos de los pixeles a formato **RGB**. Esto lo realiza de manera que los envía a 3 registros para procesarlos en 3 colores: rojo **(sCCD_R),** verde **(sCCD_G),** azul **(sCCD_B)** por cada pixel.
- **Pantalla LTM:**
    - **LCD_SPI_Controller:** Módulo que reinicia los parámetros de los registros de la **pantalla LTM**, que son los que configuran el funcionamiento interno del dispositivo.
    - **LCD_timing_Controller:** Módulo que gestiona el procesamiento y envío de **datos RGB** de la **memoria SDRAM** a la **pantalla LTM.**
    - **Adc_spi_Controller:** Módulo que va a permitir la conversión de analógico a digital de las **coordenadas x e y** del punto de la pantalla palpado.
- **Módulos comunes para ambos dispositivos:**
    - **Reset_Delay:** Genera señales de reset retardadas para permitir la inicialización escalonada de diferentes módulos del sistema.

- **Sdram_Control_4Port:** Módulo controlador de la SDRAM, que va a funcionar cómo una interfaz entre los módulos del sistema y la **memoria SDRAM**. Con este módulo se gestiona la lectura y escritura de datos en dicha memoria.

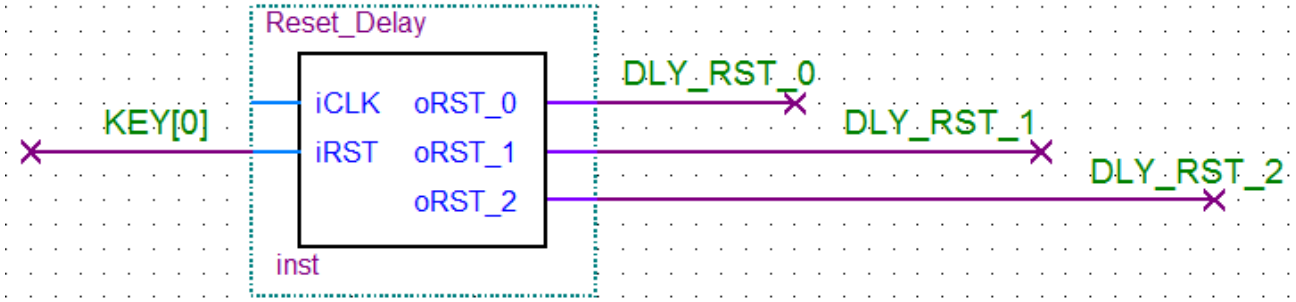
Módulos paso a paso:

lcd_spi_cotroller



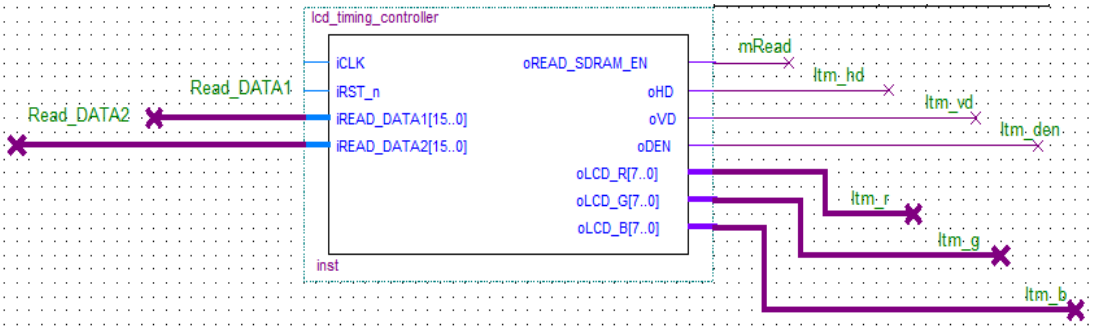
Módulo que reinicia los parámetros de los registros de la **pantalla LTM**, que son los que configuran el funcionamiento interno del dispositivo.
Ejemplo: resolución, flanco de señales de sincronización…
Mediante el registro de salida **ltm_sda** de este módulo, se envían estos datos de configuración de los registros internos al **LCD**. El envío de datos hacia el LCD se controla a través del registro de salida **ltm_sclk**, que sincroniza este envío de datos.
Este reinicio en la configuración de los registros, se producirá en el momento que se produzca un reseteo en el sistema. Esto ocurrirá al pulsar **KEY [0]** a través del módulo **Reset_Delay**.

Reset_Delay



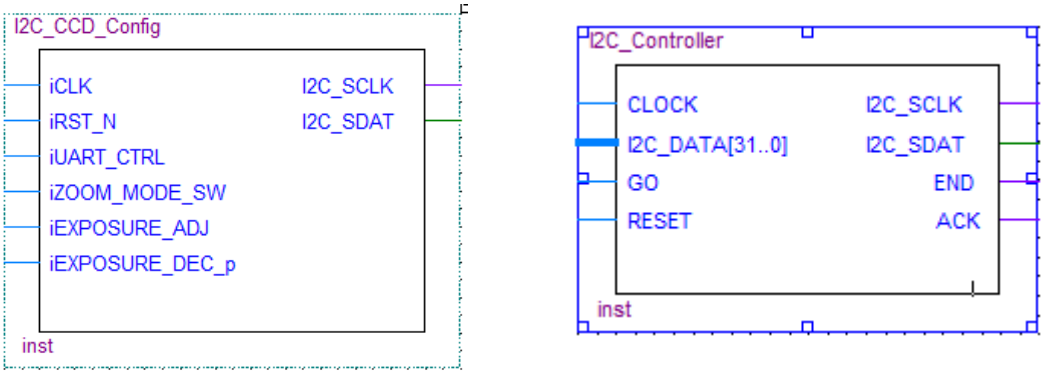
Este módulo extraído del diseño de la **cámara DE2_D5M**, permite reiniciar algunos parámetros y módulos del sistema al pulsar el botón **KEY [0].**
Este módulo genera señales de reset retardadas para permitir la inicialización escalonada de diferentes módulos del sistema.
Utilizando este módulo como base tendremos que eliminar el mismo módulo utilizado en el caso de ejemplo de la **pantalla LTM** y adaptar este módulo para que sirva como reseteo de los dos dispositivos.
Por lo tanto, todas las entradas de reset de los módulos DLY0, DLY1, DLY2, serán sustituidas por DLY_RST_0, DLY_RST_1, DLY_RST_2.

lcd_timing_controller



Módulo que gestiona el procesamiento y envío de **datos RGB** de la **memoria SDRAM** a la **pantalla LTM.**
Desde la memoria SDRAM llegarán a este módulo, a través del registro de entrada **Read_DATA1**, los datos correspondientes a los colores rojo (8 bits de mayor peso) y verde (8 bits de menor peso), y a través del registro de entrada **Read_DATA2** el dato correspondiente al color azul (8 bits de menor peso).
Una vez procesados estos datos, saldrán los tres colores por separado en 3 registros diferentes de formato de 8 bits, **ltm_r** (rojo), **ltm_g** (verde), **ltm_b** (azul).
Mediante los registros de salida **ltm_nclk**, **ltm_hd**, **ltm_vd**, se controla la sincronización en el envío de los datos de los pixeles al **LCD**. Con ello se consigue visualizar la imagen en dicha pantalla.

I2C_CCD_Config



Módulo que permite reiniciar los registros internos de la **cámara D5M**, para configurar el funcionamiento (resolución, intensidad del color, brillo).
Módulo que interviene en este módulo: **I2C_Controller.**
También en este módulo se variará la configuración de la luminosidad, para la imagen capturada en el momento que se pulsa de forma táctil la pantalla. Se verá a continuación de una forma gráfica las zonas de la pantalla para subir o bajar la luminosidad de la imagen.

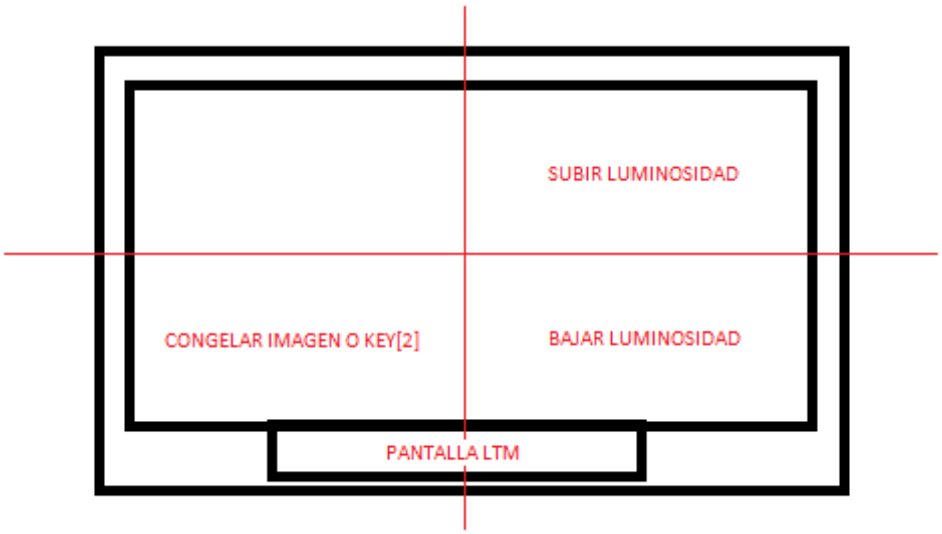
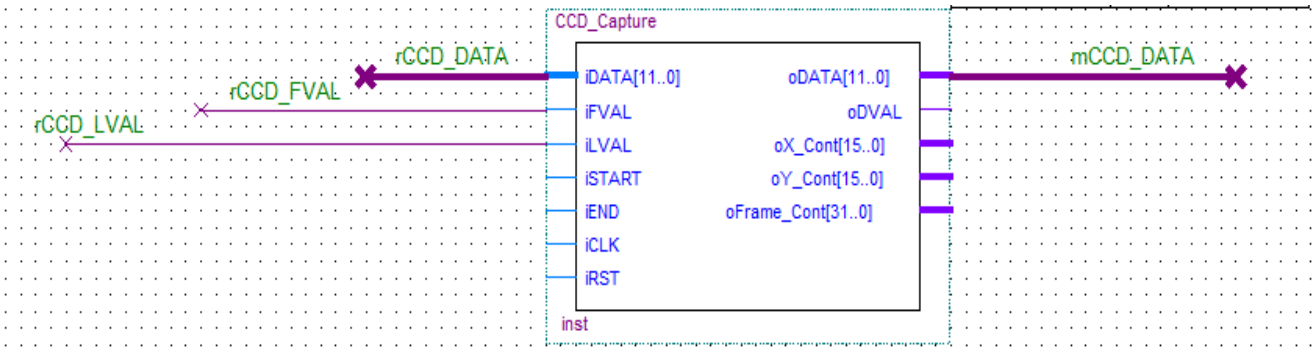


Figura 15: Zona táctil para subir o bajar la luminosidad.

Para realizar el envío de los datos de configuración de los registros internos de la cámara, este módulo también establece un protocolo de comunicación vía serie.

Los datos asignados a cada registro se enviarán a la cámara mediante el registro de **salida. I2C_SDAT(GPIO_1[23]).**
El registro de salida. **I2C_SCLK(GPIO_1[24]),** será la señal de reloj que llegue al **THDB-D5M** a través de su entrada SCLK. Con este reloj se sincroniza el envío de datos a este dispositivo, durante el proceso de su configuración.

CCD_Capture



En el momento de pulsar **KEY [3],** este módulo captura de forma continuada el dato del pixel recogido de la cámara **(rCCD_DATA)** que será enviado al módulo **RAW2RGB** a través del registro de salida **mCCD_DATA**, siempre que nos encontremos dentro del ancho de la resolución **(LVAL! =0),** porque si no este módulo nos devuelve **mCCD_DATA**=0.
Tendremos la opción de congelar la imagen pulsando **KEY [2]** o tocando una parte de la pantalla táctil que se verá en la siguiente imagen.

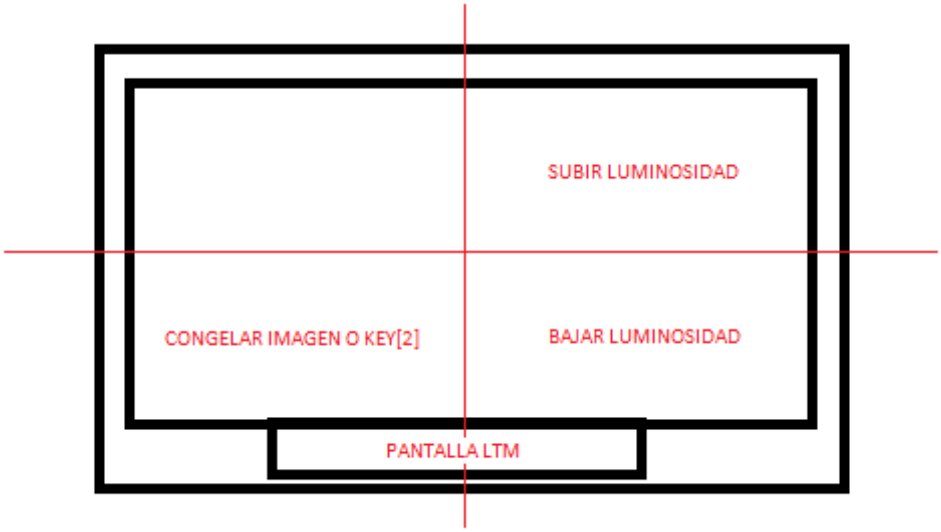
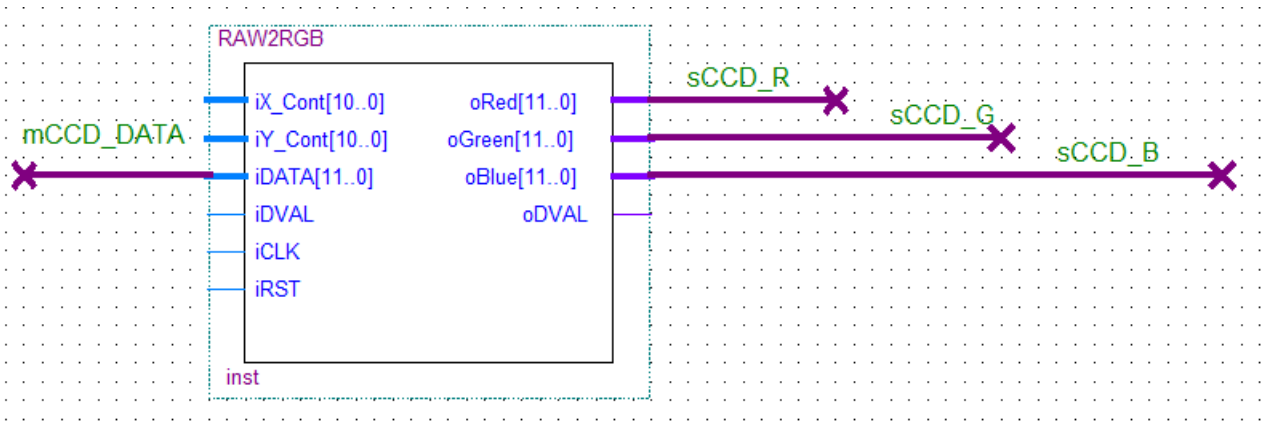


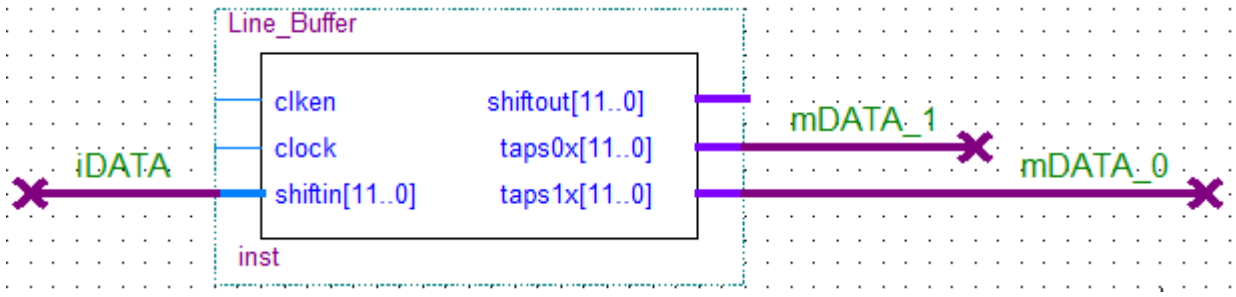
Figura 16: Zona táctil para congela la imagen.

De esta forma se congela la imagen en la pantalla, dejando de capturar datos de los pixeles e informando posteriormente a la **SDRAM**, que se envíe la información a la **pantalla LTM** de la última imagen guardada.

RAW2RGB

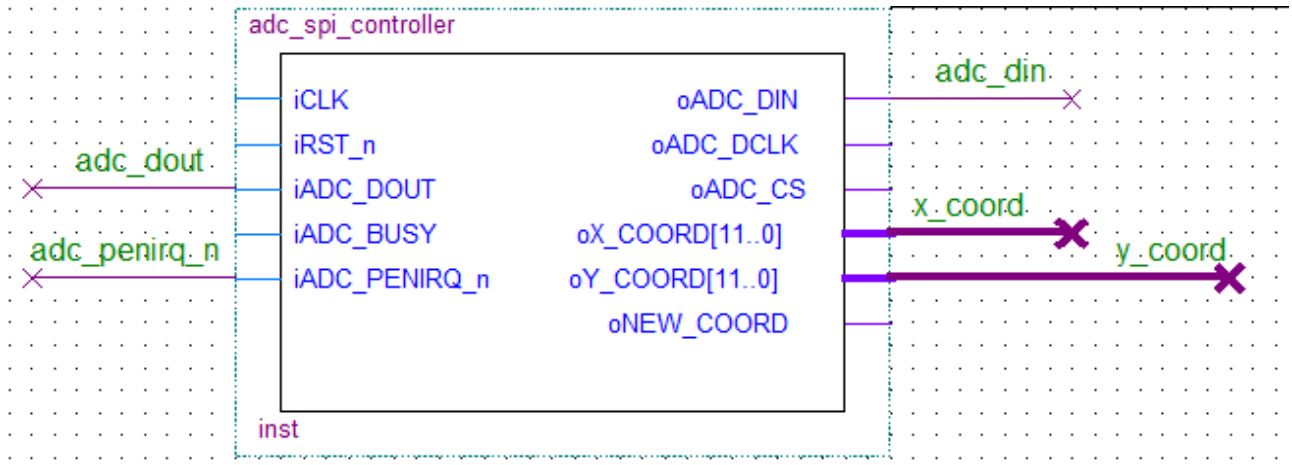


Módulo que transforma los datos de los pixeles a formato **RGB**. Esto lo realiza de manera que los envía a 3 registros para procesarlos en 3 colores: rojo **(sCCD_R),** verde **(sCCD_G),** azul **(sCCD_B)** por cada pixel. Posteriormente estos datos se puedan almacenar en la **memoria SDRAM** a través del módulo multipuerto **Sdram_Control_4Port**, que funciona a modo de controlador de la **SDRAM**.
Dentro del módulo **RAW2RGB** interviene también el módulo **Line_Buffer.**



Este módulo nos dará unos valores de salida (**mDATA_1** y **mDATA_0**). Estos se utilizarán para realizar la conversión en tres registros con los datos de los tres colores **RGB**, rojo **(mCCD_R)**, verde **(mCCD_G)** y azul **(mCCD_B).**

adc_spi_controller

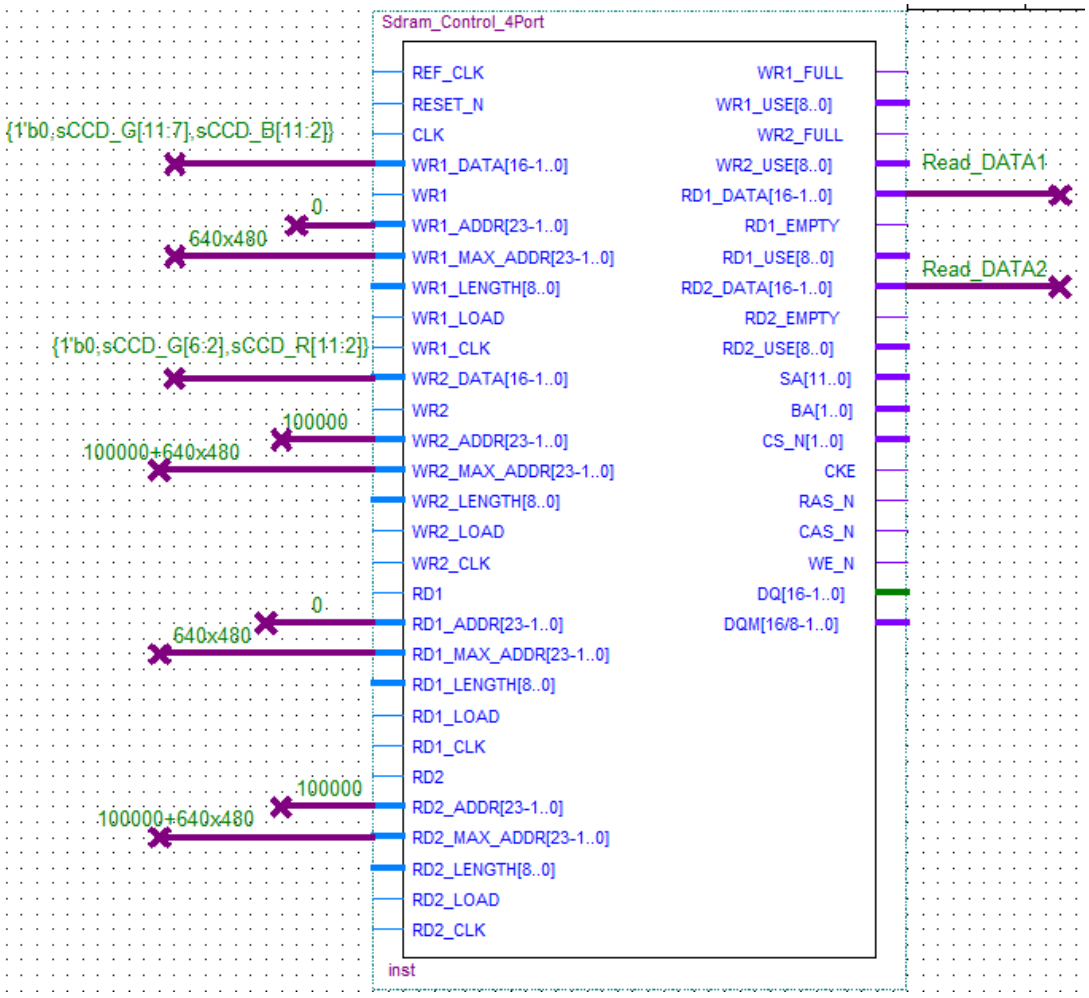


Módulo que va a permitir la conversión de analógico a digital de las **coordenadas x e y** del punto de la pantalla palpado. Esta conversión se realiza a través de la gestión del chip conversor analógico-digital ADC7843. Así se puede observar que el registro de entrada **adc_penirq_n** es el que detecta que se ha producido un palpado de la pantalla. En ese momento este módulo está en disposición de enviar la palabra de control a través del registro de salida **adc_din**. Posteriormente este módulo capturará las coordenadas donde se ha producido el palpado de la pantalla a través del registro de entrada **adc_dout**, para que nos procese en dos registros por separado tanto para la coordenada x cómo la y, a través de los registros de salida **x_coord** e **y_coord.**

SEG7_LUT_8

Módulo que muestra la posición x e y en el **Display de 7 segmentos**. Estos datos de posición le llegan desde el módulo **adc_spi_controller**, que gestiona la detección de las coordenadas.

Sdram_Control_4Port

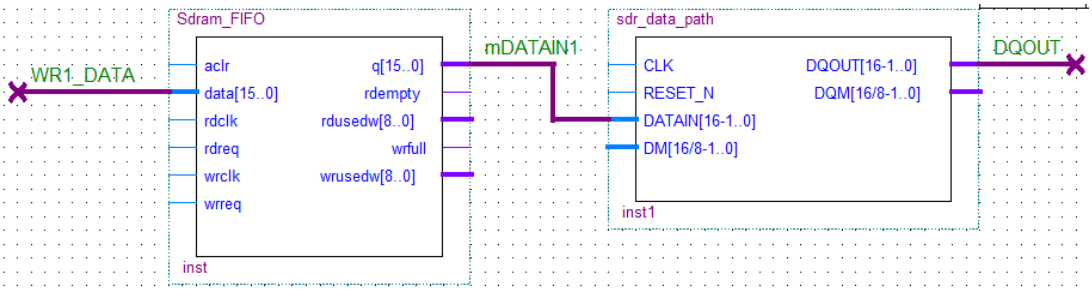


Módulo controlador de la SDRAM, que va a funcionar cómo una interfaz entre los módulos del sistema y la **memoria SDRAM**.
Con este módulo se gestiona la lectura y escritura de datos en dicha memoria. Es el controlador que almacena los **datos RGB** de los pixeles capturados de la cámara en la **SDRAM**. Posteriormente los lee para enviarlos al módulo **(lcd_timing_controller),** encargado de la gestión en el envío de los datos a la **pantalla LTM.**
Estos datos llegarán a este módulo controlador de la **SDRAM**. Este a través de sus submódulos **Sdram_FIFO write_fifoX** y **sdr_data_path data_path1.**

El propio control del flujo de datos se realiza a través de sus submódulos **control_interface control1** y **command command1**, que gestionan las órdenes de control de dicha memoria. Estas señales de control se conectarán también con la **SDRAM** a través de las salidas del módulo controlador.

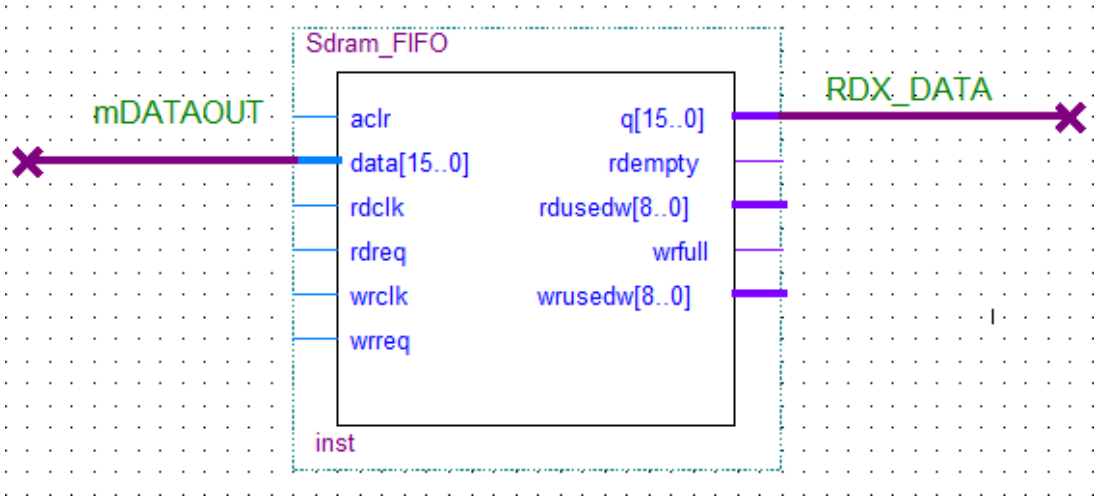
Los submódulos que interactúan con el módulo **Sdram_Control_4Port:**

Sdram_FIFO write_fifoX y sdr_data_path data_path1



Módulos que gestionan la escritura de los datos **(WR1_DATA y WR2_DATA)** de las imágenes recogidas por la cámara que van llegando a través del módulo controlador de la **SDRAM**.

Sdram_FIFO read_fifo1



Módulo que gestiona la lectura de los datos de la **memoria SDRAM**, y enviarlos al módulo **lcd_timing_controller** a través de los registros de salida **(RD1_DATA y RD2_DATA)** del módulo controlador de **SDRAM**.
Posteriormente **lcd_timing_controller,** gestionará la visualización en pantalla de las imágenes.

Por otro lado, hay que destacar las entradas del módulo controlador de la **SDRAM (Sdram_Control_4Port)**. **WRX(sCCD_DVAL),** que habilita la escritura en la **memoria SDRAM**. Lo mismo que. **RDX(Read)** habilita la lectura.
Además, se indica dirección de inicio del almacenamiento de los datos de los colores azul y los bits de mayor peso del verde (**. WR1_ADDR (0),** para la escritura, **RD1_ADDR (0),** para la lectura), y dirección final **(.WR1_MAX_ADDR(640*480)** (escritura**), .RD1_MAX_ADDR(640*480)** (lectura)). Para el almacenamiento del dato del color rojo y los bits de menor peso del verde, la dirección de inicio será **(.WR1_ADDR(22'h100000),** para la escritura, **.RD1_ADDR(22'h100000),** para la lectura), y la dirección final **(.WR1_MAX_ADDR(22'h100000+640*480)** (escritura), **.RD1_MAX_ADDR(22'h100000+640*480)** (lectura)).
También se tiene en cuenta el clock y el reseteo del módulo. El clock de este módulo procederá directamente desde el clock de 50 MHz que viene a través de la interfaz de la placa.
Con esto se consigue, un acceso bidireccional a los datos que se almacenan en la memoria SDRAM, a través del puerto bidireccional de entradas y salidas **DRAM_DQ.**

5.3. Modificaciones en el diseño del sistema

En este apartado se realizarán las modificaciones necesarias para adaptar los dos dispositivos de forma que se consiga el objetivo final.
Se realizarán cambios tales como; nombres de registros, formatos en número de bits, frecuencias de sincronización del sistema, parámetros, etc.…)
Se irán modificando registros y módulos de tal forma que se consiga realizar un sistema en el que puedan funcionar los dos dispositivos **(D5M y LTM)** a través de la **placa DE2.**
Una de las modificaciones más importantes viene dada por la resolución de pantalla ya que en el ejemplo de la **pantalla LCD** la resolución es distinta a la del ejemplo de la **cámara D5M** que trabaja con una pantalla VGA. Por lo tanto, se tendrá que adaptar la resolución a una resolución de 800x480 para ambos dispositivos, para que la imagen capturada en la cámara sea compatible en la **pantalla LTM**, a la hora de visualizarla.
A continuación, se realizarán modificaciones en el desarrollo del módulo principal, tanto en los nombres de los registros de entrada y salida, cómo en sus formatos. Posteriormente se hará lo mismo con el resto de módulos, teniendo en cuenta también las modificaciones de los parámetros de los registros internos de los dispositivos.

5.3.1. Modificaciones en el módulo principal

Para comenzar se han suprimido las declaraciones de los registros internos de los módulos que se han eliminado ya que no son necesarios.
Del mismo modo se han declarado los registros internos de entrada y salida de los módulos que hemos incluido.
Para recoger las señales de la cámara tal como en el ejemplo del mismo, tendremos que realizar una lectura constante de sus pixeles que vendrán dados a través del puerto **GPIO_1.**
Por lo tanto, se debe de introducir la declaración de las entradas y salidas de dicho puerto, así como las asignaciones de comunicación con los registros de los módulos, que en nada varían respecto al sistema de **ejemplo DE2_D5M**.

Del mismo modo, para enviar constantemente los **datos RGB** al **LCD LTM,** se realizará a través del puerto **GPIO_0**.
Tendremos que introducir la declaración de entradas y salidas de dicho puerto, así como las asignaciones de comunicación con los registros de los módulos. Que tampoco varían en nada respecto al sistema de **ejemplo DE2_LTM_Ephoto.**
La frecuencia de reloj de salida de la **cámara D5M (PIXCLK)** es igual que **XCLKIN** porque internamente se ha configurado así en los registros del dispositivo.
Mediante las siguientes líneas podremos observar que a la **cámara D5M**, le llega una frecuencia de 25 MHz debido a la frecuencia que le entra a través de la entrada de reloj **XCLKIN**. Esto se puede observar a continuación en las siguientes líneas del módulo principal:

```
**********************************************************
always@ (posedge CLOCK_50) rClk<=rClk+1;//frecuencia rClk 50 MHz
assign CCD_MCLK = rClk [0]; //realiza una división de frecuencia de 50 MHz/2
assign GPIO_1[16]   =        CCD_MCLK; //entrada de XCLKIN
**********************************************************
```

Por lo tanto, la frecuencia que sale **PIXCLK**, para sincronizar la captura de los **datos RGB** de la **cámara D5M** para almacenarlos en la **SDRAM**, será 25 MHz. Esto también se hará así en el Sistema a desarrollar.
Una vez seleccionada la frecuencia de reloj con la que se va a trabajar tendremos que mirar si el otro dispositivo está trabajando a la misma frecuencia.
La **pantalla LTM** debería de sincronizar la recepción de los **datos RGB** de los pixeles almacenados en la **SDRAM** a la misma frecuencia, es decir a 25 MHz. Por lo tanto, tendremos que modificar los registros de tal forma que la frecuencia sea la misma que la de la cámara.
Para dicha entrada de reloj se le asigna el divisor de frecuencia 50 MHz/2, y se niega este valor porque está configurado internamente, para que capture la **pantalla LTM** datos del exterior en el flanco de bajada de **nclk**.

```
*******************************************************
assign GPIO_0[9] = ltm_nclk; //entrada de clock de la pantalla LTM
assign ltm_nclk =   ~rClk[0];   //realiza una división de frecuencia de 50 MHz/2
                    //invirtiendo la señal de rClk[0]
*******************************************************
```

Al seleccionar la frecuencia de reloj del ejemplo de la **cámara D5M**, se ha decidido utilizar el módulo de reseteo del mismo dispositivo.
De este modo los registros de salida que dan la señal a cada módulo para producir este reseteo son DLY_RST_0, DLY_RST_1, DLY_RST_2, y no DLY0, DLY1, DLY2. Con lo que habrá que sustituir estos últimos registros de salida por DLY_RST_0, DLY_RST_1, DLY_RST_2.

5.3.2. Modificaciones en el módulo Sdram_Control_4Port

En este apartado se explicarán las modificaciones que se han realizado en el diseño de este módulo, tanto en registros, formatos de los bits, como en algunos parámetros de la cabecera de dicho módulo.
También se analizarán algunos cambios que se realizan en el interior del módulo.
Primeramente, se analizarán los cambios a realizar en la cabecera del módulo. Hay que tener en cuenta de que a la **pantalla LTM** le llegan 3 datos **RGB** de 8 bits por cada pixel de la imagen a mostrar. Los **datos RGB** de cada pixel **(sCCD_R, sCCD_G, sCCD_B),** llegan al módulo controlador con un formato de 12 bits desde el módulo **RAW2RGB**. Según esto, por cada registro de cada color se deben truncar los bits seleccionando los 8 bits de mayor peso, que son los que aportan mayor información de los datos **RGB** de cada pixel. Por lo tanto, con este formato de 8 bits se almacenan en la memoria **SDRAM**. Con ello, se compatibiliza con el formato de los datos **RGB** de los pixeles, necesario para mostrar las imágenes en la **pantalla LTM**.
Quedando estos registros de entrada de la siguiente manera:

- **.WR1_DATA({sCCD_R[11:4],sCCD_G[11:4]})**, para almacenar en la **memoria SDRAM** el dato correspondiente a los colores rojo y verde; y **.WR2_DATA({8'h0,sCCD_B[11:4]}),** para almacenar el dato del color azul.

- **.WRX(sCCD_DVAL),** recibe una señal desde el módulo **RAW2RGB** para habilitar la escritura de los datos en la **memoria SDRAM**.

- **.WR1_ADDR(0)** y **.WR2_ADDR(22'h100000),** son los registros de entrada donde enviamos la dirección de comienzo de almacenamiento de datos.

- **.WR1_MAX_ADDR (800*480)** y **.WR2_MAX_ADDR (22'h100000+800*480),** son los registros de entrada que envían la dirección final de almacenamiento de datos. Se puede observar que la dirección final 800*480, es la resolución o el número de pixeles de la imagen que se almacenan en la **memoria SDRAM.** Por ello que se ponga este número cómo final de dirección, porque se necesitan 800x480 posiciones de memoria, para almacenar los 800x480 datos **RGB** de los pixeles de una imagen.

- **(.WR1_LENGTH(9'h100)** y **.WR2_LENGTH(9'h100)),** que envían un parámetro que utiliza internamente el controlador de la **SDRAM**, para el almacenamiento de datos en la memoria.

- **.WRX_LOAD(!DLY_RST_0),** registro que nos reinicia los valores almacenados en la **memoria SDRAM**, mediante la entrada de reseteo **DLY_RST_0,** que en este caso es la misma que la existente en el **ejemplo de la cámara DE2_D5M.**

- **.WRX_CLK(~CCD_PIXCLK),** funciona a modo de clock en la operación de almacenamiento de **datos RGB** en la **SDRAM** que realiza el módulo controlador. Este clock será el mismo que el existente en la **cámara D5M**. La frecuencia de **~CCD_PIXCLK** será de 25 MHz.

Posteriormente los **datos RGB** capturados de la **cámara** y que están almacenados en la **memoria SDRAM**, se deben de enviar a la **pantalla LTM** por medio del módulo **lcd_timing_controller,** para mostrar la imagen en pantalla**.**

- **.RD1_DATA(Read_DATA1)** es el registro de salida que leerá el dato correspondiente a los colores rojo y verde de cada pixel, que están almacenados en la **memoria SDRAM**. **.RD2_DATA(Read_DATA2)** es el registro de salida que leerá el dato correspondiente al color azul de cada pixel almacenado en la **memoria SDRAM**. El formato de los datos leídos es de 8 bits compatibles con la **pantalla LTM**. Estos **datos RGB** llegan al módulo **lcd_timing_controller** a través de estos registros, de manera que gestione los **datos RGB** para visualizar la imagen en la **pantalla LTM.**

- **.RD1(mRead)** y **.RD2(mRead),** reciben una señal desde el módulo **lcd_timing_controller,** para habilitar la lectura de los **datos RGB** almacenados en la **memoria SDRAM** hacia el módulo **lcd_timing_controller**.

- **.RD1_ADDR(0)** y **.RD2_ADDR(22'h100000),** son los registros de entrada donde enviamos las direcciones de comienzo de la lectura de datos**.**

- **.RD1_MAX_ADDR(800*480)** y **.RD2_MAX_ADDR(22'h100000+800*480),** son los registros de entrada que envían la dirección final para la lectura de datos. Se puede observar también que la dirección final es 800*480, la resolución o el número de pixeles de la imagen que se almacenan en la **memoria SDRAM**.

- **.RD1_CLK(ltm_nclk)** y **.RD2_CLK(ltm_nclk),** funcionan a modo de clock en la operación de lectura de datos de la **SDRAM** que realiza el módulo controlador. Este clock será el mismo que el existente en la **pantalla LTM**. La frecuencia de la señal ltm_nclk, será de 25 MHz.

```
MÓDULO : Sdram_Control_4Port
port map(REF_CLK => CLOCK_50,
                    RESET_N => '1',
                    CLK => sdram_ctrl_clk,
                    WR1_DATA => sCCD_R(11 downto 4) & sCCD_G(11 downto 4),
                    WR1 => sCCD_DVAL,
                    WR1_ADDR =>          "00000000000000000000000",
                    WR1_MAX_ADDR =>    "00001011101110000000000", -- 800*480
                    WR1_LENGTH => "100000000",
                    WR1_LOAD => not DLY_RST_0,
                    WR1_CLK => not CCD_PIXCLK,

                    WR2_DATA => "00000000" & sCCD_B(11 downto 4),
                    WR2 => sCCD_DVAL,
                    WR2_ADDR =>          "00100000000000000000000",
                    WR2_MAX_ADDR =>    "00101011101110000000000", -- " + 800*480
                    WR2_LENGTH => "100000000",
                    WR2_LOAD => not DLY_RST_0,
                    WR2_CLK => not CCD_PIXCLK,

                    RD1 => mRead,
                    RD1_ADDR =>          "00000000000000000000000",
                    RD1_MAX_ADDR =>    "00001011101110000000000", -- 800*480
                    RD1_LENGTH => "100000000" ,
                    RD1_LOAD => not DLY_RST_0,
                    RD1_CLK => Itm_nclk,
                    RD1_DATA => Read_DATA1,

                    RD2 => mRead,
                    RD2_ADDR =>          "00100000000000000000000",
                    RD2_MAX_ADDR =>    "00101011101110000000000", -- " + 800*480
                    RD2_LENGTH => "100000000" ,
                    RD2_LOAD => not DLY_RST_0,
                    RD2_CLK => Itm_nclk,
                    RD2_DATA => Read_DATA2,

                    SA => DRAM_ADDR,
                    BA => ba,
                    CS_N => dram_cs,
                    CKE => DRAM_CKE,
                    RAS_N => DRAM_RAS_N,
                    CAS_N => DRAM_CAS_N,
                    WE_N => DRAM_WE_N,
                    DQ => DRAM_DQ,
                    DQM => dram_dqm,
                    SDR_CLK => DRAM_CLK
                    );
```

Figura 17: Módulo final Sdram_Control_4Port

5.3.3. Modificaciones en el módulo I2C_CCD_Config

Este módulo inicializa la configuración de los registros internos de la **cámara D5M.** A continuación, se comentará que registros hay que tener en cuenta para el control interno de la cámara.
Es en el registro **LUT_DATA** del módulo donde se asignan las configuraciones.
se configuran unos registros relacionados con la resolución de la imagen capturada de la siguiente manera:

```
**********************************************************
assign sensor_start_row = iZOOM_MODE_SW ?  24'h010036 :
24'h010036;
assign sensor_start_column = iZOOM_MODE_SW ?  24'h020010 :
24'h020010;
```

Se sabe que se necesita capturar de la cámara una imagen de una resolución de 800x480, que es la resolución de imagen correspondiente a la **pantalla LTM**. Con ello cabe pensar que la resolución de la imagen que captura de la cámara será de 800x480 (altura y anchura de la imagen). La resolución de esta imagen se puede averiguar al aplicar las siguientes fórmulas:

Para el ancho de la imagen;

$$W = 2xceil\left(\frac{Column_{Size} + 1}{2x(Column_{Skip} + 1)}\right)$$

Para la altura de la imagen;

$$H = 2xceil\left(\frac{Row_{Size} + 1}{2x(Row_{Skip} + 1)}\right)$$

En estas fórmulas aparecen los registros internos a los que se le asignan los parámetros, con el fin de conseguir la resolución adecuada.

Después de varias pruebas que se explicarán a lo largo del siguiente apartado y modificando las variables de las anteriores fórmulas, llegamos a la conclusión de utilizar estos datos como parámetros finales:
Se utilizará el modo Skip 2x, en el que se leen 1 par de pixeles por cada 2 pares de pixeles, es decir salta un par de pixeles por cada par de pixeles.
Con lo que para conseguir una resolución de 800x481 (es decir W=800, H=481), se necesitará asignar los siguientes valores: Column_Size=1599, Row_Size=961.

Estos valores traducidos a hexadecimal son los siguientes: Column_Size=63F, Row_Size=3C1.

Con todo, la inicialización de los registros para la resolución de la imagen capturada de la cámara quedará de la siguiente manera:

```
**********************************************************
assign sensor_row_size   = iZOOM_MODE_SW ?  24'h0303C1 :
24'h0303C1;
assign sensor_column_size = iZOOM_MODE_SW ?  24'h04063F  :
24'h04063F;
assign sensor_row_mode = iZOOM_MODE_SW ?  24'h220011  :
24'h220011;
assign sensor_column_mode = iZOOM_MODE_SW ?  24'h230011  :
24'h230011;
**********************************************************
```

6

Fase de Pruebas

En esta fase se explicarán todas las pruebas que se han realizado para el correcto funcionamiento de la aplicación. Pruebas tales como la correcta visualización de las imágenes, el correcto cambio de luminosidad de la imagen, la resolución de pantalla, etc.
Se ha mencionado en fases previas que un posible error en cualquiera de las pruebas puede hacer que se modifiquen pasos anteriores por lo que los resultados finales del proyecto podrán verse alterados dependiendo de los resultados obtenidos.
Una vez desarrollada la estructura del diseño, para poner en marcha el sistema tendremos que compilarlo y cargar el proyecto en la FPGA con las herramientas correspondientes del **Quartus**.
A continuación, tendremos que reiniciar los diferentes parámetros del Sistema, pulsando **KEY [0].** Posteriormente para empezar a ver la imagen en la pantalla, se pone en marcha el Sistema al pulsar **KEY [3].**
Al iniciar la trasmisión de las imágenes a la pantalla se pudo observar que la imagen que capturaba la cámara no se veía nítida por lo que habría que hacer una serie de modificaciones en la sincroniza del sistema.
Ambos dispositivos tienen que tener la misma frecuencia de reloj, en la captura de datos, en el almacenamiento en la **SDRAM**, en la lectura de datos de la **SDRAM**, y en la recepción de datos en la **pantalla LTM**.
Otras de las modificaciones que se han realizado es la de la resolución de la imagen capturada para la correcta visualización en la pantalla.
Para obtener la resolución deseada, se ha tenido que modificar algunos parámetros en la cabecera del módulo **Sdram_Control_4Port** y en su interior.
A continuación, se explicará de una forma más detallada los cambios realizados a lo largo de esta fase:

6.1. Pruebas relacionadas con el sincronismo de la imagen

Esta es una de las pruebas más importantes a la hora de poder lograr visualizar la imagen que capturaba la cámara en la **pantalla LCD.**
Se hicieron varias pruebas modificando las señales de reloj a los diferentes dispositivos y al sistema que los controla.
Vimos que la **cámara D5M** funcionaba a una frecuencia de reloj de 25 MHz en la captura de datos en el Sistema **ejemplo DE2_D5M**, que es la misma frecuencia con la que se sincroniza la escritura de datos en la **memoria SDRAM** a través del módulo **Sdram_Control_4Port.**
Sin embargo, en el Sistema ejemplo **DE2_LTM_Ephoto**, la frecuencia de reloj en la lectura de datos de la **SDRAM** y su envío a **la pantalla LTM** es de 33 MHz.
Manteniendo estas frecuencias de reloj para cada dispositivo se observó una desincronización en la captura y almacenamiento de datos en la SDRAM respecto a la lectura y muestra de los datos en la **pantalla LCD.**
Se llegó a la conclusión de que no era viable mantener las dos frecuencias de reloj ya que no conseguíamos visualizar la imagen y se tomó la decisión de que todo el sistema tuviese la misma frecuencia de reloj.
Al ver que la muestra de las imágenes en pantalla era el paso más complicado ya que la resolución de 800x480 viene dada por el sistema ejemplo **DE2_LTM_Ephoto**, se decidió mantener la frecuencia de 33MHz para todo el sistema, igualando la entrada **XCLKIN** a **ltm_nclk**, que a su vez rescata la frecuencia 33 MHz.
Al realizar la prueba vimos que tampoco funcionaba por lo que la última opción que nos quedaba era sincronizar todo el sistema con la frecuencia de 25 MHz.
De esta forma la cámara mantenía su clock de 25 MHz, y al registro de entrada **ltm_nclk** se le igualaba la misma frecuencia de clock. A esta frecuencia se le cambia el signo, porque está configurado internamente, para que la **pantalla LTM** capture datos del exterior en el flanco de bajada de nclk.
Una vez realizadas las modificaciones necesarias se puso en marcha para ver su funcionamiento. Al ver las imágenes en la pantalla de forma correcta se dio por finalizada esta fase de pruebas.

6.2. Pruebas relacionadas con el módulo controlador SDRAM

Muchos de los parámetros del módulo Sdram_Control_4Port han tenido que ser cambiados tal y como hemos visto y aparecen explicados en la fase anterior de modificaciones realizadas acerca de este módulo.
Sin embargo, varios registros internos de este módulo se han tenido que ir modificando para obtener una imagen correcta en la pantalla.
Valores que se han modificado referentes a las direcciones finales del almacenamiento de datos.
En el sistema ejemplo de la pantalla se envía como dirección máxima para el almacenamiento de datos 640*480 y en el proyecto **DE2_D5M**, se envía como dirección máxima para el almacenamiento de datos 800*480.
Al probar los dos casos añadiendo dichos valores en los registros de almacenamiento tanto para la escritura de datos **WRX_MAX_ADDR(),** como para la lectura de datos **RDX_MAX_ADDR(),** se observó que con el valor de 800*480, se veían las imágenes en **pantalla LTM** con bastante mejor nitidez.
Otro de los registros internos que tenía dos posibilidades de funcionamiento era el asignado a los registros. WRX**_LENGTH(9'h100),**
.RDX_LENGTH(9'h100), que es un parámetro que utiliza el controlador internamente para el almacenamiento de datos. En el módulo del Sistema ejemplo de la pantalla, se envía en esta entrada de registro el dato en hexadecimal 80, y en el Sistema ejemplo de la cámara, utilizaba el dato 100.
Al probar ambos casos se decidió utilizar el valor 100 para ambos registros ya que con el otro valor las imágenes aparecían divididas en cuadros.

6.3. Pruebas relacionadas con la resolución de la imagen

En cuanto a la resolución de la imagen que captura la cámara, hay que recordar que existen las siguientes fórmulas:

Para el ancho de la imagen;

$$W = 2xceil\left(\frac{Column_{Size} + 1}{2x(Column_{Skip} + 1)}\right)$$

Para la altura de la imagen;

$$H = 2xceil\left(\frac{Row_{Size} + 1}{2x(Row_{Skip} + 1)}\right)$$

Aplicando dichas fórmulas vemos que con un Row Skip=0 y Column Skip=0, y una resolución de 800x480, es decir W=800 y H=480. Tenemos Column Size=799 en decimal, Column Size=31F en hexadecimal, y Row Size=479 en decimal, Row Size=1DF en hexadecimal. También hay que mencionar que Column Bin=0 y Row Bin=0. Aplicando dichos valores a cada uno de los registros correspondientes, las líneas de programa en el módulo **I2C_CCD_Config** quedarían de la siguiente manera:

```
//////////////////////////////////////////////////////
assign sensor_row_size          = iZOOM_MODE_SW ?  24'h0301DF :
24'h0301DF;
assign sensor_column_size       = iZOOM_MODE_SW ?  24'h04031F  :
24'h04031F;
assign sensor_row_mode          = iZOOM_MODE_SW ?  24'h220000  :
24'h220000;
assign sensor_column_mode       = iZOOM_MODE_SW ?  24'h230000  :
24'h230000;
//////////////////////////////////////////////////////
```

Al ejecutar el sistema con estos parámetros asignados se pudo observar la imagen en pantalla, pero con una franja recorriendo la pantalla de arriba abajo. Alguno de los parámetros introducidos no eran los adecuados por lo que se decidió probar con otros valores.

Se probó por mejorar la nitidez de la imagen aumentando los parámetros Colum Bin=1 y Row Bin=1. Con estos parámetros, los valores legales para el modo Skip serían Row Skip=1 y Column Skip=1, con ellos se podría conseguir la misma resolución 800x480. En ese caso para esta resolución (W=800 y H=480), aplicando la fórmula anterior, tenemos Column Size=1599 en decimal, Column Size=63F en hexadecimal, y Row Size=959 en decimal, Row Size=3BF en hexadecimal.

```
**********************************************************
assign sensor_row_size          = iZOOM_MODE_SW ?  24'h0303BF :
24'h0303BF;
assign sensor_column_size       = iZOOM_MODE_SW ?  24'h04063F  :
24'h04063F;
assign sensor_row_mode          = iZOOM_MODE_SW ?  24'h220011  :
24'h220011;
assign sensor_column_mode       = iZOOM_MODE_SW ?  24'h230011  :
24'h230011;
**********************************************************
```

Al realizar la siguiente prueba se observó nuevamente la franja recorriendo la pantalla por lo que teníamos un problema en el parámetro de la altura de la pantalla. Por otro lado, conseguimos mejorar la nitidez de la imagen seleccionando el modo Row Skip=1 y Column Skip=1.

Manteniendo la nitidez de la imagen con los valores de modo de lectura anteriores (con Row Bin=1 y Column Bin=1), se solucina el problema que sucede con la franja, aumentando la altura de la resolución a H=481, y manteniendo el ancho de la pantalla a W=800. Con estos valores, aplicando la fórmula tenemos Column Size=1599 en decimal, Column Size=63F en hexadecimal, y Row Size=961 en decimal, Row Size=3C1 en hexadecimal.

```
*****************************************************
assign sensor_row_size          = iZOOM_MODE_SW ?  24'h0303C1 :
24'h0303C1;
assign sensor_column_size       = iZOOM_MODE_SW ?  24'h04063F  :
24'h04063F;
assign sensor_row_mode          = iZOOM_MODE_SW ?  24'h220011  :
24'h220011;
assign sensor_column_mode       = iZOOM_MODE_SW ?  24'h230011  :
24'h230011;
*****************************************************
```

Al ejecutar el sistema conseguimos eliminar la franja y conseguir visualizar la imagen de forma correcta.
Por lo tanto, los parámetros finales para capturar una imagen a una resolución acorde a la **pantalla LTM** son:

- Resolución 481x800
- Modo de lectura, Row Bin=1 y Column Bin=1; Row Skip=1 y Column Skip=1.

6.4. Pruebas relacionadas con la intensidad del color

La intensidad con la que se captura cada color en los pixeles de la imagen, se puede modificar variando los parámetros asignados para una serie de registros internos. Estos registros internos son: RX02B (Registro para calcular la ganancia para el color Verde 1), RX02C (Registro para ganancia del color azul), RX02D (Registro para el color Rojo), RX02E (Registro para el color Verde2).
Estos registros modificarán la intensidad de cada color de cada pixel, variando los valores de los bits de los siguientes parámetros.
Los registros internos relacionados con las ganancias de cada color que se acaban de mencionar, se pueden encontrar en el interior del módulo **I2C_CCD_Config**.

Los valores más adecuados para cada registro, correspondiente a cada color son:

```
6:  LUT_DATA  <=  24'h2B003F;    //      Green 1 Gain
7:  LUT_DATA  <=  24'h2C003F;    //      Blue Gain
8:  LUT_DATA  <=  24'h2D04BF;    //      Red Gain
9:  LUT_DATA  <=  24'h2E003F;    //      Green 2 Gain
```

Estos datos se han sacado de la siguiente tabla:

		GRADO INTENSIDAD EN CADA COLOR Y EN LA LUMINOSIDAD			
GAIN	**PARAMETRO ASIGNADO**	**BLUE COLOR**	**GREEN COLOR**	**RED COLOR**	**LUMINOSIDAD**
GREEN G.	81F	Regular	Bastante	Bastante	Normal
	31F	Regular	Normal	Bastante	Normal
	10	Regular	Normal	Bastante	Normal
BLUE G.	0DBF	Excesivo	Nada	Normal	Mucho
	07BF	Bastante	Poco	Normal	Bastante
	03BF	Normal	Regular	Normal	Normal
	003F	**Regular**	**Normal**	**Normal**	**Normal**
RED G.	0DBF	Bastante	Poco	Excesivo	Poco
	08BF	Bastante	Normal	Excesivo	Regular
	04BF	**Normal**	**Normal**	**Normal**	**Normal**
	003F	Bastante	Mucho	Regular	Normal
	10	Excesivo	Poco	Nada	Poco
GREEN G.1 & GREEN G.2	0D3F	Nada	Excesivo	Poco	Mucho
	81F	Bastante	Bastante	Bastante	Bastante
	31F	Regular	Bastante	Bastante	Normal
	003F	**Normal**	**Normal**	**Normal**	**Normal**
	10	Bastante	Nada	Excesivo	Poco

Figura 18: Tabla-relación intensidad en la luminosidad

Hay que resaltar que la luminosidad de la imagen se puede variar, pulsando de forma táctil la pantalla. Extremo superior derecha de la pantalla para aumentar la luminosidad y extremo inferior derecha para bajar la luminosidad.

7

Funcionamiento y conclusiones del Diseño

7.1. Funcionamiento del Programa

El Sistema que se acaba de desarrollar tiene la función de capturar las imágenes que la **cámara TRDB_D5M** recoge, y mostrarlas en la pantalla **TRDB_LTM** en tiempo real, a través de la **placa DE2**, a la cual están conectados por medio de dos puertos expansores de 40 pines.
Antes de poner en marcha el Sistema hay que reiniciarlo para inicializar los registros de cada dispositivo de forma adecuada tal y como lo hemos visto a lo largo de la memoria. Esto se consigue en el momento que se pulse el botón **KEY [0],** en ese momento el módulo **Reset_Delay** envía señales a través de sus registros de salida a los diferentes módulos para que se reinicien los dispositivos.

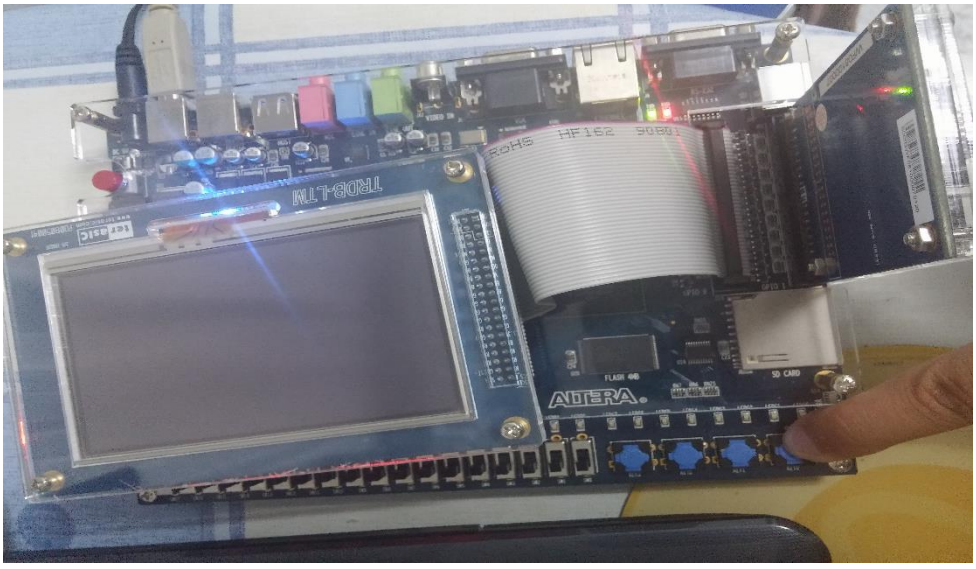


Figura 19: Imagen reiniciando el sistema

Tenemos dos módulos que reinician cada uno de los dispositivos. El encargado de enviar las configuraciones de los registros internos de la **pantalla LTM** es el módulo **(lcd_spi_cotroller)**, que hace que controle el funcionamiento interno del dispositivo y el flujo de datos que se envían a la pantalla para que muestre la imagen en ella. También establece la resolución de la imagen, así como el tipo de flanco en el sincronismo en la recepción de la imagen.
El otro módulo encargado de la inicialización de la **cámara D5M** en el momento que se produce el reseteo es **I2C_CCD_Config**, este módulo envía las configuraciones de los registros internos de dicha cámara para el control en la captura de las imágenes recogidas. Entre estos registros, está el que establece la resolución de la imagen capturada de la cámara, los que hacen referencia a la intensidad en el color de los pixeles, o el que adopta el sincronismo del reloj de salida.
Después de configurar los registros de cada uno de los dispositivos es necesario pulsar el botón **KEY [3]** para que comience a funcionar. En ese momento el módulo **CCD_Capture** se encarga de recoger los datos de la señal de los pixeles de la imagen que captura la cámara. Estos datos salen a través del registro de salida **mCCD_DATA,** para llegar al módulo **RAW2RGB**, para que este módulo me procese cada dato recogido en 3 **datos RGB** por cada pixel capturado de la imagen. También este módulo enviará la señal al módulo **Sdram_Control_4Port**, para habilitar la escritura de datos en la **SDRAM.**

Figura 20: Imagen iniciando la transmisión de imágenes

Los **datos RGB** de cada pixel se almacenan en la **memoria SDRAM** para una posterior lectura para el envío de dichos datos almacenados al módulo que controla la recepción de datos al LTM **(lcd_timing_controller)**.
El módulo **lcd_timing_controller**, envía la señal al módulo **Sdram_Control_4Port**, para habilitar la lectura de datos de la **SDRAM**. Una vez recibe estos **datos RGB**, gestiona su envío a la **pantalla LTM**, para visualizar las imágenes en ella.
Los pixeles que recibe la **pantalla LTM** se sincronizan a través de la señal **ltm_nclk**, que será de 25 MHz. Esta frecuencia también será igual que la frecuencia de la señal de reloj de la **cámara D5M (PIXCLK),** que va a ser la que sincronice el control de la escritura de datos en la **SDRAM** a través del módulo **Sdram_Control_4Port.**
Por último, destacar dos detalles más que están implementados en la aplicación.
El primer detalle es que cuando se pulsa **KEY [2],** se congela la imagen en la pantalla. En ese momento deja de capturar datos de los pixeles y ordenan a la **SDRAM** a través de la señal **mCCD_DVAL** proveniente del módulo **CCD_Capture,** que se envíe la información a la pantalla LTM de la última imagen guardada.
Otra de las maneras que tenemos para congelar la imagen es hacerlo desde la pantalla táctil tocando el extremo inferior izquierdo de la pantalla tal y como hemos visto anteriormente.

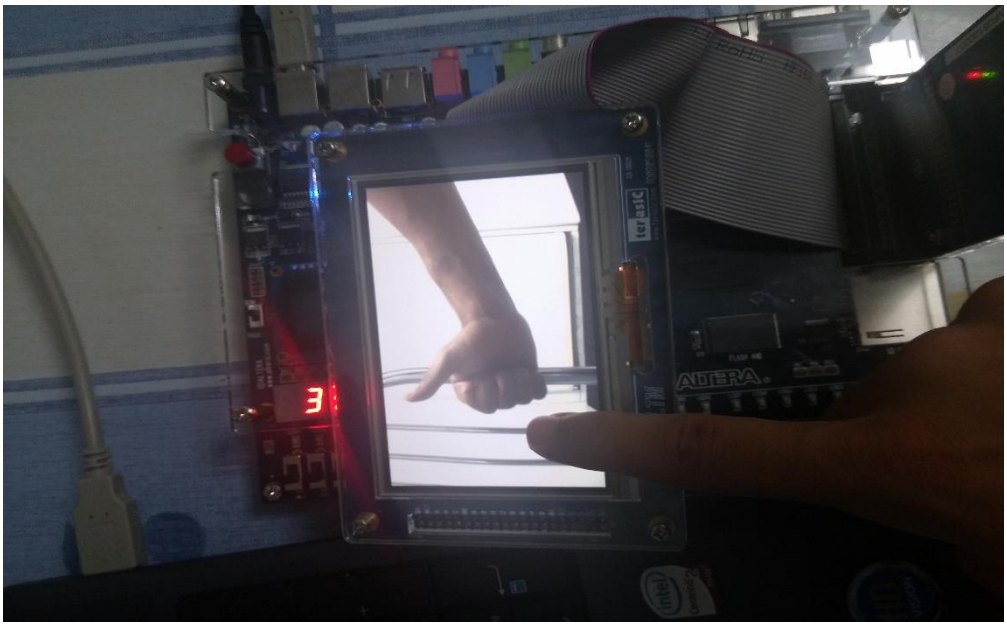


Figura 21: Imagen congelando la transmisión de datos

El segundo detalle es la introducción de la posibilidad de variar la luminosidad para la imagen capturada.
Podremos aumentar la luminosidad de la imagen tocando la parte superior derecha de la pantalla y del mismo modo disminuirla tocando la parte inferior derecha de la pantalla.
Al pulsar una de las dos zonas mencionadas, se variará el parámetro que se le asigna al registro que regula la luminosidad de la imagen capturada (RX09) del módulo **I2C_CCD_Config.**

Figura 22: Imagen bajando la luminosidad

7.2. Conclusiones del proyecto

Crear un sistema en el que la **cámara THDB-D5M** y la **pantalla THDB-LTM** fueran compatibles era el objetivo principal del proyecto por lo que se puede decir que el objetivo fundamental del proyecto se ha cumplido.
Para la realización de este proyecto no ha sido suficiente con lo aprendido en la asignatura de Diseño y Sistemas Digitales ya que se han introducido nuevos lenguajes no vistos en clase como es el caso de **verilog**, ambos diseños ejemplos diseñados en dicho lenguaje. También se ha tenido que profundizar en el control de flujo de señales de una imagen para la iteración con ambos dispositivos.
Se ha tenido que buscar más información acerca de los dispositivos para poder modificar los parámetros, los módulos y la sincronización correspondiente para el correcto desarrollo del Sistema.
Además de todo lo mencionado, en líneas generales se ha estudiado cómo interactúan los dispositivos con cada uno de los Sistemas ejemplo a los que se aplica, su funcionamiento interno en el tratamiento del flujo de **datos RGB** de los pixeles de las imágenes, su modo de inicialización, y los diferentes registros internos que describirán el control de cada uno de los dispositivos.
Para terminar, creo que con todo lo visto en este proyecto, se puede decir que se ha utilizado todo lo aprendido a lo largo de la asignatura e incluso se ha investigado ciertas cosas no vistas anteriormente.

7.3. Conclusiones personales

El desarrollo del proyecto ha sido bastante interesante ya que, se ha tenido que aprender cómo funcionan dos dispositivos que, aunque sea a menor escala son dos dispositivos que hoy en día se utilizan en todo el mundo como puede ser la utilización de una simple cámara de móvil como una GOPRO.
Se ha realizado un Sistema en tiempo real a partir de dos sistemas aplicables a dos dispositivos (**cámara D5M** y **pantalla LTM**), con el fin de que interaccionen entre ellos.
Otro de los aspectos más interesantes y gratificantes ha sido a la hora de hacer las pruebas de la puesta en marcha del Sistema ya que han ido surgiendo problemas que con las modificaciones que se han comentado a lo

largo del proyecto se podía observar cómo se conseguía mostrar las imágenes capturadas por la **cámara** en la **pantalla LCD.**
Por último, tengo que decir que la realización de este proyecto puede ser muy amplio ya que hay muchas variables y aspectos que se pueden introducir por lo que, dada su envergadura se ha decidido crear un sistema base en **VHDL** para que en un futuro si alguna persona quiere, pueda continuar y ampliar este proyecto. Algunas de las posibilidades a introducir pueden ser:

- Cuando se pulsa **KEY [2]** congelamos la imagen por lo que se podría hacer que dicha imagen se guardara en una memoria flash para después poder disponer de ella.

- Se podría introducir un temporizador que pasado ese tiempo la cámara congelase la foto.

- Se podría introducir un cronometro para observar durante cuánto tiempo está grabando, es decir, desde que se pulsa **KEY [3]** hasta que congelamos la imagen.

Estas son algunas de las posibles ampliaciones que se podrían introducir por lo que dejo este sistema a disposición de la persona que le guste este tipo de proyectos y quiera ampliarlo.

8

Bibliografía

8.1. Bibliografía

Como principal documentación para la realizar el proyecto han sido unos manuales que venían en formato PDF, de los dispositivos utilizados:

-Cámara TRDB-D5M

Documentos en PDF, **TRDB_D5M_UserGuide** y **THDB-D5M_Hardware specification.**

-Pantalla TRDB-LTM)

Documentos en PDF, **DE2_70_Application_note_02** y **TRDB_LTM_UserGuide_v1.23.**

-Placa DE2.

El documento en PDF, **DE2_UserManuall.**

-**aprendizaje del lenguaje verilog.**

Se ha utilizado internet para obtener información, aprender e interpretar el lenguaje **verilog**.

-Herramientas de diseño de sistemas digitales.

Se ha utilizado internet para obtener información sobre **Quartus**.