UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERIA ELECTRICA

"DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE UNIFICACIÓN DE SEÑALES DE VIDEO"

MEMORIA PARA OPTAR AL TITULO DE INGENIERIO CIVIL ELECTRICISTA

NELSON JAVIER CASTILLO DURAN

PROFESOR GUIA:
MAURICIO BAHAMONDE BARROS

MIEMBROS DE LA COMISION:
HECTOR AGUSTO ALEGRIA
NICOLAS BELTRAN MATURANA

SANTIAGO DE CHILE
ENERO 2010

RESUMEN DE LA MEMORIA
PARA OPTAR AL TITULO DE
INGENIERO CIVIL ELECTRICISTA
POR: NELSON JAVIER CASTILLO DURÁN
FECHA: 23/03/2010
PROF. GUIA: Sr. MAURICIO BAHAMONDE

"DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE UNIFICACIÓN DE SEÑALES DE VIDEO"

Este proyecto nace de una propuesta del profesor guía de esta memoria, que plantea la realización de un *multiviewer,* dispositivo capaz de mostrar varias entradas de video a través de una sola pantalla, para ser utilizado en televisoras y compañías de televisión por cable para la administración, verificación y edición de la señales de transmisión de TV. Además el mercado de este producto es mucho mayor teniendo un gran uso en sistemas de seguridad, monitoreo de transito, eventos y posibles usos en comercialización de dispositivos de video.

El objetivo general del presente trabajo consiste en diseñar e implementar el *software* y *hardware* de un prototipo básico del sistema *multiviewer,* que permita sentar las bases de un futuro producto comercial.

Este trabajo parte con la recopilación de información, investigando características de *multiviewers* comerciales de las cuales se elijen para implementar: el soporte de múltiples estándares, el control de tamaño y posición de sub-imágenes. La implementación del dispositivo requiere indagar en tecnologías de procesamiento lo que lleva a comparar las tecnologías *Digital Signal Processor* (DSP) y *Field Programmable Gate Array* (FPGA), llegando a la conclusión de la superioridad de la segunda por sus atributos de procesamiento paralelo. Luego de esto se define el diseño del dispositivo asignándole las funciones de control a la FPGA y el manejo de estándares a dispositivos externos. Las funciones de control son diseñadas como manejo de memoria para el control de posición, como interpolación y filtrado de señales en dos dimensiones para el control de tamaño, esto es implementado a través de *Verilog* un Lenguaje de Descripción de *Hardware* (HDL, *Hardware Description Language*) que permite la configuración de una FPGA. Por otra parte, el *hardware* del sistema se diseña para utilizar la plataforma de desarrollo *Spartan-3A Starter Kit*, con la cual se reduce el tiempo de desarrollo, y se limita el diseño a una placa de circuito impreso (PCB, *Printed Circuit Board*) con conexión a la plataforma de desarrollo. La PCB diseñada contiene los chips encargados de los estándares y los conectores necesarios para capturar las entradas y emitir la salida de video.

El objetivo de diseñar e implementar el *software* del sistema fue completado, obteniendo la codificación en HDL de un sistema de dos entradas y una salida de video en formato YC_bC_r con muestreo 4:2:2 acorde a la recomendación UIT-R BT.601 de la Unión Internacional de Telecomunicaciones (UIT) y con una interfaz acorde a la recomendación UIT-R BT.656. La salida de video contiene la imagen de un máximo de dos sub-pantallas de tamaño y posición seleccionable por el configurador, las cuales pueden contener una de las entradas del sistema.

El segundo objetivo de esta memoria se alcanzó parcialmente, logrando obtener los planos de la placa de circuito impreso, pero no así su construcción, la cual se plantea como un trabajo futuro.

El sistema diseñado permite la proyección de dos posibles sistemas comerciales a futuro, uno de bajo costo y prestaciones básicas para usos domiciliarios en seguridad y otro profesional para tareas de mayor envergadura que necesiten más entradas de video por monitor, una mejor calidad de imagen y mayor control sobre ellas.

Agradecimientos

Deseo agradecer a mis padres Heriberto y Alicia por brindarme los valores, la educación y el cariño que son el pilar de lo que soy el hoy en día, a mi hermana por darme su cariño y compañía, a mi gran amigo Mathias el que ha sido una gran influencia en mi vida y a los compañeros y amigos que he encontrado durante mi paso en la universidad, en especial a Daniela, Francisco, Oscar y Romina, a quienes estimo y respeto mucho.

Agradezco al profesor Mauricio Bahamonde por brindarme la oportunidad de trabajar en un tema interesante y novedoso, que ha significado un gran reto profesional y que me ha ayudado a perder el miedo al emprendimiento, en el cual espero desarrollar mi futuro profesional.

INDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1 INTRODUCCIÓN

El presente documento relata el trabajo realizado por el alumno de la Universidad de Chile Nelson J. Castillo Duran para la obtención del título de Ingeniero Civil Electricista.

El proyecto realizado corresponde al proceso de diseño y construcción del dispositivo electrónico llamado *multiviewer*, aparato que permite la visualización de múltiples señales de video por una única pantalla.

1.1 Motivación

Este proyecto nace de una propuesta del profesor guía de esta memoria, que plantea la realización de este sistema como un futuro producto comercial para ser utilizado en televisoras y compañías de televisión por cable para la administración, verificación y edición de señales de transmisión de TV. Además el mercado de este producto es mucho mayor teniendo un gran uso en sistemas de seguridad, monitoreo de transito, evento y posibles usos en comercialización de dispositivos de video.

La principal motivación de este proyecto es la visión a futuro de comercializar un producto tecnológico con un gran mercado objetivo y que en la actualidad es usufructo de pocas compañías.

1.2 Objetivos Generales

Este trabajo corresponde al primer diseño de un *multiviewer* en Chile, por lo que el objetivo principal de esta memoria es desarrollar un sistema base que sirva de inicio para la posterior fabricación comercial de este dispositivo.

El diseño con lenguaje de descripción de hardware (*Hardware Description Language, HDL)*, ha permitido a la industria de la electrónica disminuir los tiempo de desarrollo y aumentar la complejidad de los dispositivos diseñados, por lo que esta memoria tiene como objetivo secundario ganar experiencia en el diseño con HDL.

1.3 Objetivos Específicos

Los objetivos específicos del diseño son los siguientes:

- Investigar y determinar la mejor tecnología para la implementación de un *multiviewer*.
- Desarrollar todas las etapas del diseño en HDL, para una arreglo de compuertas programables por campo (*Field Programmable Gate* Array, FPGA).
- Diseñar una plataforma de prueba de bajo costo, que permita verificar el funcionamiento de diseño en HDL.
- Construcción de la plataforma de prueba.
- Realizar pruebas con señales de video reales a la plataforma de pruebas.

1.4 Metodología

La metodología usada para la realización del proyecto se inicia con la adquisición de información sobre temas como las características de *multiviewers* comerciales, conceptos básicos de video y dispositivos que permitan procesamiento de señales. Esta etapa permite determinar que un *multiviewer* básico debe realizar la función de adquirir múltiples señales de video, las cuales son procesadas para tener una única codificación, lo que simplifica el procesamiento posterior correspondiente al redimensionamiento de cada imagen de video y su posterior posicionamiento en la salida de video que contiene una selección de las múltiples entradas de video, permitiendo mostrar una o varias de estas por un mismo dispositivo de salida de video, pantalla o televisor. Además se hace una comparativa entre los dispositivos FPGA y Procesador Digital de Señales (*Digital Signal Processor*, DSP) donde se ve la superioridad del primero para este tipo de sistema (ver Capitulo 3.2.5).

La elección de la tecnología FPGA conlleva la selección del fabricante de chips, entre los cuales se encuentran Xilinx y Altera. El primero de estos ha sido utilizado con anterioridad, por lo que se tiene un mayor manejo de las herramientas computacionales de esta compañía. Los programas usados de Xilinx para el diseño en FPGA son ISE (*Integrated Software Enviroment*) y EDK (*Embedded Development Kit*). Además de estos programas se para las simulaciones de los diseños fue utilizado el programa ModelSim de la empresa Mentor Graphics.

El siguiente paso usado corresponde al diseño del *Multiviewer*, el cual se separa en dos etapas que fueron realizadas en paralelo y que el término de ellas permitiría la construcción del sistema, estas corresponden al diseño de la plataforma de pruebas y al diseño en FPGA.

1.4.1 Diseño de plataforma de prueba

Esta etapa tiene por objetivo diseñar una placa de circuito impreso (Printed Circuit Board, PCB) de bajo costo, que permita verificar las características básicas de un *Multiviewer* generadas en la etapa de diseño en FPGA y seleccionar y probar dispositivos codificadores y decodificadores de video, para futuros desarrollos. Esta etapa se separa en tres fases secuenciales que se explican a continuación.

1.4.1.1 Investigación de dispositivos codificadores y decodificadores

En esta fase se buscan dispositivos que permitan la decodificación y codificación de señales de video, con soporte de múltiples estándares de transmisión.

1.4.1.2 Diseño de esquemáticos

En esta fase se investigan PCB ya realizados que cumplan funciones similares a las buscadas por la plataforma de pruebas, con lo que se determinan componentes a usar y las interconexiones que deben tener para cumplir las funciones deseadas. Esta fase es realizada con el programa Eagle de la compañía Cadsoft.

1.4.1.3 Diseño de PCB

En esta etapa se realizan diseños de los empaquetados de los dispositivos considerando las características dadas en sus hojas de datos. Estos diseños son usados posteriormente para definir las posiciones donde se ubicaran los dispositivos en la PCB. Luego las conexiones definidas en los esquemáticos del diseño son usados para crear rutas que unen los pines de los dispositivos. Una vez que todas las conexiones entre dispositivos son llevadas a rutas en la PCB, se genera un archivo con la información requerida para la fabricación de la PCB. Esta etapa de diseño es realizada con el programa Eagle.

1.4.2 Diseño en FPGA

Esta fase del diseño tiene múltiples etapas que son realizadas secuencialmente y que en caso de encontrar errores se debe retornar a etapas anteriores. En la Figura 1-1 se muestran las diversas etapas y la secuencia que se debe seguir para completarlas todas. La mayoría de las etapas son realizadas con el software ISE de Xilinx. En las siguientes secciones se da una breve explicación de cada etapa.

1.4.2.1 Codificación en HDL

Esta etapa corresponde a la codificación del *Multiviewer* básico a través de un HDL. El lenguaje elegido para la codificación corresponde al estándar de la *Institute of Electrical and Electronics Engineers* (IEEE) Verilog[1].

1.4.2.2 Simulación de comportamiento

Esta simulación permite verificar la sintaxis y funcionalidad del código HDL. A través de código HDL que contiene definiciones de señales de pruebas. Esta etapa se realiza con el programa ModelSim.

1.4.2.3 Síntesis de código HDL

Esta etapa es realizada completamente en ISE, el cual verifica la correcta sintaxis del código HDL y optimiza el código generando niveles de registros de transferencia (*Register Transfer Level, RTL*).

1.4.2.4 Asignación de restricciones al diseño

Esta etapa permite controlar las características requeridas del diseño a través de un archivo distinto de los que contienen código HDL, en el cual se indican: frecuencias de relojes, cronometraje de entradas y salida, asignación de pines y estándares, cronometraje de trayectorias y ubicación de componentes.

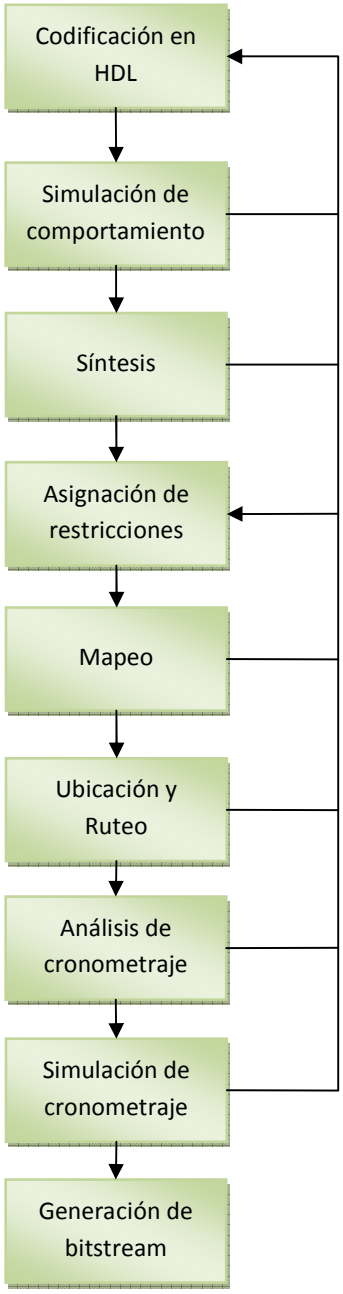


Figura 1-1: Etapas de diseño en FPGA.

1.4.2.5 Mapeo

Esta etapa es realizada por ISE y corresponde a la determinación de los elementos de la FPGA para la implementación del código HDL. Esta etapa puede fallar por insuficiencia de recursos en la FPGA lo que obliga a rediseñar el código HDL para optimizar la utilización de recursos o a modificar las restricciones de diseño.

1.4.2.6 Ubicación y ruteo

Esta etapa es realizada completamente en ISE y corresponde a la asignación de posiciones a los elementos determinados en la etapa anterior y la posterior generación de rutas que permitan la interconexión entre elementos. Esta etapa puede fallar al no poder generarse todas las rutas necesarias para interconectar todos los elementos del diseño, esto obliga al rediseño del código HDL o a la modificación de las restricciones del diseño.

1.4.2.7 Análisis de cronometraje

Este análisis es realizado en ISE a las trayectorias que tienen asignadas restricciones, calculando los retrasos de las trayectorias entre dispositivos *flipflops,* generados por la utilización de compuertas logicas y las rutas asignadas. Esta etapa puede fallar al no cumplirse las restricciones asignadas al diseño, esto obliga al rediseño del código HDL o a la modificación de las restricciones del diseño.

1.4.2.8 Simulación del cronometraje

Esta simulación utiliza un modelo del sistema que considera los tiempos de propagación de las rutas y el tiempo de ejecución de los elementos utilizados en la implementación final, con lo que se pueden obtener la mejor representación del sistema final. En esta simulación se verifica el cumplimiento de los tiempos de seteo y sostenimiento de los elementos del sistema.

1.4.2.9 Generación del bitstream

Esta etapa es realizada en ISE y corresponde a la generación de los archivos binarios que serán montados en la FPGA.

2 REVISIÓN BIBLIOGRÁFICA

2.1 Introducción a Video[1]

En la visión humana, el mundo tridimensional es proyectado por el cristalino del ojo en la retina, la cual está poblada por células foto receptoras que responden a la luz con una longitud de onda entre los 400nm y 700nm. En video, las cámaras son hechas teniendo un lente y un dispositivo foto sensitivo, para imitar como el mundo es percibido por la visión. A pesar que la forma de la retina es aproximadamente la sección de una esfera, es topológicamente bidimensional. En las cámaras, por razones prácticas son usadas imágenes planas, en vez de la sección de una esfera.

Las señales capturadas desde el mundo físico son traducidas a formato digital a través de la digitalización, la cual involucra los procesos de muestreo y cuantización, los que se pueden contemplar en la Figura 2-1.

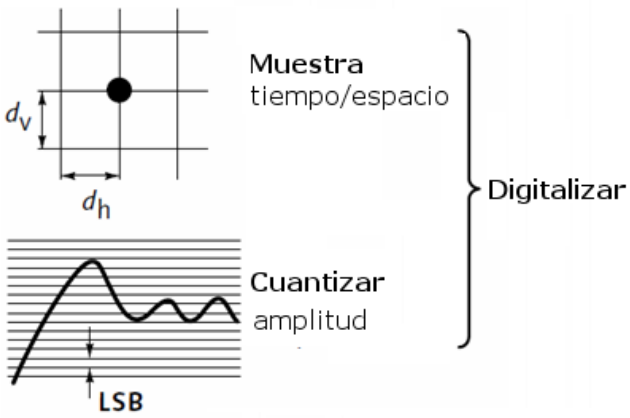


Figura 2-1: Digitalización.

Una imagen digital es representada por un arreglo rectangular de pixeles. En los sistemas de escala de grises, cada pixel compone un único componente cuyo valor está relacionado a lo que sin mucho rigor se le llama luminosidad. En los sistemas de color, cada pixel está compuesto por varios componentes, usualmente tres, cuyos valores están muy relacionados a la percepción del color por los humanos.

[1] Traducción de extractos del libro *Digital Video and HDTV Algorithms and Interfaces*[2].

Una cámara de video, tiene en el plano de imagen, uno o más sensores CCD (*Charge-coupled device*), cada uno formado por cientos de miles células fotoeléctricas arregladas en una rejilla (*lattice*). El número total de pixeles en una imagen es simplemente el producto del numero de columnas de la imagen (técnicamente, muestras por línea activa o en ingles *samples per active line*, S_{AL}) y el número de filas de la imagen (líneas activas o en ingles *active lines*, L_A). Los arreglos de pixeles de diversos estándares de imagen son esquematizados en la Figura 2-2.

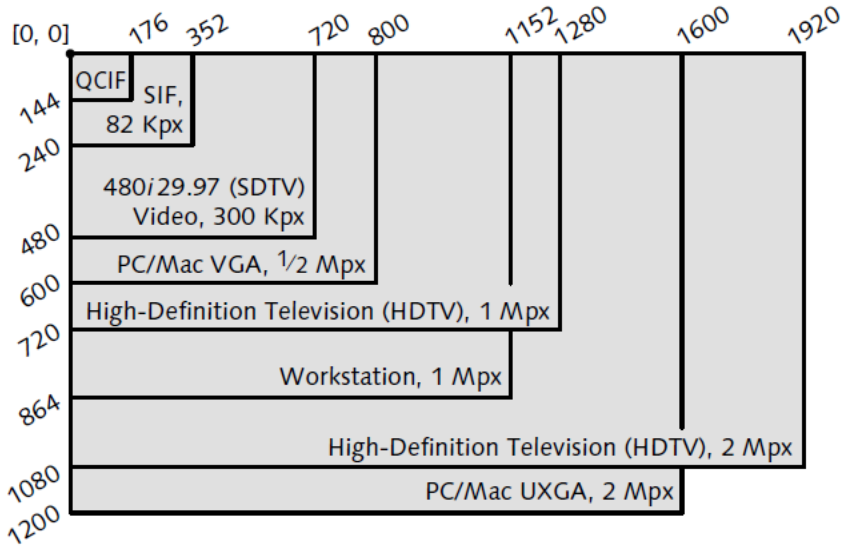


Figura 2-2: Arreglos de pixel para diversos estándares de imagen.

En video, el muestreo del arreglo de pixeles es uniformemente secuenciado en el tiempo para formar escaneo de líneas (*scan lines*), los cuales son a su vez secuenciados en el tiempo durante todo el intervalo del cuadro (*frame)*.

En video análogo, la información del plano de imagen es escaneado de izquierda a derecha a una tasa uniforme durante un intervalo de tiempo fijo. El escaneo establece una relación fija entre la posición en la imagen y el tiempo e instante de la señal. Líneas sucesivas son escaneadas a una tasa uniforme desde la parte superior a la parte inferior de la imagen, por lo que también existirá una relación fija entre la posición vertical y el tiempo.

El patrón estacionario de las líneas escaneadas en paralelo dispuestas a través de la imagen es el denominado *raster*. En video digital las muestras de la matriz de la imagen son transportadas en el mismo orden que es transportada la información video análogo.

En las cámaras y pantallas, un cierto intervalo de tiempo es usado en avanzar la operación de escaneo de una línea a la siguiente. Varios tiempos de líneas son consumidos en pasar de la última línea a la primera línea del siguiente cuadro. El bombardeo de electrones en un CRT (*Cathode Ray Tube*) debe ser apagado (blanqueado) durante estos intervalos, por lo que ellos son llamados intervalos de blanqueo. El intervalo horizontal de blanqueo ocurre entre el escaneo de líneas; el intervalo vertical de blanqueo (*vertical blanking interval,* VBI) ocurre entre *cuadros* (o campos). En la Figura 2-3 se muestran los intervalos de blanqueo de un video de "525 líneas".

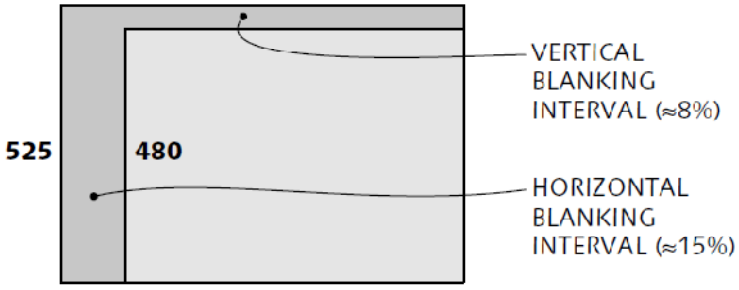


Figura 2-3: Ejemplo de intervalos de blanqueo en video de "525 líneas".

En escaneo progresivo, todas las líneas de la imagen son escaneadas en orden, desde la superior a la inferior. En la Figura 2-4 indica los parámetros básicos de escaneo:

- Líneas totales (*Total Lines*, L_T) comprende todas las líneas escaneadas, esto es, tanto el intervalo vertical de blanqueo y las líneas de la imagen.
- Líneas activas (*Active Lines*, L_A) contienen la imagen.
- Muestras por línea completa (*Samples per total line*, S_{TL}) comprende las muestras en toda una línea, incluyendo el blanqueo horizontal.
- Muestras por línea activa (*Samples per active line*, S_{AL}) cuenta las muestras que tienen permitido tomar valores distintos al nivel de blanqueo.

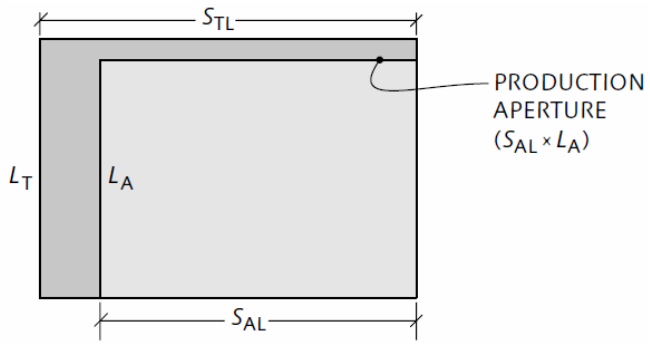


Figura 2-4: Parámetros básicos de escaneo.

La abertura de producción (*Production Aperture*), esquematizada en la Figura 2-4 comprende el arreglo de S_{AL} columnas por L_A filas. Las muestras en la abertura de producción, comprenden el arreglo de pixeles activos. Todos los otros intervalos de muestras que comprenden el blanqueo, están inactivos, aunque ellos pudieran transportar intervalos de información vertical tales como: VITS (*Vertical interval test signal*), VICT (*Vertical interval timecode*), o subtítulos electrónicos (*close captions*).

Aparte del escaneo progresivo existe el escaneo entrelazado, el cual fue diseñado para disminuir la distancia a la que es imperceptible el escaneo de líneas. Este método de escaneo consiste en escanear el alto total de la imagen con un lugar angosto, dejando espacios en la dirección vertical. Entonces, 1/50 o 1/60 [s] después, la imagen completa es nuevamente escaneada, pero compensada verticalmente para así llenar los espacios. Ahora un cuadro comprende dos campos (*fields)*, denotados primero y segundo. El método de escaneo es esquematizado en la Figura 2-5.

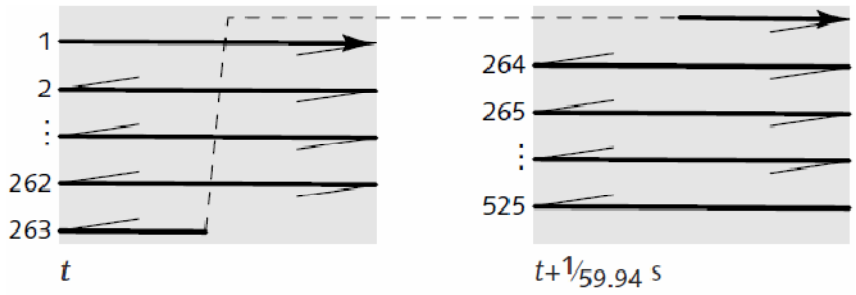


Figura 2-5: Escaneo entrelazado.

La capacidad de los canales de transmisión en los inicios de la emisión de televisión era insuficiente para transmitir tres componentes de color por separado. Las técnicas NTSC (proveniente de *National Television System Comittee,* comité creador de la norma) y PAL (*Phase Alternating Line)* fueron inventadas para combinar los tres componentes de color en una única señal compuesta.

Los sistemas de video pueden ser clasificados como HDTV (*High Defini*tion *Television*) por componente, SDTV (*Standard Definition Television*) por componente o SDTV compuesta. Independientemente, un sistema puede ser clasificado como digital o análogo. La Tabla 2-1 indica seis clasificaciones, con su asociada codificación de color. La codificación de video NTSC y PAL compuesto solo es usada en los sistemas 480i y 576i; los sistemas HDTV solo usan video por componentes. S-video es una combinación de video análogo por componentes y NTSC o PAL compuesto; en la Tabla 2-1 está clasificado como una séptima categoría.

		Análoga	Digital
HDTV	Componente	R'G'B', $^{709}Y'P_BP_R$	4:2:2 $^{709}Y'C_BC_R$
SDTV	Componente	R'G'B', $^{601}Y'P_BP_R$	4:2:2 $^{601}Y'C_BC_R$
	Hibrido	S-video	
	Compuesto	NTSC, PAL	$4f_{SC}$

Tabla 2-1: Clasificación de sistemas de video.

2.2 Recomendación ITU-R BT.656-5

Esta recomendación tiene el título "Interfaces para las señales de vídeo con componentes digitales en sistemas de televisión de 525 líneas y 625 líneas que funciona en nivel 4:2:2 de la recomendación ITU-R BT.601"[3] y su objetivo es describir el medio de interconexión entre los equipos de televisión digital de 525 y 625 líneas. Este documento se separa en dos partes, donde la primera describe el formato de la señal digital de la interfaz y la segunda las características particulares de la interfaz para bits en serie. En este documento solo se realizará una pequeña reseña de la primera parte de la recomendación.

2.2.1 Descripción general de la interfaz

La interfaz proporciona una conexión unidireccional entre un solo origen y un solo destino. Las señales de datos van en forma de información binaria codificada en palabras de 8 bits o de 10 bits. Esas señales son: señales de video; datos de supresión digital; señales de referencia de temporización; y señales auxiliares.

Las palabras de un bytes ocupan los bits más significativos de la izquierda de la palabra de 10 bits; es decir, del noveno al segundo, siendo el noveno el bit más significativo.

2.2.2 Datos de video

2.2.2.1 Características de codificación

Los datos de vídeo son conformes a la Recomendación ITU-R BT.601, y a las definiciones de los intervalos de supresión de trama que se incluyen en la Tabla 2-2.

		625	525
V-supresión de trama digital			
Trama 1	Comienzo (V = 1)	Línea 624	Línea 1
	Final (V = 0)	Línea 23	Línea 20
Trama 2	Comienzo (V = 1)	Línea 311	Línea 264
	Final (V = 0)	Línea 336	Línea 283
F-identificación de trama digital			
Trama 1	F = 0	Línea 1	Línea 4
Trama 2	F = 1	Línea 313	Línea 266

Tabla 2-2: Definición de los intervalos de trama segun Recomendación ITU-R BT.601.

2.2.2.2 Formato de los datos de vídeo

Las palabras de datos cuyos 8 bits más significativos están todos puestos a "1" o a "0" se reservan para fines de identificación de datos y, en consecuencia, sólo pueden utilizarse 254 de las 256 palabras posibles de 8 bits (o 1016 de las 1024 palabras posibles de 10 bits) para expresar el valor de una señal.

Las palabras de datos de video se transmiten en forma de un múltiplex de 27 Mpalabras/s, en el orden C_B, Y, C_R, Y, C_B, Y, C_R, etc. Donde la secuencia de palabras (C_B, Y, C_R) se refiere a las muestras de luminancia y de diferencia de color correspondiente a un mismo punto y la siguiente palabra (Y) corresponde a la siguiente muestra de luminancia.

2.2.2.3 Estructura de la señal en la interfaz

La Figura 2-6 muestra la manera en que los datos de la muestra de video se incorporan en el tren de datos de la interfaz. La identificación de las muestras en la Figura 2-6 concuerda con la Recomendación ITU-R BT.601[4].

2.2.2.4 Códigos de referencia para la temporización de la señal de video (SAV, EAV)

Existen dos códigos de referencia para la temporización, uno al comienzo de cada bloque de datos de video (*Start of Active Video*, SAV) y la otra al final de cada bloque de datos de video (*End of Active Video*, EAV), como se muestra en la Figura 2-6.

Cada señal de referencia de temporización consiste en una secuencia de cuatro palabras con el formato siguiente: 3FF 000 000 XYZ (valores expresados en notación hexadecimal). Las tres primeras palabras son un preámbulo fijo. La cuarta palabra contiene información que define la identificación de la trama 2, el estado del intervalo de supresión de trama y el estado de

supresión de línea. En la Tabla 2-3 se muestra la asignación de los bits dentro del código de referencia de temporización.

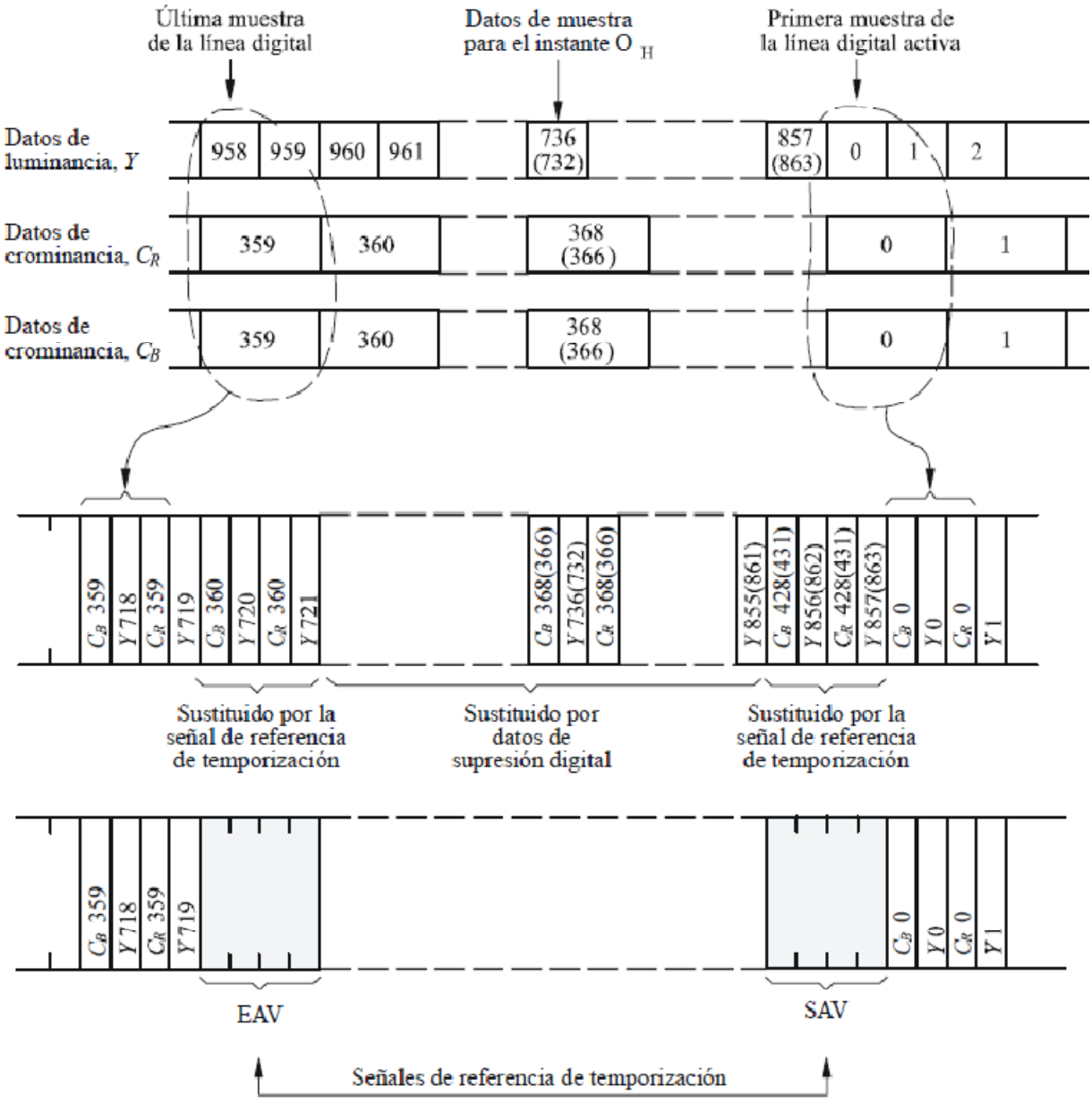


Figura 2-6: Composición del tren de datos en la interfaz.

Número del bit de datos	Primera palabra (FF)	Segunda palabra (00)	Tercera palabra (00)	Cuarta palabra (XY)
9 (MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P_3
4	1	0	0	P_2
3	1	0	0	P_1
2	1	0	0	P_0
1	1	0	0	0
0	1	0	0	0

Tabla 2-3: Códigos de referencia de temporización de video.

Donde F es 0 durante la trama 1 y 1 durante la trama 2, V es 0 fuera de la supresión de trama vertical y 1 durante esta, H es 0 en SAV y 1 en EAV, MSB es el bit más significativo y P3, P2, P1, P0 son bits de protección los cuales dependen de los bits F, V y H, como se muestra en la Tabla 2-4. En el receptor, esta disposición permite la corrección de un bit erróneo y la detección de dos bits erróneos.

F	**V**	**H**	**P_3**	**P_2**	**P_1**	**P_0**
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Tabla 2-4: Bits de protección.

2.3 *Field Programmable gate array* (FPGA)

Es un dispositivo semiconductor que puede ser programado luego de su producción. En vez de estar limitado a una funcionalidad de hardware, una FPGA permite ser programado para las características y funciones del producto, adaptándose a nuevos estándares, y reconfigurando hardware para aplicaciones especificas, incluso después de que el producto ha sido instalado. Las FPGAs pueden usarse para implementar cualquier función lógica que un circuito integrado para aplicaciones especificas o ASIC (*Aplication Specific Integrated Circuit*) pueda realizar, pero la habilidad de para actualizar las funcionalidades luego de su producción es lo que ofrece ventajas para muchas aplicaciones.

A diferencia de generaciones anteriores de FPGA, las cuales usan entradas y salidas con lógica y interconexiones programables, las FPGA actuales consisten en varias mezclas de SRAM embebida, transceptor (*transceiver)* de alta velocidad, I/Os de alta velocidad, bloques lógicos y ruteos programables. Una FPGA contiene componentes lógicos programables y una jerarquía de interconexiones reconfigurables que permite a los componentes lógicos estar conectados. Los componentes lógicos pueden ser configurados para realizar funciones combinacionales complejas, o simplemente compuertas lógicas como AND y XOR. En la mayoría de las FPGAs, lo bloques lógicos contienen además elementos de memoria, los cuales pueden ser simples *flipflops* o completos bloques de memoria.

Las ventajas que tiene el diseño con FPGA en comparación al utilizado con ASIC es la rapidez para hacer prototipos, los menores tiempo al mercado, la habilidad de reprogramar para el *debugging*, y disminuye los costos por no recurrencia de ingeniería o NRE (*Non-recurring engineering*).

2.4 Familia Extended Spartan-3A[2]

Esta familia de FPGAs está compuesta por las plataformas Spartan-3A, Spartan-3AN, y Spartan-3A DSP, la cual están especialmente diseñada para cumplir con las necesidades de aplicaciones de alto volumen de producción y bajos costos.

Esta familia utiliza la tecnología de 90nm en su producción, la cual permite mayor funcionalidad y ancho de banda por dólar que tecnologías anteriores. En cuanto la familia Extended Spartan-3A es una evolución en la familia Spartan-3E mejorando su configuración y reduciendo la utilización de energía, produciendo menores costos. Entre sus plataformas se encuentra la plataforma Spartan- 3AN la cual añade el beneficio de poseer memoria no volátil y la plataforma Spartan-3A DSP que extiende el rango de densidad y añade recursos altamente utilizados en aplicaciones de procesamiento digital de señales (DSP, *Digital Signal Processing*).

A continuación se muestra una tabla comparativa entre las tres plataformas, que muestra el soporte de distintas funcionalidades y aplicaciones.

Aplicaciones/Funciones	**Plataformas Familia Extended Spartan-3A**		
	Spartan-3A	**Spartan-3AN**	**Spartan-3A DSP**
Características básicas de diseño			
Mas de 1.5M compuertas de sistema			+
500 o menos pines de entrada/salida	++	++	++
Procesamiento Embebido			
Procesador de 32-bit MicroBlaze	++	++	++
Procesador de 8-bit PicoBlaze	+	+	+
Interfaces con memoria DDR SDRAM			
DDR SDRAM	++	++	++
DDR2 SDRAM	++	++	++
Entrada/Salida Diferencial			
LVDS	+++	+++	+++
RSDS	+++	+++	+++
miniLVDS	++	++	++
TMDS/PPDS	+	+	+
Interfaz PCI®/PCI Express®			
Interfaz PCI 33 MHz	++	++	++
Interfaz PCI 66 MHz	++	++	++
Interfaz PCI Express PIPE	+	+	+
Manejo de Energía	+	+	+

Nota: + = soportado, ++ = mejor, +++ = el mejor.

Tabla 2-5: Funcionalidades y aplicaciones de familia Extended Spartan-3A.

[2] Traducción de extractos de documento *Spartan-3 Generation FPGA User Guide[5]*

Aplicaciones/Funciones	Plataformas Familia Extended Spartan-3A		
	Spartan-3A	Spartan-3AN	Spartan-3A DSP
Capacidades de Entrada/Salida			
Hot Swap	++	++	++
High Output Drive Current	++	++	++
Retrasos de entradas programables	+++	+++	+++
Aplicaciones de solo 3.3V	++	++	++
Recursos de Reloj			
Digital Clock Managers (DCMs)	++	++	++
Low-Skew Global Clocks	++	++	++
Configuración FPGA			
Plataforma Flash PROM	+	+	+
Configuración por SPI Flash	+	+	+
Configuración por Parallel Flash	+	+	+
MultiBoot	+	+	+
Protección de Diseño de bajo costo	++	+++	++
No Volatil		+	
Flash del usuario integrada		+	
Procesamiento Digital de Señales (DSP)			
18x18 Multiplicador por Hardware	++	++	+++
DSP48A			+
Registros de Block RAM	++	++	+++

Nota: + = soportado, ++ = mejor, +++ = el mejor.

Nota: + = soportado, ++ = mejor, +++ = el mejor.

Tabla 2-5: **Funcionalidades y aplicaciones de familia Extended Spartan-3A. (Continuación)**

2.4.1 Recursos de Familia Extended Spartan-3A

A continuación se muestra un sumario con los dispositivos de cada una de las plataformas de la familia Extended Spartan-3A.

Device	System Gates	Equivalent Logic Cells	CLBs	Slices	Distributed RAM Bits	Block RAM Bits	DSP48As	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
XC3SD1800A	1800K	37.440	4.160	16.640	260K	1.512K	84	8	519	227
XC3SD3400A	3400K	53.712	5.968	23.872	373K	2.268K	126	8	469	213

Tabla 2-6: Sumario de Recursos de dispositivos de plataforma Spartan-3A DSP.

Device	System Gates	Equivalent Logic Cells	CLBs	Slices	Distributed RAM Bits	Block RAM Bits	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs	In-System Flash bits
XC3S50AN	50K	1.584	176	704	11K	54K	3	2	108	50	1M
XC3S200AN	200K	4.032	448	1.792	28K	288K	16	4	195	90	4M
XC3S400AN	400K	8.064	896	3.584	56K	360K	20	4	311	142	4M
XC3S700AN	700K	13.248	1472	5.888	92K	360K	20	8	372	165	8M
XC3S1400AN	1400K	23.344	2816	11.264	176K	576K	32	8	502	227	16M

Tabla 2-7: Sumario de Recursos de dispositivos de plataforma Spartan-3AN.

Device	System Gates	Equivalent Logic Cells	CLBs	Slices	Distributed RAM Bits	Block RAM Bits	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
XC3S50A	50K	1.584	176	704	11K	54K	3	2	144	64
XC3S200A	200K	4.032	448	1.792	28K	288K	16	4	248	112
XC3S400A	400K	8.064	896	3.584	56K	360K	20	4	311	142
XC3S700A	700K	13.248	1472	5.888	92K	360K	20	8	372	165
XC3S1400A	1400K	23.344	2816	11.264	176K	576K	32	8	502	227

Tabla 2-8: Sumario de Recursos de dispositivos de plataforma Spartan-3A.

2.4.2 Arquitectura de familia

La familia de FPGAs consiste en cinco elementos funcionales programables los cuales se comentan en las siguientes sub-secciones.

2.4.2.1 Bloques lógicos configurables (CLBs, Configurable Logic Blocks)

Los CLBs constituyen el principal recurso lógico para implementar tanto circuitos síncronos como combinacionales. Cada CLB contiene cuatro tajadas (*slices*) y cada tajada contiene dos *Look-Up Tables* (LUTs) para implementar lógica combinacional y dos elementos de almacenamiento que pueden ser usados como *flipflops* o *latches*. Las *LUTs* pueden ser usados como memorias de 16x1 (RAM16) o como un registro de cambio (*shift register)* de 16 bits (SRL16), y junto con multiplexadores y lógica de acarreo (*carry logic*) adicionales permiten la simplificación de funciones aritméticas y lógicas más complejas.

Los CLBs están posicionados en arreglos regulares de filas y columnas como se muestra en la Figura 2-7.

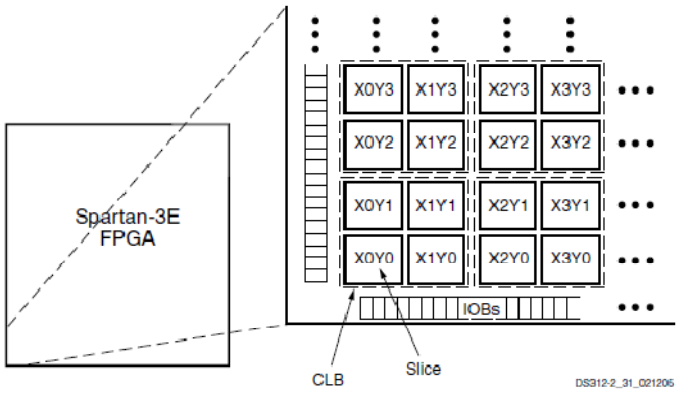


Figura 2-7: Localización de CLBs.

Como fue mencionado cada CLB está constituido por cuatro tajadas interconectadas como se muestra en la Figura 2-8 .Estas tajadas están agrupadas en pares. Cada par está organizado en columnas con cadenas de acarreo (*carry chain*) independientes. El par izquierdo soporta tanto lógica como funciones de memoria y estas tajadas son llamados SLICEM. El par derecho solo soporta lógica y estos son llamados SLICEL. Una mejor apreciación de los recursos de un CLB es mostrado en la Figura 2-9.

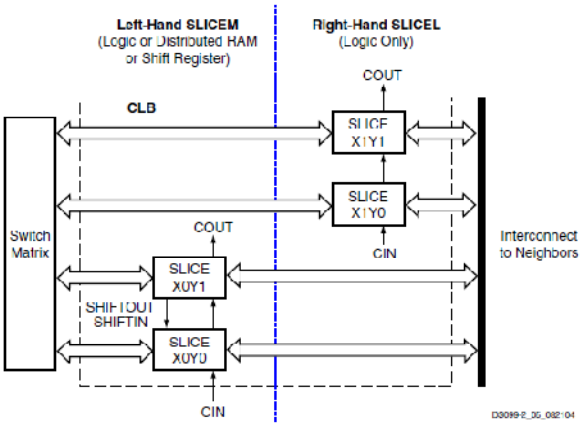


Figura 2-8: Configuración de *Slices* dentro de un CLB.

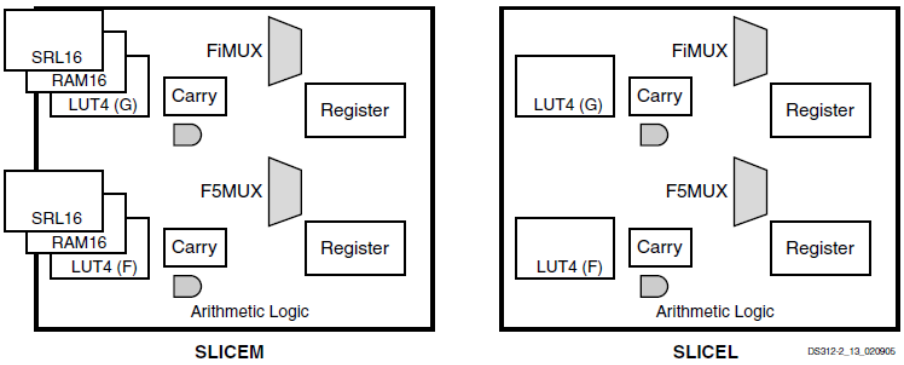


Figura 2-9: Recursos en una *slice*.

Las *Look-Up Tables* o LUTs son generadores de funciones basados en RAM y son el principal recurso para la implementación de funciones lógicas. Más aun, las LUTs en cada par SLICEM puede ser configurado como RAM Distribuida o un *shift register* de 16 bits como fue descrito anteriormente.

Cada LUT en la *slice* tiene cuatro entradas lógicas y una salida. Cualquier operación booleana de cuatro variables puede ser implementada en una LUT. Funciones con más entradas pueden ser implementadas distribuyendo LUTs en cascadas o usando *wide multiplexer*.

Los *wide-function multiplexer* efectivamente combinan LUTs para permitir operaciones lógicas más complejas. Cada *slice* tiene dos de estos *multiplexers* con F5MUX en la parte inferior de la *slice* y FiMUX en la superior. El F5MUX multiplexa las dos LUTs en la *slice.* En tanto FiMUX multiplexa dos entradas al CLB que conectan directamente a las salidas de F5MUX y FiMUX de la misma o otras *slices.*

La cadena de acarreo (*carry chain)*, junto con varias compuertas aritméticas lógicas dedicadas, permiten una eficiente y rápida implementación de operaciones matemáticas. La lógica de acarreo es automáticamente usada por la mayoría de las funciones aritméticas en un diseño. Las compuertas y *multiplexers* del acarreo y lógica aritmética pueden ser usadas también para lógica de uso general.

Los elementos de almacenamiento, que pueden ser programados tanto en un *flipflop* tipo D o en un *latch* transparente sensible al nivel, proveen un medio para sincronizar la información a una señal de reloj, entre otros usos. Los elementos de almacenamiento en las porciones inferior y superior de la *slice* son llamados FFX y FFY respectivamente. FFY tiene un *multiplexer* fijo en su entrada seleccionando ya sea la salida combinacional Y o la señal de *bypass* BY. FFX tiene un *multiplexer* fijo en su entrada seleccionando ya sea la salida combinacional X o la señal de *bypass* BX.

La funcionalidad de un elemento de almacenamiento, en el caso de ser programado como un *flipflop* D que se esquematiza en la siguiente figura, puede ser descrita por la Tabla 2-9. Todas las señales tienen polaridad programable; por defecto es descrita la funcionalidad activo en alto (*active-High)*.

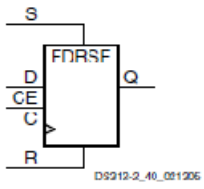


Figura 2-10: Esquema de componente Flipflop con Reset, Set y Clock Enable síncronos.

Entradas					Salidas
R	**S**	**CE**	**D**	**C**	**Q**
1	X	X	X	↑	0
0	1	X	X	↑	1
0	0	0	X	X	Sin Cambio
0	0	1	1	↑	1
0	0	1	0	↑	0

Tabla 2-9: Descripción Funcional de Flipflop D con Reset, Set y Clock Enable Sincronos.

En la Figura 2-11 se muestra un diagrama de la distribución e interconexión de los elementos antes comentados para una SLICEM.

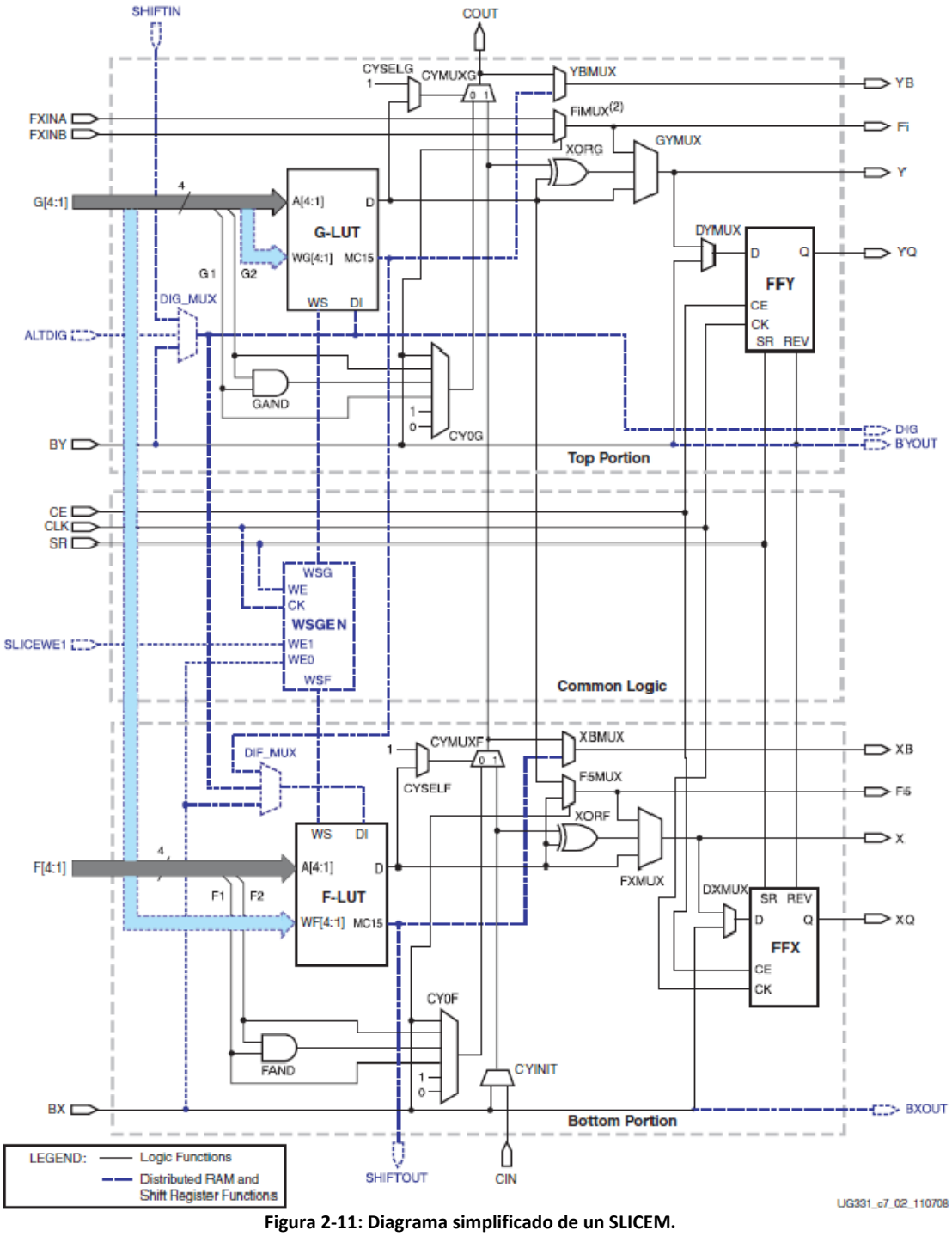


Figura 2-11: Diagrama simplificado de un SLICEM.

2.4.2.2 Input/Output Blocks (IOBs)

Los IOB proveen una interfaz programable, unidireccional o bidireccional entre los pines del chip y la lógica interna de la FPGA, soportando una gran variedad de estándares de interfaz. La colección de características robustas incluye un control programable de la fuerza y tasa de slew de la salida, entradas y salidas combinacionales o registradas con registros doble tasa de transferencia de datos (DDR, *Double Data Rate*), retrasos de entrada programables, terminaciones en el chip y la capacidad de *hot-swap*.

La Figura 2-12 muestra un diagrama simplificado de la estructura interna de los IOBs. Existen tres caminos principales de señales dentro de los IOBs; el camino de salida, el camino de entrada y el camino 3-estados. Cada camino tiene su propio par de elementos de almacenamiento que pueden actuar ya sea como registros o *latches*. Los tres caminos principales son los siguientes:

- El camino de entrada lleva los datos desde el *pad*, el cual esta adherido a un pin del chip, a través de un elemento de retraso programable directamente a la línea I. Después del elemento de retraso, existen rutas alternativas a través de un par de elemento de almacenamiento a las líneas IQ1 y IQ2. Las salidas de IOB I, IQ1 y IQ2 conduce a la lógica interna de la FPGA. El elemento de retraso se puede fijar para asegurar un tiempo de *hold* nulo.

- El camino de salida, parte con las líneas O1 y O2, llevan datos desde la lógica interna de la FPGA a través de un multiplexador y luego un manejador de 3-estados al pad IOB. Adicionalmente al camino directo, el multiplexador provee la opción de insertar un par de elementos de almacenamiento.

- El camino de 3-estados determina cuando el manejador de salida está en alta impedancia. Las líneas T1 y T2 transportan información desde la lógica interna de la FPGA a través del multiplexador, al manejador de salida. Adicionalmente al camino directo, el multiplexador provee la opción de insertar un par de elementos de almacenamiento.

Todos los caminos de señales entrando al IOB, incluidas las asociadas con los elementos de almacenamiento tienen la opción de un inversor. Cualquier inversor puesto en estos caminos es automáticamente absorbido en el IOB.

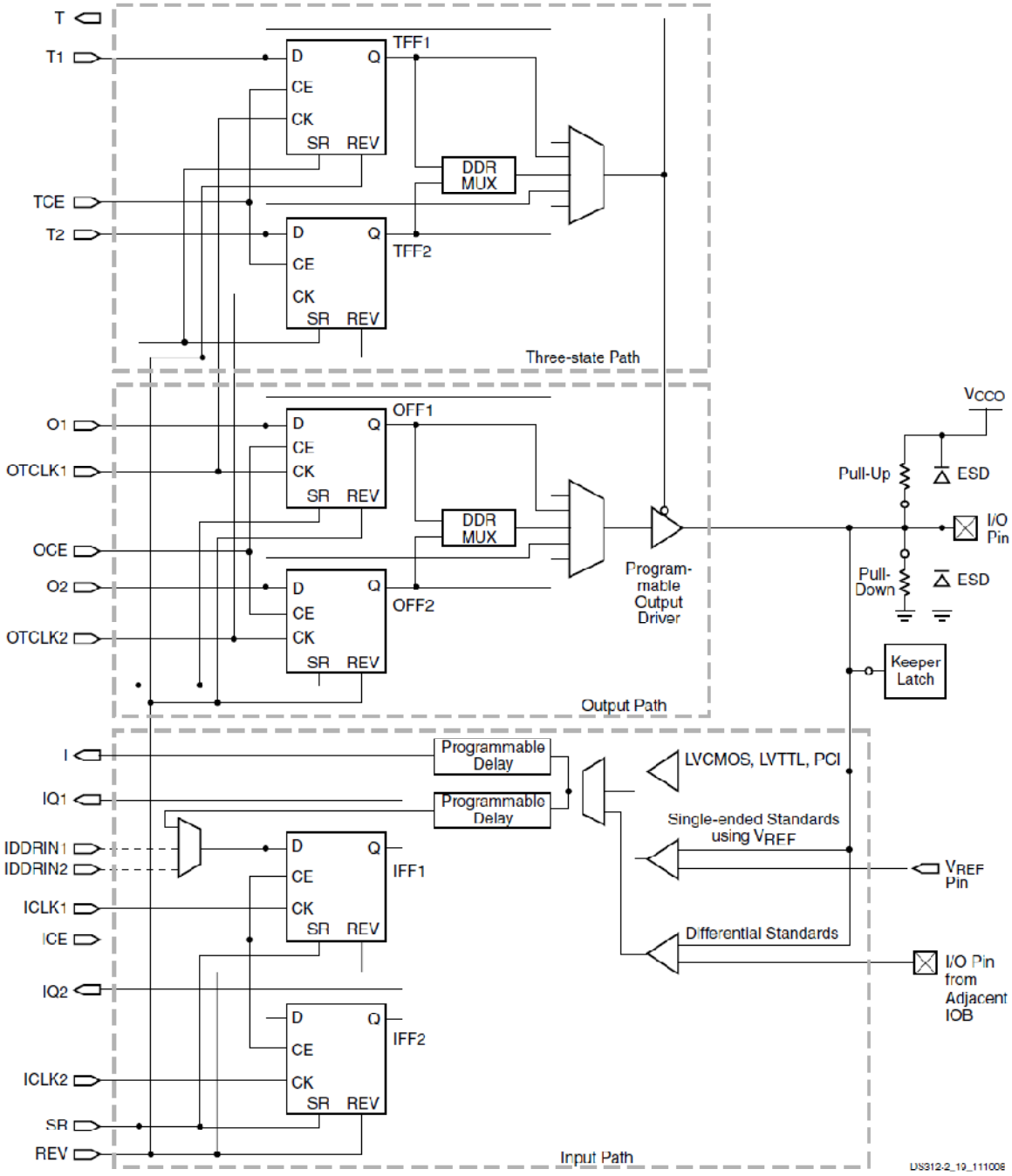


Figura 2-12: Diagrama simplificado de IOB.

Otra característica de las entradas y salidas de los IOBs es el amplio rango de estándares de señales de una terminación que soporta. La mayoría de las entradas/salidas también pueden ser usadas para formar pares diferenciales que soportan cualquiera de los estándares de este tipo de señales. Esta flexibilidad permite al usuario seleccionar el mejor estándar para cada pin, con tal de cumplir los requerimientos de la aplicación. La Tabla 2-10 muestra los estándares soportados por la familia Extended Spartan-3A.

Estándar	Descripción	Especificación	Uso/Patrocinador	Buffer de entrada	Buffer de Salida
Estándares de una terminación					
LVTTL	Low Voltage TTL	JESD8C	Uso general 3.3V	LVTTL	Push-Pull
LVCMOS	Low Voltage CMOS	JESD8C	Uso general	LVCMOS	Push-Pull
PCI	Peripheral Component Interconnect	PCI SIG	Bus PCI	LVTTL	Push-Pull
GTL	Gunning Transceiver Logic	JESD8-3	High-speed bus, backplane; Xerox	Basado en V_{ref}	Open-Drain
GTL+	GTL plus		Intel Pentium Pro		
HSTL	High-Speed Transceiver Logic	JESD8-6	Hitachi SRAM; IBM; soporta 3 de 4 clases	Basado en V_{ref}	Push-Pull
SSTL3	Stub Series Terminated Logic for 3.3V	JESD8-8	Bus SRAM/SDRAM; Hitachi y IBM; dos clases	Basado en V_{ref}	Push-Pull
SSTL2	SSTL for 2.5V	JESD8-9			
SSTL18	SSTL for 1.8V	JC42.3			
Estándares Diferenciales					
LVDS	Low Voltage Differential Signaling	ANSI/TIA/EIA-644-A	High-speed interface, backplane, video; National, TI	Par diferencial	Par diferencial
BLVDS	Bus LVDS	AMSI/TIA/EIA-644-A	Multipoint LVDS	Par diferencial	Par diferencial
LVPECL	Low Voltage Positive ECL	Freescale Semiconductor (formely Motorola)	High-Speed clocks	Par diferencial	Par diferencial
LDT	Lightning Data Transport (HyperTransport)	HyperTransport Spec v3.0	Bidirectional serial/parallel high-bandwidth, low-latency computer; HyperTransport Consortium	Par diferencial	Par diferencial
MINI_LVDS	Mini-LVDS	TI	Flat Panel Display	Par diferencial	Par diferencial
LVDSEXT	LVDS Extended	Extensión de LVDS	Higher drive requirements	Par diferencial	Par diferencial
RSDS	Reduced Swing Differential Signaling	National Semiconductor	Flat panel display	Par diferencial	Par diferencial

Tabla 2-10: Estándares de señales entrada/salida.

2.4.2.3 Block RAM

Para aplicaciones que requieren grandes cantidades de memoria embebida la familia Extended Spartan-3A, provee abundante y eficientes bloques de memoria SelectRAM. Usando diversas opciones de configuración, los bloques SelectRAM permiten crear RAM, ROM, FIFOs, grandes LUTs, convertidores del ancho de datos (*data width*), *buffers* circulares, y *shift registers*, cada uno soportando varios anchos de datos y profundidades (*data depth*).

Los bloques de RAMs están organizados en columnas como muestra la Figura 2-13, donde se muestra su distribución para el dispositivo XC3200A. Además se puede apreciar que cada bloque de RAM está adyacente a un multiplicador embebido, esto permite mejorar el desempeño de ciertas funciones DSP. Interconexiones especiales rodean los bloques de RAM, las cuales proveen una distribución eficiente de señales de dirección y datos.

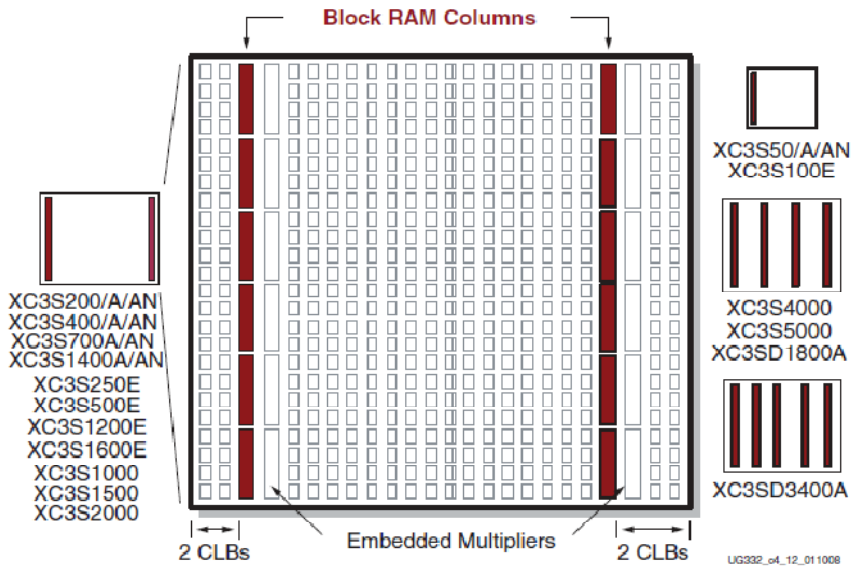


Figura 2-13: Bloques de RAM arreglados en columnas con *Floorplan* detallado de dispositivo XC3S200.

La cantidad total de bloques de RAM dependen del tamaño del dispositivo, tal como se muestra en la Tabla 2-11.

Dispositivo	Columnas	Bloques de RAM por columna	Total de Bloques de RAM	Total RAM Bits	Total RAM Kbits
XC3SD1800A	4	20-22	84	1.548.288	1.512K
XC3SD3400A	5	24-26	126	2.322.432	2.268K
XC3S50A/AN	1	3	3	55.296	54K
XC3S200A/AN	2	8	16	294.912	288K
XC3S400A/AN	2	10	20	368.640	360K
XC3S700A/AN	2	10	20	368.640	360K
XC3S1400A/AN	2	16	32	589.824	576K

Tabla 2-11: Bloques de RAM disponibles en familia Extended Spartan-3A FPGAs.

Cada bloque de RAM contiene 18.432 bits de *fast static RAM*, de los cuales 16Kbits estan destinados a almacenamiento de información y, en algunas configuraciones de memoria, 2K adicionales son destinados para paridad o bits adicionales de información. Físicamente, un bloque de RAM tiene dos puertos de accedo completamente independientes, etiquetados como *Port A* y *Port B*. La estructura es totalmente simétrica, y ambos puertos son intercambiables y soportan operaciones de lectura y escritura de datos. Cada puerto de memoria es síncrono con su propio reloj, habilitadores de reloj y escritura (*clock enable y write enable*). Las operaciones de lecturas también son síncronas y requieren de un canto del reloj y un habilitador de reloj.

Si bien físicamente es una memoria de *Dual-Port*, el bloque de RAM simula una memoria *Single-Port* en una aplicación, como se muestra en la Figura 2-14. Más aun, cada bloque de memoria soporta múltiples configuraciones, las que se señalan en la Tabla 2-12 .

Los bloques de RAM en la plataforma Spartan-3A DSP incluye un registro de salida opcional. Este registro de salida permite operación a velocidades superiores a los 250 MHz para todos los anchos de datos.

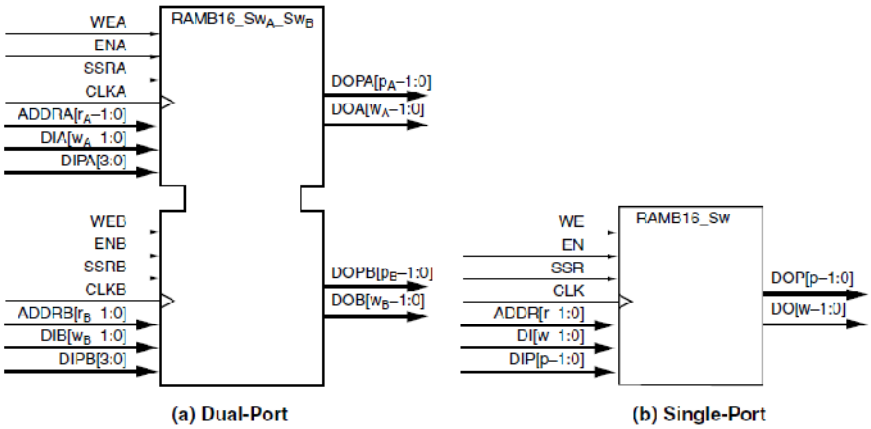


Figura 2-14: Bloques SelectRAM de 18K funcionando como Dual-Port (a) y Single-Port (b).

Cantidad Total de RAM bits, incluyendo paridad	18,432(16K datos + 2K paridad)
Organización de Memoria	16Kx1 8Kx2 4Kx4 2Kx8 (sin paridad) 2Kx9 (x8 + 1 bit de paridad) 1Kx16 (sin paridad) 1Kx18 (x16 + 2 bits de paridad) 512x32 (sin paridad) 512x36 (x32 + 4 bits de paridad) 256x72 (solamente *single-port*)
Paridad	Disponible y opcional solo en las organizaciones mayores a un byte de ancho.
Desempeño	240+ MHz(Dependiente de dispositivos)
Interface de Cronometraje	Interfaz síncrona. Similar a la lectura y escritura de un registro con *setup time* para operaciones de escritura y un retraso *clock-to-output* para operaciones de lectura.

Tabla 2-12: Características y aplicaciones de bloque de memoria SelectRAM 18K.

2.4.2.4 Bloques de Multiplicadores Embebidos

Multiplicadores de 18x18 agilizan la lógica DSP en la familia Extended Spartan-3A. Los multiplicadores son rápidos y eficientes para implementar multiplicaciones con y sin signo de hasta 18 bits. Además de la función básica de multiplicación, los bloques de multiplicadores embebidos pueden ser usados como *shifter* o generar magnitud o retornar el valor en complemento de dos. Los multiplicadores pueden ser organizados en cascadas con otros multiplicadores o lógica CLB para más largas y complejas funciones.

La plataforma Spartan-3A DSP incluye bloques DSP48A de alto desempeño, los cuales soportan operaciones multiplicativas acumulativas hasta 250MHz.

2.4.2.4.1 Multiplicador 18x18

La primitiva del multiplicador de 18x18 es MULT18X18SIO el cual es mostrado en Figura 2-15.Cada multiplicador realiza la operación principal P = A x B, donde A y B son palabras de 18-bit en complemento dos y P es un producto de precisión completa de 36 bits, también en complemento dos. Los 18 bits de entrada representan valores que fluctúan desde -131.072_{10} hasta $+131.071_{10}$ y el producto resultante fluctúa desde $-17.179.738.112_{10}$ hasta $+17.179.869.184_{10}$.

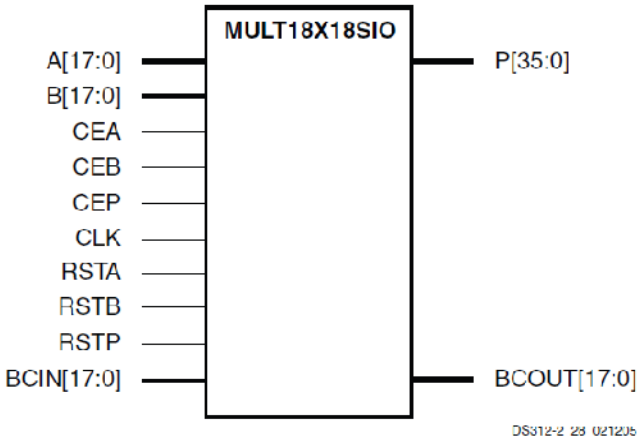


Figura 2-15: Primitiva MULT18X18SIO.

2.4.2.4.2 DSP48A

Cada *slice* DSP48A tiene pre-sumador de 18 bits seleccionable que genera un resultado en 18 bits. El pre-sumador es seguido por un multiplicador de dos entradas, multiplexadores y un sumador/restador de dos entradas. El multiplicador acepta dos coeficientes de 18 bits y un producto de 36 bits. El resultado es extendido en signo a 48 bits y puede opcionalmente ser suministrado al sumador/restador. El sumador/restador acepta dos entradas de 48 bits y genera un resultado en 48 bits.

En Figura 2-16 se muestra la primitiva del DSP48A, para mayor información sobre el uso de éste referirse al texto "XtremeDSP DSP48A for Spartan-3A DSP FPGAs"[6].

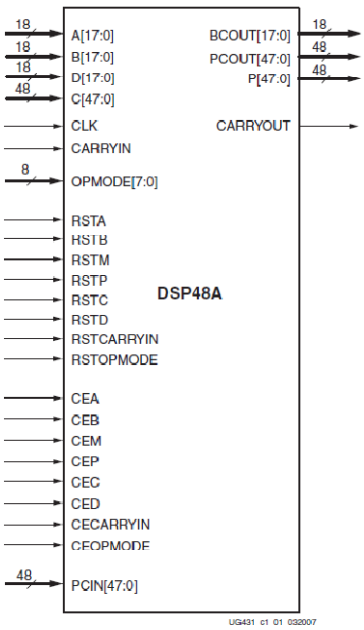


Figura 2-16: Primitiva DSP48A.

2.4.2.5 Digital Clock Manager (DCM) Blocks

Los DCMs integran capacidades avanzadas de reloj directamente en la red global de distribución de relojes de las FPGAs. Consecuentemente, los DCMs resuelven una variedad de asuntos comunes de los relojes, especialmente para alto desempeño y aplicaciones de alta frecuencia:

- **Elimina el desajuste del reloj (*Clock Skew)***, ya sea dentro del dispositivo o en componentes externos, para mejorar el desempeño general del sistema y eliminar los retrasos de la distribución de los relojes.
- **El cambio de fase** de la señal de reloj, ya sea en una fracción fija del periodo del reloj o por valores incrementales.
- **Reacondicionamiento del reloj,** asegurándose una salida del reloj limpia con ciclo de servicio de 50%.
- **Reflejo, Adelantamiento o Rebuffer de la señal de reloj,** usualmente para eliminar el desajuste del reloj y convertir la señal de entrada a un diferente estándar de entrada/salida; por ejemplo, adelantando y convirtiendo una señal entrante de reloj LVTTL a LVDS.
- **Cualquiera o todas las funciones anteriores simultáneamente.**

2.4.2.5.1 Descripción Funcional

Una sola entidad que es generalmente llamada DCM, consiste en cuatro unidades funcionales distintas, como se representa en el diagrama mostrado en la Figura 2-17 y descrito en las siguientes secciones. Estas unidades operan independientemente o en conjunto.

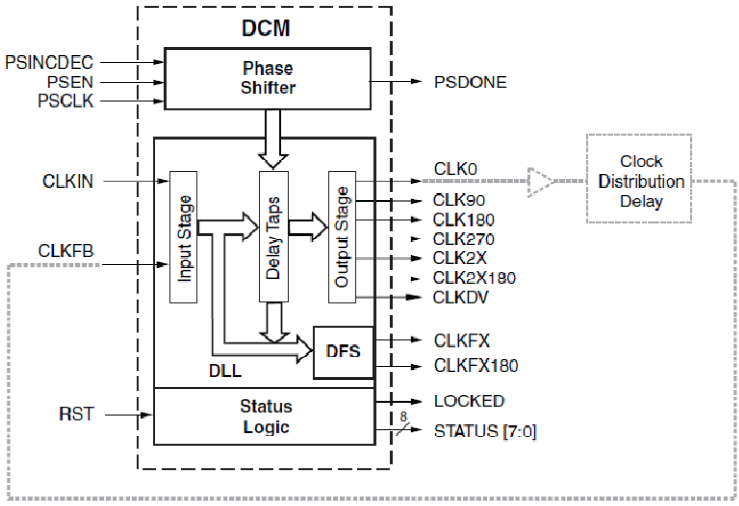


Figura 2-17: Diagrama de bloques de funcionalidades de un DCM.

2.4.2.5.1.1 Retardo de bucle cerrado (DDL, Delay-Locked Loop)

La unidad DLL provee la circuitería digital para la eliminación del desajuste de reloj el cual efectivamente genera una salida de reloj con un retraso nulo en la red. El circuito de eliminación de desajuste compensa los retrasos de las rutas de red de reloj a través del monitoreo de la salida de reloj, ya sea la salida CLK0 o CLK2X.

Las entradas de la unidad DLL son CLKIN y CLKBF. Las señales de salidas del DLL son CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, y CLKDV.

2.4.2.5.1.2 Sintetizador Digital de Frecuencia (DFS, Digital Frequency Synthesizer)

Esta unidad provee un amplio y flexible rango de frecuencias de salida basadas en la proporción de dos números enteros definidos por el usuario, un multiplicador (CLKFX_MULTIPLY) y un divisor (CLKFX_DIVIDE). La frecuencia de salida es derivada del reloj de entrada (CLKIN) por la multiplicación y división simultanea de frecuencia. La característica del DFS puede ser usada en conjunto o por separado de la característica del DLL del DCM. Si el DLL no es usado, entonces no existe relación de fase entre CLKIN y las salidas del DFS.La unidad DFS genera las salidas de frecuencia sintetizada CLKFX y CLKFX180.

2.4.2.5.1.3 Cambiador de Fase (PS, Phase Shift)

La unidad PS controla la relación de fase de las salidas del reloj a la entrada CLKIN. Además cambia la fase de las 9 salidas de reloj del DCM por una fracción fija del periodo del reloj de entrada. El valor del cambio de fase es fijado en el diseño y cargado en el DCM durante la configuración de la FPGA.

La unidad PS también provee una interface digital para la aplicación FPGA de dinámicamente avanzar o retrasar el valor actual de cambio, llamado cambio variable de fase.

2.4.2.5.1.4 Lógica de Estado (Status Logic)

La lógica de estado indica el estado actual del DCM a través de las señales de salida LOCKED y STATUS [3:0]. La señal LOCKED indica si las salidas del DCM están en fase con la entrada CLKIN y STATUS indica el estado de las operaciones de las unidades DLL y PS.

2.4.2.5.2 Primitiva del DCM

La primitiva del DCM es mostrada en la Figura 2-18, esta representa todas las características dentro del DCM. El nombre de la primitiva es DCM_SP.

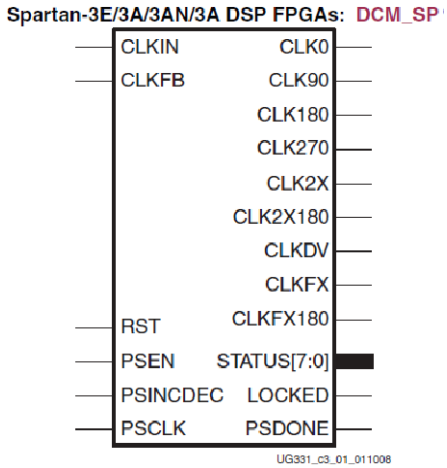


Figura 2-18: Primitiva del DCM.

2.5 Plataforma de desarrollo Spartan-3A FPGA Starter Kit[3]

2.5.1 Componentes y características

- Xilinx 700K-compuertas XC3S700A en el paquete 484-ball BGA (FGG484).
- PROM para la configuración de la plataforma de Xilinx de 4 Mbits.
- DDR2 SDRAM de 64Mbytes, en interfaz de datos de 32Mx16.
- Memoria *Flash* NOR paralela de 4 Mbytes.
- Dos memorias *Flash* seriales SPI de 16Mbits.
- Pantalla LCD de 16 caracteres.
- Puerto PS/2.
- Puerto VGA de 12-bits de color.
- 10/100 Ethernet PHY.
- Dos puertos RS-232 (estilos DTE y DCE).
- Reloj oscilador de 50 MHz.
- Socket DIP de 8-pines para segundo oscilador
- Conector SMA para entradas o salidas de de reloj.

[3] Traduccioón de extractos de documento *Spartan-3A/3AN FPGA Starter Kit Board User Guide*[7].

- Conector de expansión Hirose FX2 de 100-pines, con un máximo de 43 I/Os del usuario de la FPGA.
- Conectores diferenciales I/O de alta velocidad.
- Dos conectores de expansión de 6-pines.
- Convertidor Digital a Análogo (*Digital to Analog Converter*, DAC).
- Convertidor Análogo a Digital (*Analog to Digital Converter,* ADC).
- *Rotary-encoder* con eje botón.
- Ocho LEDs.
- Cuatro interruptores.
- Cuatro botones.

2.5.2 Conexiones de reloj

Cada entrada de reloj conecta directamente con un *buffer* de entrada global. Como se muestra Tabla 2-13 en la cada entrada se tambien se puede conectar opcionalmente a un DCM asociado.

Entrada de Reloj	Pin FPGA	I/O Bank	Global Buffer	DCM Asociado	LOC
CLK_50MHZ	E12	0	GCLK5	Superior Derecho	DCM_X2Y3
CLK_AUX	V12	2	GCLK2	Inferior Izquierdo	DCM_X2Y0
CLK_SMA	U12	2	GCLK3		

Tabla 2-13: Entradas de reloj y buffers globales y DCMs asociados.

El reloj oscilador de 50MHz tiene un ciclo de trabajo de 40% a 60% y una precisión de ±2500 Hz o ±50ppm. Además un reloj oscilador de 133MHz está instalado en el socket de reloj adicional.

2.5.3 DDR2 SDRAM

La plataforma Spartan-3A FPGA Starter Kit incluye una memoria DDR2 SDRAM Micron de 512Mbit (MT47H32M16) con una interfaz de datos de 16 bits, como se muestra en la Figura 2-19.

Todos los pines de la interfaz DDR2 SDRAM conectan con el Banco 3 de I/Os de la FPGA. El Banco 3 y la DDR2 SDRAM son ambos alimentados con 1,8V, administrados por un regulador LP3906 de la empresa National Semiconductor de fuente de entrada de 5V. Los 0,9V de voltaje de referencia, común para la FPGA y DDR2 SDRAM, también es entregado por este regulador.

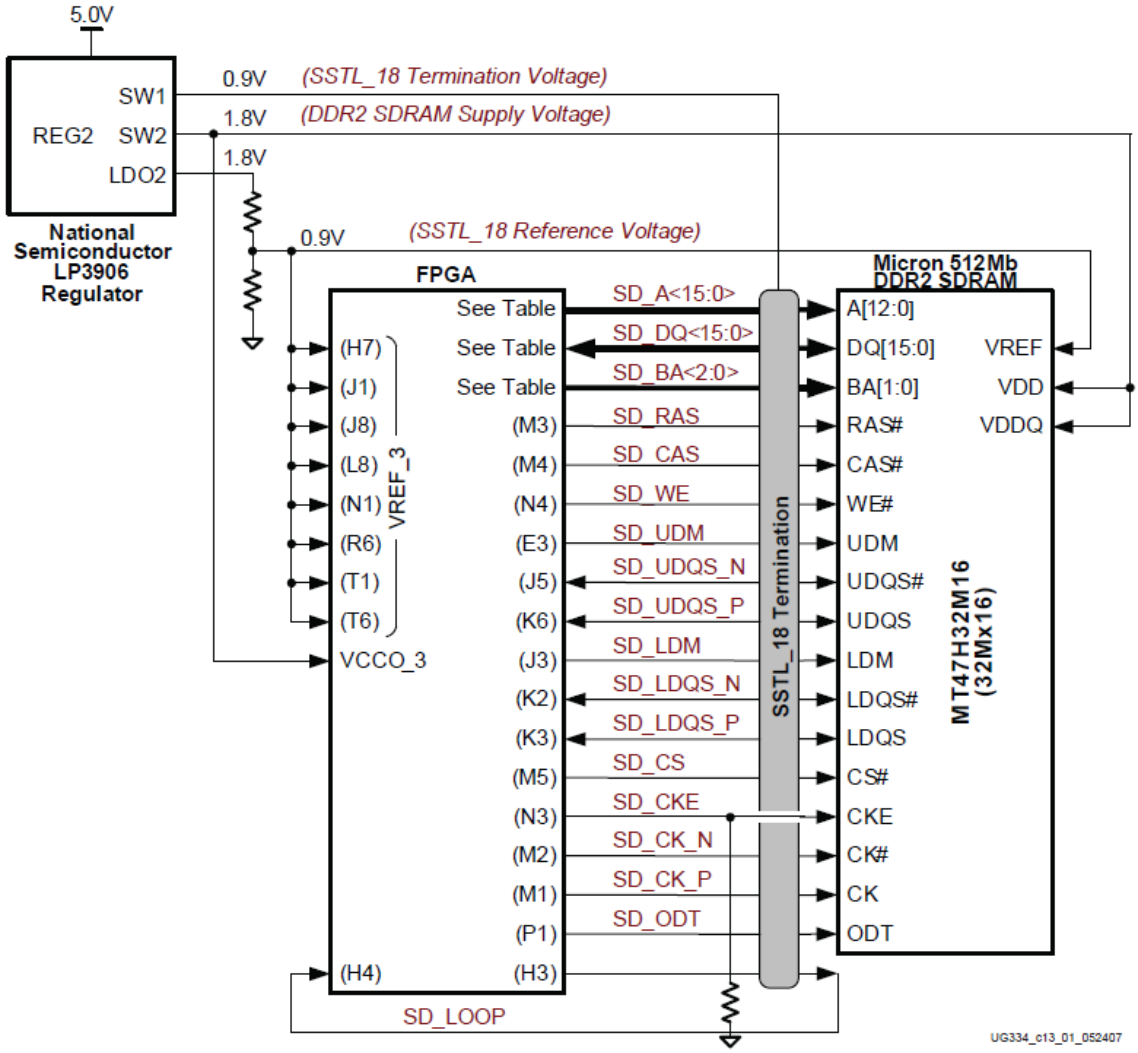


Figura 2-19: Interfaz de FPGA con DDR2 SDRAM.

2.5.3.1 Conexiones de DDR2 SDRAM

La siguiente tabla muestra las conexiones entre la FPGA y la DDR2 SDRAM.

Categoria	Nombre de señal DDR2 SDRAM	Pin FPGA	Función
Dirección	SD_A15	W3	No usadas en aparato DDR2 SDRAM, pero provistas para futuras mejoras
	SD_A14	V4	
	SD_A13	V3	

Tabla 2-14: Conexiones de FPGA a DDR2 SDRAM.

Categoria	Nombre de señal DDR2 SDRAM	Pin FPGA	Función
Dirección	SD_A12	Y2	Entradas de dirección.
	SD_A11	V1	
	SD_A10	T3	
	SD_A9	W2	
	SD_A8	W1	
	SD_A7	Y1	
	SD_A6	U1	
	SD_A5	U4	
	SD_A4	U2	
	SD_A3	U3	
	SD_A2	R1	
	SD_A1	T4	
	SD_A0	R2	
Datos	SD_DQ15	F3	Entrada/Salida de datos.
	SD_DQ14	G3	
	SD_DQ13	F1	
	SD_DQ12	H5	
	SD_DQ11	H6	
	SD_DQ10	G1	
	SD_DQ9	G4	
	SD_DQ8	F2	
	SD_DQ7	H2	
	SD_DQ6	K4	
	SD_DQ5	L1	
	SD_DQ4	L5	
	SD_DQ3	L3	
	SD_DQ2	K1	
	SD_DQ1	K5	
	SD_DQ0	H1	
Control	SD_BA2	P5	Entradas de dirección de banco.
	SD_BA1	R3	
	SD_BA0	P3	
	SD_RAS	M3	Entradas de comando.
	SD_CAS	M4	
	SD_WE	N4	
	SD_CK_N	M2	Entradas diferenciales de reloj.
	SD_CK_P	M1	
	SD_CKE	N3	Entrada activa en alto habilitadora de reloj.
	SD_CS	M5	Entrada activa en bajo seleccionadora de chip.
	SD_UDM	E3	Mascara de datos. Mascara superior e inferior.
	SD_LDM	J3	
	SD_UDQS_N	J5	*Strobe* diferencial de datos superiores.
	SD_UDQS_P	K6	
	SD_LDQS_N	K2	*Strobe* diferencial de datos inferiores.
	SD_LDQS_P	K3	
Misceláneo	SD_LOOP_IN	H4	Bucle de auto calibración de IOs. La dirección puede ser revertida si es más conveniente para la aplicación FPGA.
	SD_LOOP_OUT	H3	
	SD_ODT	P1	Control de terminación en el aparato de DDR2 SDRAM

Tabla 2-14: Conexiones de FPGA a DDR2 SDRAM (continuación).

2.5.3.2 Conector Hirose FX2 de 100-pines

Este conector es un Hirose FX2-100P-1.27DS *header* de 1.27mm de *pitch.* En la Figura 2-20, se muestra la interfaz entre la FPGA y el conector FX2.

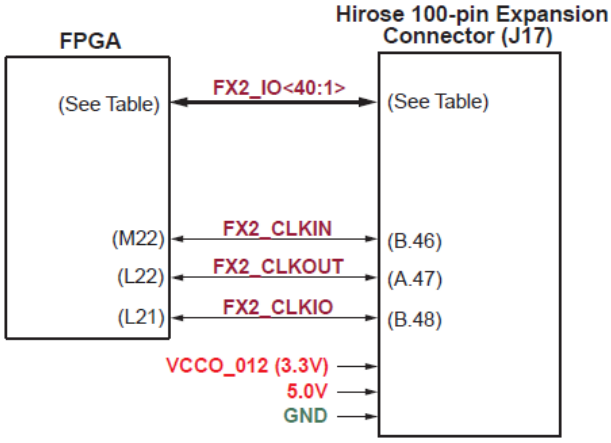


Figura 2-20: Conexiones de FPGA a conector Hirose.

La plataforma Starter Kit provee de energía al conector FX2 y a cualquier tarjeta conectada a está, a través de dos fuentes. La fuente de 5.0V provee de voltaje a cualquier lógica de 5V o reguladores de la plataforma conectada. Una fuente separada provee el mismo voltaje que los aplicados a los bancos I/O 0, 1, 2 de la FPGA. Esta fuente es de 3.3V por defecto. Todos los I/Os de la FPGA de la interfaz con el conector FX2 son de los bancos 0 y 1.

Nombre de Señal	Pin de FPGA	Conector FX2		Pin de FPGA	Nombre de Señal
		A (superior)	B (inferior)		
Fuente a bancos de FPGA	VCCO_012	1	1		Blindaje
	VCCO_012	2	2		GND
TMS_B		3	3		TDO_XC2C
JTSEL		4	4		TCK_B
TDO_FX2		5	5	GND	GND
FX2_IO1	A13	6	6	GND	GND
FX2_IO2	B13	7	7	GND	GND
FX2_IO3	A14	8	8	GND	GND
FX2_IO4	B15	9	9	GND	GND
FX2_IO5	A15	10	10	GND	GND
FX2_IO6	A16	11	11	GND	GND
FX2_IO7	A17	12	12	GND	GND
FX2_IO8	B17	13	13	GND	GND

Tabla 2-15: Pines de conector Hirose FX2 y conexiones con FPGA.

Nombre de Señal	Pin de FPGA			Pin de FPGA	Nombre de Señal
		A (superior)	B (inferior)		
FX2_IO9	A18	14	14	GND	GND
FX2_IO10	C18	15	15	GND	GND
FX2_IO11	A19	16	16	GND	GND
FX2_IO12	A19	17	17	GND	GND
FX2_IO13	B19	18	18	GND	GND
FX2_IO14	A20	19	19	GND	GND
FX2_IO15	B20	20	20	GND	GND
FX2_IO16	C19	21	21	GND	GND
FX2_IO17	D18	22	22	GND	GND
FX2_IO18	E17	23	23	GND	GND
FX2_IO19	D20	24	24	GND	GND
FX2_IO20	D21	25	25	GND	GND
FX2_IO21	D22	26	26	GND	GND
FX2_IO22	E22	27	27	GND	GND
FX2_IO23	F18	28	28	GND	GND
FX2_IO24	F19	29	29	GND	GND
FX2_IO25	F20	30	30	GND	GND
FX2_IO26	E20	31	31	GND	GND
FX2_IO27	G20	32	32	GND	GND
FX2_IO28	G19	33	33	GND	GND
FX2_IO29	H19	34	34	GND	GND
FX2_IO30	J18	35	35	GND	GND
FX2_IO31	K18	36	36	GND	GND
FX2_IO32	K17	37	37	GND	GND
FX2_IO33	K19	38	38	GND	GND
FX2_IO34	K20	39	39	GND	GND
FX2_IO35	L19	40	40	GND	GND
FX2_IO36	L18	41	41	GND	GND
FX2_IO37	M20	42	42	GND	GND
FX2_IO38	M18	43	43	GND	GND
FX2_IO39	L20	44	44	GND	GND
FX2_IO40	P20	45	45	GND	GND
GND	GND	46	46	M22	FX2_CLKIN
FX2_CLKOUT		47	47	GND	GND
GND	GND	48	48	L21	FX2_CLKIO
5.0V		49	49		5.0V
5.0V		50	50		Blindaje

Tabla 2-15: Pines de conector Hirose FX2 y conexiones con FPGA (continuación).

2.6 Modulo de Propiedad Intelectual (Intellectual Property, IP), Multi-Port Memory Controller (MPMC)[4]

MPMC es un controlador de memoria completamente parametrizable que soporta memorias SDRAM/DDR/DDR2. MPMC provee acceso a memoria por uno a ocho puertos, donde cada puerto puede elegirse de un conjunto de módulos de interfaz personal (*Personality Interface Modules*, PIMs). Adicionalmente soporta las opciones de código de corrección de errores (*Error Correcting Code*) y monitoreo de rendimiento (*Performance Monitoring*, PM).

En las siguientes sub-secciones se muestran características y elementos relevantes del modulo MPMC, las cuales son considerados en el proyecto.

2.6.1 Codificación de Dirección

MPMC no realiza validación del rango de direcciones, por lo que responde a todas las direcciones, a causa de esto es responsabilidad del PIM de asegurarse de entregar direcciones validas a través de la interface NPI.

Los siguientes parámetros son usados para la codificación de las direcciones:

- C_MEM_DATA_WIDTH
- C_MEM_PART_NUM_COL_BITS
- C_MEM_PART_NUM_ROW_BITS
- C_MEM_PART_NUM_BANK_BITS
- C_MEM_NUM_RANKS
- C_MEM_NUM_DIMMS

Con estos parámetros son calculadas las compensaciones (*offset)* de las direcciones como se indica a continuación:

- col_addr_startbit = log2(C_MEM_DATA_WIDTH/8)
- row_addr_startbit = col_addr_startbit + C_MEM_PART_NUM_COL_BITS
- bank_addr_startbit = row_addr_startbit + C_MEM_PART_NUM_ROW_BITS
- rank_addr_startbit = bank_addr_startbit + C_MEM_PART_NUM_BANK_BITS

[4] Extractos de documento *Multi-Port Memory Controller (MPMC) Product Specification*[8]

- dimm_addr_startbit = rank_addr_startbit + C_MEM_NUM_RANKS
- total_addr_startbit = dimm_addr_startbit + C_MEM_NUM_DIMMS

En la Tabla 2-16 se muestra la correspondencia entre los bits de dirección y el tipo de dirección que representan.

Tipo de dirección de memoria	Corresponde a :
Columna	PIM<Port_Num>_Addr[row_addr_startbit : col_addr_startbit]
Fila	PIM<Port_Num>_Addr[bank_addr_startbit-1: row_addr_startbit]
Banco	PIM<Port_Num>_Addr[rank_addr_startbit-1: bank_addr_startbit]
Rango	PIM<Port_Num>_Addr[dimm_addr_startbit-1:rank_addr_startbit]
DIMM	PIM<Port_Num>_Addr[total_addr_startbit-1:dimm_addr_startbit]
Espacio total de memoria	PIM<Port_Num>_Addr[total_addr_startbit-1:0]

Tabla 2-16: Tipo de dirección y parámetros correspondientes.

2.6.2 Interface de puerto nativo (Native Port Interface, NPI) PIM

Este PIM permite la conexión del MPMC con módulos creados por el usuario. Entre las características de este PIM se encuentran:

- Ofrece una simple interfaz a memoria que puede ser fácilmente adaptada a casi cualquier protocolo.
- Provee señales de dirección, datos y control, para generar peticiones de escritura y lectura a la memoria.
- Configuración de FIFO de escritura, lectura o ambos.
- Permite empujar y tirar simultáneamente de las FIFOs del puerto.
- Tiene un ancho de datos configurable a 32 y 64 bits.
- Cuando se usa 32 bits NPI, MPMC permite los siguientes tamaños de transferencia: byte, media palabra, palabra, *cacheline* de 4 palabras, *cacheline* de 8 palabras, *burst* de 16 palabras, *burst* de 32 palabras, y *burst* de 64 palabras.
- Cuando se usa 64 bits NPI, MPMC permite los siguientes tamaños de transferencia: byte, media palabra, palabra, doble palabra, *cacheline* de 4 palabras, *cacheline* de 8 palabras, *burst* de 16 palabras, *burst* de 32 palabras, y *burst* de 64 palabras.

2.6.2.1 Señales de entrada y salida de PIM NPI

En la Tabla 2-17 se muestran las señales de entrada y salida de un PIM NPI.

Nombre de Señal	Dirección	Estado Inicial	Descripción
PIM_<Port_Num>_Addr	Entrada	x	Indica la dirección de partida de una petición en particular.
PIM_<Port_Num>_AddrReq	Entrada	x	Esta señal de activación en alto indica que el NPI está listo para que MPMC arbitre un petición de dirección
PIM_<Port_Num>_RNW	Entrada	x	Read/Not Write: 0: Petición de escritura. 1: Petición de lectura.
PIM_<Port_Num>_Size	Entrada	x	Indica el tipo de transferencia de la petición: • 0x0 = Transferencia de palabra (solo NPI 32 bits) • 0x0 = Transferencia de dos palabras (solo NPI 64 bits) • 0x1 = Transferencia *cache-line* de 4 palabras • 0x2 = Transferencia *cache-line* de 8 palabras • 0x3 = Transferencia *burst* de 16 palabras • 0x4 = Transferencia *burst* de 32 palabras • 0x5 = Transferencia *burst* de 64 palabras
PIM_<Port_Num>_RdModWr	Entrada	x	Esta señal de activación en alto indica que si es una petición de escritura, MPMC deberia hacer un *read/modify/write*. Solo es válida si existe soporte ECC.
PIM_<Port_Num>_InitDone	Salida	0	1 indica que la inicialización esta completa y que las FIFOs esta disponibles para usar.
PIM_<Port_Num>_AddrAck	Salida	0	Esta señal de activación en alto indica que MPMC ha empezado el arbitraje para la petición de dirección.
PIM_<Port_Num>_WrFIFO_Data	Entrada	x	Información a ser empujada en el FIFO de escritura del MPMC.
PIM_<Port_Num>_WrFIFO_BE	Entrada	x	Indica cuales bytes de *PIM_<Port_Num>_WrFIFO_Data* debe escribir.
PIM_<Port_Num>_WrFIFO_Push	Entrada	x	Esta señal de activación en alto indica que se empuja *PIM_<Port_Num>_WrFIFO_Data* a la FIFO de escritura.
PIM_<Port_Num>_WrFIFO_Flush	Entrada	x	Reservada. Manejar con 0.
PIM_<Port_Num>_WrFIFO_Empty	Salida	1	Esta señal de activación en alto indica que existen menos de C_MEM_DATA_WIDTH bits de datos en la FIFO de escritura.
PIM_<Port_Num>_WrFIFO_AlmostFull	Salida	0	Esta señal de activación en alto indica que *PIM_<Port_Num>_WrFIFO_Push* no puede activarse en el siguiente ciclo de reloj.
PIM_<Port_Num>_RdFIFO_Pop	Entrada	x	Esta señal de activación en alto indica la FIFO de lectura debe entregar el siguiente valor de *PIM_<Port_Num>_RdFIFO_Data*.
PIM_<Port_Num>_RdFIFO_Flush	Entrada	x	Esta señal de activación en alto indica que las banderas de la FIFO de lectura deben ser reiniciadas.
PIM_<Port_Num>_RdFIFO_Data	Salida	0	Datos a ser tirado (Poped out) de la FIFO de lectura del MPMC.
PIM_<Port_Num>_RdFIFO_RdWdAddr	Salida	0	Indica la palabra que *PIM_<Port_Num>_RdFIFO_Data* corresponde de la transferencia de *cacheline*.
PIM_<Port_Num>_RdFIFO_Empty	Salida	1	Cuando esta señal es 0, indica que no existe suficientes datos la FIFO de lectura para leer.
PIM_<Port_Num>_RdFIFO_Latency	Salida	0,1,2	Indica el numero de ciclos de tiempo desde que *PIM_<Port_Num>_RdFIFO_Pop* es activado y/o *PIM_<Port_Num>_RdFIFO_Empty* es desactivado y la señales *PIM_<Port_Num>_RdFIFO_Data* y *PIM_<Port_Num>_RdFIFO_RdWdAddr* son validas.

Tabla 2-17: Señales de entrada y salida de PIM NPI.

2.6.2.2 Diagramas de tiempo de NPI PIM

Los siguientes diagramas de tiempo ilustran la funcionalidad del PIM en su configuración de 32 bits. Para facilidad de lectura se ha omitido el prefijo PIM_<Port_Num>_ de los nombres de las señales.

En la Figura 2-21 se muestra una transferencia de escritura *cacheline* de 8 palabras, esta parte con la activación de la señal WrFIFO_Push, la que empuja los datos de WrFIFO_Data a la fifo de escritura. Cuando se está empujando el octavo dato, se activa la señal AddrReq con la cual se inicia la petición de dirección. Esta señal debe mantenerse activada hasta que la señal AddrAck se active. Las señales Addr, RNW, Size y RdModWr deben mantenerse fijas mientras AddrReq este activa.

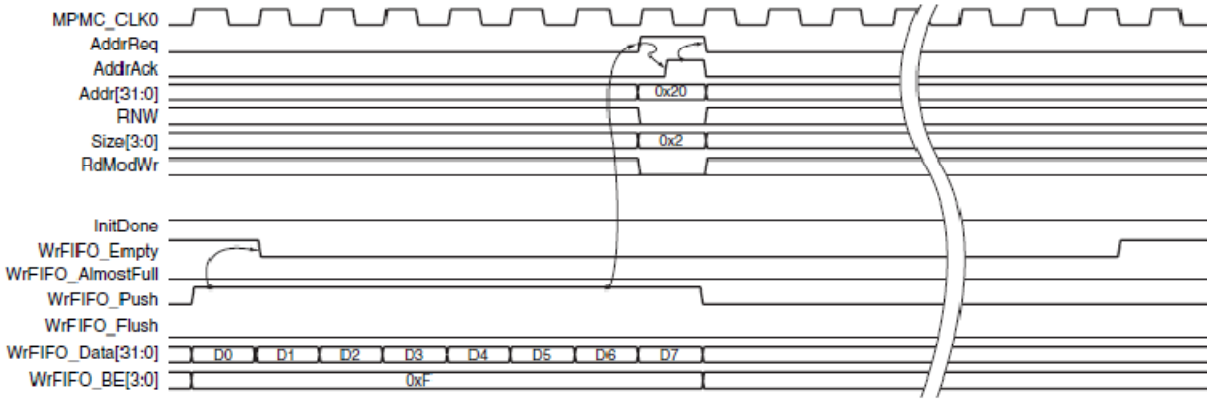


Figura 2-21: Transferencia de escritura cacheline de 8 palabras en NPI de 32 Bits.

En la Figura 2-22 se muestra una transferencia de lectura *cacheline* de 8 palabras, esta parte con la activación de la señal AddrReq, la cual debe mantenerse activa hasta la activación de AddrAck. Las señales Addr, RNW, Size y RdModWr deben mantenerse fijas mientras AddrReq esté activa. RdFIFO_Pop, no puede activarse hasta que RdFIFO_Empty se desactive, una vez desactivada RdFIFO_Pop puede mantenerse activa hasta vaciar la FIFO de escritura. Los buses de datos RdFIFO_Data y RdFIFO_RdWdAddr tendrán datos validos en una cantidad de ciclos de reloj equivalente a lo mostrado por RdFIFO_Latency.

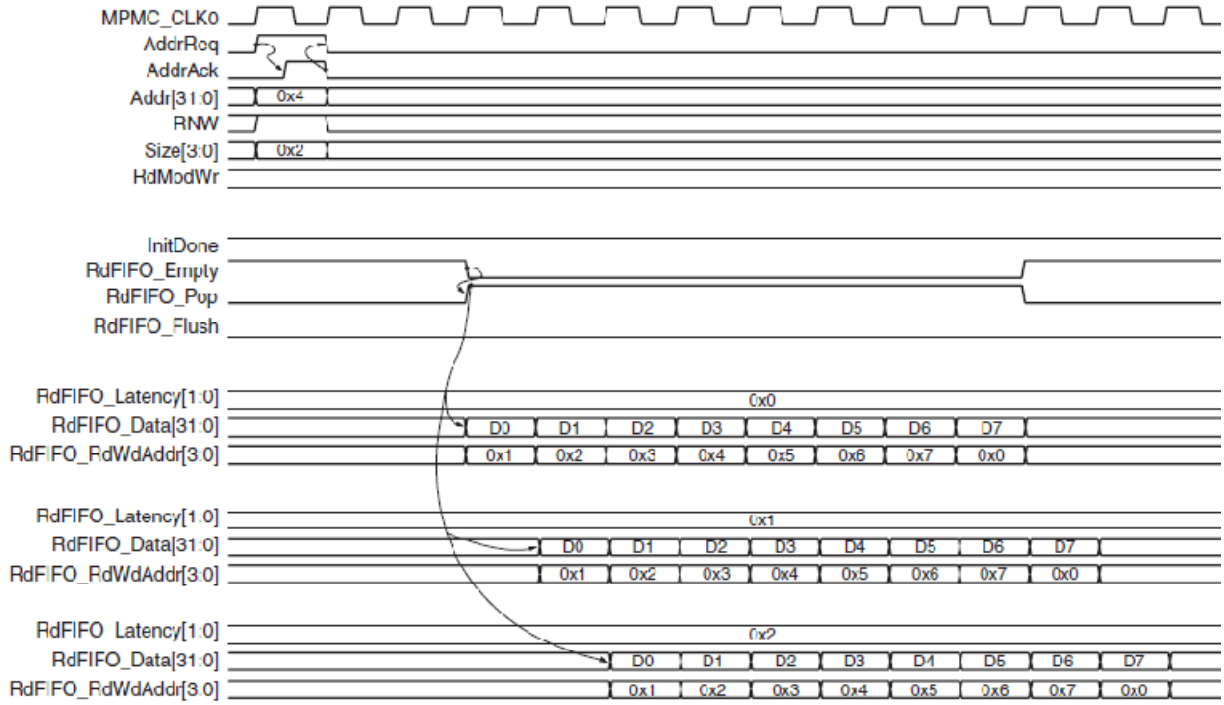


Figura 2-22: Transferencia de lectura cacheline de 8 palabras en NPI de 32 Bits.

2.6.3 Rangos de frecuencia en operación por dispositivo

En la Tabla 2-18 los rangos de frecuencia en operación para el modulo MPMC para una variedad de familias de dispositivos FPGA y grados de velocidad.

Familia FPGA	Rango Objetivo FMAX (MHz)
Spartan-3, -4 Grado de Velocidad	125-133
Spartan-3, -5 Grado de Velocidad	133-166
Virtex 4, -10 Grado de Velocidad	165-185
Virtex 4, -11 Grado de Velocidad	185-205
Virtex 4, -12 Grado de Velocidad	200-225
Virtex 5, -1 Grado de Velocidad	175-200
Virtex 5, -2 Grado de Velocidad	185-220
Virtex 5, -3 Grado de Velocidad	210-250

Tabla 2-18: Rangos de frecuencia en operación por dispositivo.

3 IMPLEMENTACIÓN

3.1 Descripción funcional de solución

El primer paso para la implementación del sistema es desarrollar un contextualización del tema, describiendo las funcionalidades que debe poseer el producto y las restricciones que se deben tener en cuenta para su implementación. Sobre este tema, se desea realizar la implementación de un sistema multiviewer, el cual como ya se explicó anteriormente permite la visualización de varias entradas de video, en un único monitor o pantalla.

En un primer acercamiento a la solución se infiere la necesidad de reducir el tamaño de las imágenes de entradas y un adecuado posicionamiento de cada una de ellas para su visualización.

Una segunda etapa en el planteamiento de la solución, es definir que este dispositivo tenga la característica de poder manejar distintos estándares de transmisión de video, esto debido a que una limitación en el tipo de entrada reduciría los posibles usuarios finales del aparato. La característica antes planteada define dos problemas en la implementación del sistema, el primero corresponde a una recepción de datos que soporte los estándares a los cuales se le desea dar soporte. Una segunda problemática es el tratamiento que se le debe dar a estos estándares para poder obtener el resultado final esperado. Como se explicó con anterioridad los estándares se diferencian en diversos aspectos, los cuales influyen en el resultado final o en la complejidad del dispositivo. El primer aspecto a tener en consideración es la codificación del color, ya que si no son llevadas todas las entradas a una única codificación se puede tener que pixeles de un mismo color que están en diferentes entradas tengan colores diferentes en la salida. Un segundo aspecto, es el entrelazado de la imagen, el que dificulta el procesamiento de ésta, por esta razón es recomendable llevar toda entrada de tipo entrelazada a un tipo progresivo. Por último hay que tener presente las diferencias de las tasas de refresco de las entradas y la salida, debido a que una mala sincronización entre estas podría llevar a tener resultados indeseables en la señal final.

Finalmente se requiere transmitir la señal de salida para que pueda ser visualizada en un monitor.

Todo lo anterior se puede llevar al diagrama de bloques que se muestra en la Figura 3-1, el cual ejemplifica los módulos que debe tener el proyecto para obtener el resultado esperado.

Figura 3-1: Diagrama de bloques de implementación de un multiviewer.

3.2 Diseño conceptual

3.2.1 Módulos de recepción, transmisión y decodificación de señales

Esta parte del *multiviewer* presenta una dificultad superior a la requerida en esta memoria, por lo que se plantea su implementación, con la utilización de dispositivos del mercado, que permitan realizar la recepción, transmisión y decodificación de diversos estándares de video.

Los requerimientos que se le pide al dispositivo buscado es el manejo de señales en *Standard, Enhanced y High Definition* y la obtención de un único estándar. Estas cualidades solo fueron encontradas en el *decoder* ADV7401 de la marca Analog Devices y el *encoder* ADV7391 de la misma marca.

El ADV7401 es un *decoder* de múltiples formatos de video y un digitalizador de gráficas. Este soporta la conversión de los estándares PAL, NTSC y SECAM en los formatos *composite* o S-video al formato digital ITU-R BT.656. Además de lo anterior soporta la decodificación de las señales de video por componentes RGB/YPrPb a una salida en *stream* digital de pixeles en formato YCbCr o RGB. Dentro del soporte al video por componentes el dispositivo incluye los estándares tales como 525i, 656i, 525p, 625p, 720p, 1080i, 1250i y muchos otros estándares HD y SMPTE. Como se menciono el ADV7401 soporta la digitalización de gráficas, la cual es capaz de digitalizar señales graficas RGB de tasas desde VGA hasta SXGA y convertirlas en un stream digital de pixel RGB o YCbCr.

El ADV7391 es un *encoder* de video digital a análogo que tiene soporte para salidas análogas *composite* (CVBS), S-Video (YC) o por componente en los formatos de video de definición estándar (SD) y alta definición (HD). Además posee una entrada de video de 8-bit que soporta SD con una interfaz SDR (*Single Data Rate*) y HD con una interfaz DDR (*Dual Data Rate*). Otras características de este dispositivo son el soporte de códigos de sincronización embebidos EAV/SAV, señales externas de sincronización y los protocolos de comunicación I2C y SPI. En la Tabla 3-1 se señalan los estándares soportados por el ADV7391.

Resolución	I/P	Tasa de Refresco (Hz)	Entrada de Reloj (MHz)	Estándar
720x240	P	59,94	27	
720x288	P	50	27	
720x480	I	29,97	27	ITU-R BT.601/656
720x576	I	25	27	ITU-R BT.601/656
720x480	I	29,97	24,54	NTSC Square Pixel
720x576	I	25	29,5	PAL Square Pixel
720x483	P	59,94	27	SMPTE 293M
720x483	P	59,94	27	BTA T-1004
720x483	P	59,94	27	ITU-R BT.1358
720x576	P	50	27	ITU-R BT.1358
720x483	P	59,94	27	ITU-R BT.1362
720x576	P	50	27	ITU-R BT.1362
1920x1935	I	30	74,25	SMPTE 240M
1920x1935	I	29,97	74,1758	SMPTE 240M
1280x720	P	60;50;30;25;24	74,25	SMPTE 296M
1280x720	P	23,97;59,94;29,97	74,1758	SMPTE 296M
1920x1080	I	30;25	74,25	SMPTE 274M
1920x1080	I	29,97	74,1758	SMPTE 274M
1920x1080	P	30;25;24	74,25	SMPTE 274M
1920x1080	P	23,98;29,97	74,1758	SMPTE 274M
1920x1080	P	24	74,25	ITU-R BT.709.5

Tabla 3-1: Estándares soportados por ADV7391.

Los dos dispositivos antes mencionados son la mejor opción para la implementación de un sistema *multiviewer* con soporte para HD, pero por razones de disponibilidad en el mercado, se debió buscar un remplazo para el dispositivo ADV7401. No se logró encontrar un remplazo que cumpliera con las características deseadas, por lo que se decidió buscar dispositivos que tuvieran soporte únicamente para SD. Se encontraron dos alternativas, el dispositivo TVP5154 de Texas Instrument y el ADV7184 de Analog Devices. Entre estos dos dispositivos el TVP5154 es el que tiene las mejores prestaciones al tener cuatro canales *decoder*, esto permitiría un ahorro en precio y espacio utilizado, pero se ha tomado la decisión de utilizar la solución de Analog Devices, debido a que este componente tiene una mayor similitud con el componente inicialmente elegido por lo que en el futuro permitirá una transición más fácil para obtener soporte para HD.

3.2.2 Modulo de Conversión a Escaneo Progresivo

Existen diversos algoritmos que permiten generar video progresivo a partir de uno entrelazado. A continuación se explica en qué consisten estos métodos y como son sus resultados usando el ejemplo de la Figura 3-2, el cual corresponde a un circulo en movimiento que es capturado de forma entrelazada.

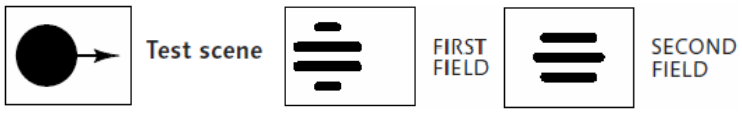


Figura 3-2: Ejemplo video entrelazado.

3.2.2.1 Duplicación de escaneo de línea (scanline duplication)

Este algoritmo como su nombre lo dice consiste en tomar uno de los campos de la imagen y repetir cada una de sus líneas, mientras la información del otro campo es descartada para mantener su tasa de refresco. Este método reproducirá un elemento estacionario con a lo más la mitad de su potencial resolución vertical. Además la repetición de líneas introduce un aspecto de bloques en la imagen y un aparente cambio descendente de una fila de la imagen. Este efecto es ilustrado en la Figura 3-3.

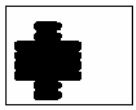


Figura 3-3: Ejemplo duplicación de escaneo de línea.

3.2.2.2 Interpolación de escaneo de línea (scanline interpolation)

Los cuadros de salida son producidos llenando las líneas faltantes del campo actual con la interpolación lineal de las líneas sobre y bajo ella. Debido a que solo el campo recibido en el momento es usado, para mantener la tasa de refresco de la entrada es necesario descartar la mitad de los campos. Este procedimiento soluciona la desventaja del aspecto de bloques del método de duplicación, pero no compensa la pérdida de resolución vertical.

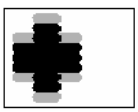


Figura 3-4: Ejemplo interpolación de escaneo de línea.

3.2.2.3 Tejido (Weave)

Los cuadros de salida son creados completando las líneas faltantes del campo actual con las líneas del campo previo. Este método da buenos resultados para escenas estáticas, o que contienen elementos con movimientos lentos. Sin embargo, el campo previo está retrasado en medio cuadro (típicamente 1/60s o 1/50s). Si la escena contiene un elemento con mucho movimiento, el objeto sufrirá *field tearing*, o sea será reproducido con bordes dentados, el efecto es mostrado en la Figura 3-5.

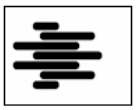


Figura 3-5: Ejemplo Algoritmo Weave.

3.2.2.4 Movimiento Adaptivo (Motion Adaptive)

Este método consiste en detectar si el pixel resultante pertenece a un elemento en movimiento o a un elemento estático. Si el elemento probablemente esta en movimiento, se usa para determinar el pixel el procedimiento de interpolación. Si el elemento probablemente es estacionario el pixel se obtiene utilizando *weave*. El movimiento es detectado comparando un campo con un campo previo. Idealmente, un campo similar debería ser usado, por ejemplo si el movimiento es estimado para el campo 1, entonces el campo 1 previo debería ser usado como punto de referencia. Por lo explicado anteriormente esta metodología requiere el almacenamiento de un cuadro. En comparaciones a los métodos anteriores, este procedimiento genera los mejores resultados.

3.2.2.5 Elección de método de conversión entrelazado a progresivo

La elección del método a utilizar se basa en dos parámetros, la calidad de la imagen de salida y la cantidad de recursos necesarios para implementarlo. Como ya se mencionó en la sección anterior, el método a elegir rigiéndose por el primer parámetro es el movimiento adaptivo, pero este tiene la desventaja de ser el que requiere la mayor cantidad de recursos. Basándose en el segundo parámetro el método de duplicación de líneas es el más recomendable en primera instancia, pero dado que a esté módulo le sucede el redimensionamiento como se muestra en la Figura 3-1, el método que requiere la menor cantidad de recursos pasa a ser la interpolación de escaneo de línea, debido a que este módulo puede usar la interpolación que realiza el bloque de redimensionamiento para generar la señal progresiva. Finalmente, debido a la plataforma final en la que se implementará el proyecto, de la cual se darán detalles más adelante, se selecciona el método de interpolación, ya que los recursos de memoria son limitados.

3.2.3 Modulo de redimensionamiento.

El redimensionamiento de video es el proceso de convertir una imagen de entrada de dimensiones X_{in} pixels por Y_{in} líneas en una imagen de salida de dimensiones X_{out} pixels por Y_{out} líneas.

El redimensionamiento de video tiene la forma de la operación de un filtro 2D, el cual puede ser aproximado por la Ecuación 3-1.

$$O(x,y) = \sum_{i=1}^{HTaps} \sum_{j=1}^{VTaps} F\left(x - \left(\frac{HTaps}{2}\right) + i, y - \left(\frac{VTaps}{2}\right) + j\right) \times Coef_{ij}$$

Ecuación 3-1.

En esta ecuación, x e y son posiciones discretas de una rejilla de muestras; *O(x,y)* es la intensidad del pixel de salida que es generado en la posición *(x,y)*, *F(x,y)* es la intensidad del pixel de entrada que es usado por el re-dimensionado; $Coef_{ij}$ es un arreglo de coeficientes; y *HTaps, VTaps* son el número de *taps* horizontales y verticales en el filtro.

Los coeficientes de la Ecuación 3-1 representan el peso aplicado al conjunto elegido de muestras de entrada para representar el pixel de salida, según la razón de redimensionamiento.

Una implementación directa de la ecuación sugiere un filtro con *VTaps x HTaps* multiplicaciones por el número de salidas requeridas. Sin embargo, se puede aproximar la salida del filtro bidimensional por la operación de dos etapas de filtros de una dimensión en cascada, una etapa de filtrado vertical y una etapa de filtrado horizontal. El concepto de cascada corresponde, a que el resultado de la primera etapa es entregado como entrada a la segunda etapa.

La etapa de filtrado vertical solo filtra en el dominio vertical, por cada incremento horizontal de la posición x en el escaneo del *raster*, lo que es representado por la Ecuación 3-2.

$$VO(x,y) = \sum_{j=1}^{VTaps} F\left(x, y - \left(\frac{VTaps}{2}\right) + j\right) \times Coef_j$$

Ecuación 3-2.

El resultado de la etapa anterior es entregado al filtro horizontal el cual es representado por la Ecuación 3-3.

$$O(x,y) = \sum_{j=1}^{HTaps} VO\left(x - \left(\frac{HTaps}{2}\right) + i, y\right) \times Coef_i$$

Ecuación 3-3.

Para expresar un pixel de salida en términos de pixeles de entrada, es necesario saber o estimar la posición del pixel de salida en relación al pixel de entrada más cercano cuando se superponen las rejillas de muestras de la entrada y salida en un espacio bidimensional equivalente. Con este conocimiento el algoritmo aproxima el valor del pixel de salida usando un filtro con coeficientes acordes. Los *taps* del filtro corresponden a puntos consecutivos de la imagen de entrada.

Como ejemplo, la Figura 3-6 muestra una rejilla de salida de 5x5 representada por "O" que está superpuesta sobre una rejilla de entrada de 6x6 representada por "X", ocupando un espacio común. En este caso, estimar para la posición de salida (1,1), muestra que el pixel de entrada y salida están en la misma posición, por lo que los pesos de los coeficientes deben reflejar el peso total en el pixel de entrada que está en la misma posición. En la posición de salida (2,2) existe una compensación (offset) del enrejado de entrada tanto en la dimensión vertical como la horizontal, por lo que los coeficientes elegidos deben reflejar esto, asignándole un mayor peso al coeficiente de la posición de entrada (2,2).

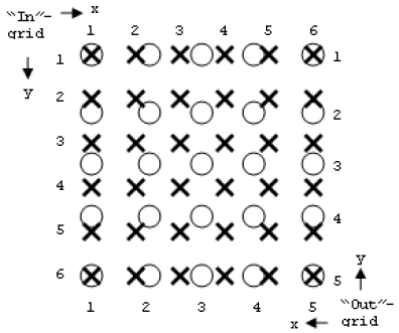


Figura 3-6: Ejemplo de superposición de rejilla de entrada y salida en un redimensionador.

El espacio entre dos pixeles de entradas consecutivos en cada dimensión está conceptualmente particionado en un número de cuencas o fases. La posición de cualquier pixel de salida arbitrario siempre cae en uno de estas cuencas, definiendo así la fase de los coeficientes usados. La arquitectura del filtro debería poder aceptar cualquiera de las diferentes fases de los coeficientes, cambiando de fase de muestra en muestra. En la Figura 3-7 se muestra el caso para una dimensión, donde se ilustra que los 5 pixeles de salida pueden tener de izquierda a derecha las fases 0, 1, 2, 3, 0.

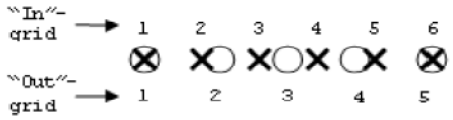


Figura 3-7: Superposición de rejillas de una dimensión.

3.2.4 Sincronización entradas y salida

La diferencia en tasas de muestro y refresco entre las entradas y la salida puede llevar a la obtención de errores, debido principalmente a la escritura y lectura simultánea de espacio de memoria. Para solucionar esto se deben usar espacios de memoria diferentes para la escritura y lectura. Finalmente el problema de sincronización se traduce en administrar los espacios de memoria de forma que se intercale la escritura y lectura en estos, asegurándose que el cuadro leído de la memoria sea el último completamente escrito en ella.

3.2.5 Elección de tecnología

Dada la característica del proyecto, de procesamiento digital de señales (DSP) a altas frecuencias se puede restringir la tecnología a usar, a dos tipos: FPGA y microprocesador DSP (uP DSP). La elección del tipo de tecnología se basó en el estudio realizado por Berkeley Design Technology, Inc. (BDTi) expuesto en la presentación "Comparing FPGAs and DSPs for High-Performance DSP Applications"[9], donde se comparan las tecnologías en base al costo por canal de dos plataformas por tecnología. Las plataformas que se comparan en el estudio son las FPGAs Virtex-4 SX25 de Xilinx y Stratix II 2S15 de Altera y los procesadores DSP TMS320C6410 de Texas Instrument y MSC8144 de Freescale, llegando a los resultados que se muestran en el gráfico de la Figura 3-8.

Las conclusiones obtenidas del estudio fueron que las FPGAs de alto desempeño sobrepasan a los procesadores DSP en ciertas tareas DSP que requieren una gran cantidad de cómputos y un alto grado de paralelización. Además son superiores en desempeño por dólar en estas tareas. Por otra parte los procesadores DSP tienen la ventaja en su infraestructura de desarrollo, tiempo al mercado y la familiaridad de los desarrolladores.

Finalmente la tecnología elegida fue la FPGA, debido a que las ventajas señaladas por el estudio son las que se requieren en el proyecto, dado que el multiviewer requiere el procesamiento de varias señales de video en paralelo.

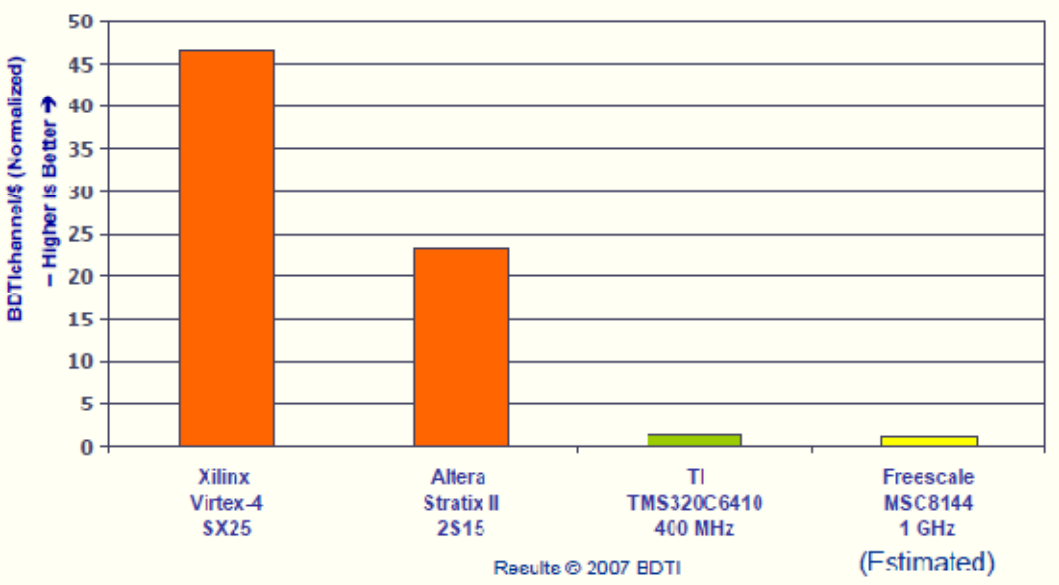


Figura 3-8: Resultados de estudio "Comparing FPGAs and DSPs for High-Performance DSP Applications".

3.3 Diseño de plataforma de pruebas

Este diseño se inicia con la elección de las opciones: realizar una plataforma final que contenga todos los elementos necesarios de un *multiviewer*, o una plataforma intermedia de menor valor que permita probar, las capacidades de los dispositivos incorporados y los diseños en HDL. Dado el carácter de ser un primer diseño, el tiempo con que se dispone y los recursos necesarios para la implementación, se elige la opción de partir con la construcción una plataforma intermedia y una vez que esta pase todas las pruebas necesarias para acreditar su funcionamiento óptimo, se pase al diseño y posterior construcción de una plataforma final.

Como se mencionó en la sección diseño conceptual, la plataforma deberá contar con un *enconder* para obtención de una salida y la demostración del dispositivo, una FPGA que realizará el procesamiento de las señales y al menos dos *decoder* que permitirán probar la recepción y sincronización de señales.

Para el procesamiento de señales se ha tomado la decisión de utilizar la plataforma de desarrollo Spartan-3A Starter Kit, la cual se tiene a disposición y que permitirá una disminución en los costos de la plataforma. Las características de esta plataforma que abalan su uso en este proyecto, son: posee una FPGA XC3S700A, la cual pertenece al rango medio alto de la familia Extended Spartan-3A en cuanto número de recursos lógicos; incorpora una memoria DDR2 SDRAM de 16 bits; y posee un puerto de conexión de 100 pines que permite su interacción con placas externas.

El diseño de los esquemáticos y PCB de la plataforma son desarrollados con el programa Eagle 5.4.0 Lite. Esta versión tiene la característica de ser gratis, pero la restricción de un tamaño máximo de 8x10 cm^2 y una cantidad máxima de 2 capas en el diseño del PCB. Por lo que se crea una restricción de espacio, la cual obliga a tener el mínimo de las prestaciones necesarias o lo que se traduce en: dos decoder conectados con un solo tipo de conector de video cada uno; un encoder también con un único conector de video; un conector Hirose FX2 de 100 pines para la conexión con la plataforma Spartan-3A Starter Kit; una sección de regulación de voltaje; y protecciones de voltaje para los pines de entrada y salida de los chips. Las secciones de regulación de voltaje y protección, fueron diseñadas basándose en la plataforma VDEC1 y el resto en las recomendaciones de las respectivas hojas de datos, obteniéndose los esquemáticos incorporados en el Anexo B.1.

El detalle de los componentes usados se muestra en la Tabla 3-2.

Componente	Fabricante	Descripción	Unidades Requeridas
MM3Z3V3CCT	Fairchild Semiconductor	Diodo Zener 200MW 3.3V	40
TSP79301	Texas Instrument	Regulador de Voltaje	3
FX2-100S-1.27DS(L)	Hirose	Conector Receptor 100 pines	1
RCJ-32265	CUI Inc	Conector RCA Jack rojo/verde/azul	3
CS10	Citizen	Cristal 28,63636 MHz 18pF SMD	2
ADV7184BSTZ	Analog Devices	Multiformat SDTV Video Decoder	2
ADV7391BCPZ	Analog Devices	10Bit SD/HD Video Encoder	1
MCR18EZHF33R0	Rohm Semiconductor	Res 33 Ohm 1/4W 1% 1206 SMD	50
CC1206JRNP09BN470	Yageo	Cap Ceramico 47pF 50V NP0 1206	10
ECJ-3YB0J226M	Panasonic - ECG	Cap 22uF 6.3V Ceramico X5R 1206	2
CC1206KKX7R7BB105	Yageo	Cap 1uF 16V Ceramico X7R 1206	20
CC1206KRX7R9BB104	Yageo	Cap 0,1uF 50V Ceramico X7R 1206	50

Tabla 3-2: Componentes usados en PCB de prueba.

El diseño del PCB se realiza, creando una librería que contiene los empaquetamientos de los dispositivos que no se encuentran definidos en las librerías que vienen por defecto en Eagle. Los elementos creados son todos menos las resistencias y capacitores, para los cuales se usan empaquetamiento SMD (*Surface Mount Device*) de tamaño 1206. En cuanto los elementos creados se siguen las especificaciones de cada uno de los datasheet, creando los objetos con las especificaciones que se muestran en el Anexo B.2.

Componente	Fabricante	Descripción	Unidades Requeridas
CC1206KRX7R9BB103	Yageo	Cap 10nF 50V Ceramico X7R 1206	40
12065C823KAT2A	AVX Corporation	Cap Ceramico 0,82 uF 10% 50V X7R 1206	10
ECJ-3VB1C154K	ECJ	Cap 0,15uF 16V Ceramico X7R 1206	3
CC1206KRX7R9BB123	Yageo	Cap 12nF 50V Ceramico X7R 1206	10
CC1206KRX7R9BB222	Yageo	Cap Ceramico 1nF 50V NP0 1206	10
CC1206JRNP09BN102	Yageo	Cap 2,2nF 50V Ceramico X7R 1206	10
MCR18EZHF75R0	Rohm Semiconductor	Res 75 Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF1000	Rohm Semiconductor	Res 100 Ohm 1/4W 1% 1206 SMD	10
MCR18EZHF1690	Rohm Semiconductor	Res 169 Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF5100	Rohm Semiconductor	Res 510 Ohm 1/4W 1% 1206 SMD	10
MCR18EZHF1691	Rohm Semiconductor	Res 1,69K Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF2201	Rohm Semiconductor	Res 2,2K Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF2701	Rohm Semiconductor	Res 2,7K Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF4701	Rohm Semiconductor	Res 4,7K Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF8201	Rohm Semiconductor	Res 8,2K Ohm 1/4W 1% 1206 SMD	10
MCR18EZPF1004	Rohm Semiconductor	Res 1M Ohm 1/4W 1% 1206 SMD	10
C3216X5R0J106M	TDK Corporation	Cap Ceramico 10uF 6,3V X5R 20% 1206	10
ECJ-3YB0J106K	Panasonic - ECG	Cap 10uF 6,3V Ceramico X5R 1206	1

Tabla 3-2: Componentes usados en PCB de prueba. (continuación)

Posteriormente con la información del esquemático y los diseños de los dispositivos se tiene que rutear las pistas que unen a estos dispositivos. Las consideraciones tomadas en el ruteo de la placa son las siguientes:

- Los componentes asociados a los chips grandes (decoders, encoder, reguladores de voltaje) deben estar cercanos a los respectivos chips.
- Las pistas de datos de los chips grandes deben tener largos similares considerando su punto de partida el pin del chip y el de término el pad del conector.
- Tierras separadas para los componentes análogos y digitales, las cuales son conectadas en un único punto a través de un inductor de ferrita.
- No colocar vias debajo de los componentes.

Siguiendo estas reglas se logró obtener el ruteo que se muestra en el ANEXO 0, y obtener los largos de pistas de datos que se muestran en la Tabla 3-3.

Señal	Largo (mm)	Señal	Largo (mm)
Salida de Video			
pr_out	28,43	pb_out	28,48
y_out	28,39	adv7391_p0	47,28
adv7391_p1	46,60	adv7391_p2	46,91
adv7391_p3	47,06	adv7391_p4	46,40
adv7391_p5	46,44	adv7391_p6	44,90
adv7391_p7	47,28	adv7391_clkin	46,71
Entrada de Video A			
Pr_a	43,91	Pb_a	43,21
Y_a	43,69	Adv7184a_p15	40,87
Adv7184a_p14	40,94	Adv7184a_p13	41,00
Adv7184a_p12	40,12	Adv7184a_p11	41,04
Adv7184a_p10	40,90	Adv7184a_p9	41,00
Adv7184a_p8	41,03	Adv7184a_llc1	40,51
Entrada de Video B			
Pr_b	27,06	Pb_b	26,83
Y_b	26,82	Adv7184b_p15	46,82
Adv7184b_p14	46,87	Adv7184b_p13	46,86
Adv7184b_p12	46,81	Adv7184b_p11	46,82
Adv7184b_p10	46,64	Adv7184b_p9	46,73
Adv7184b_p8	46,82	Adv7184b_llc1	40,51

Tabla 3-3: Largos de pistas de señales de datos en plataforma de prueba.

3.4 Diseño en HDL

El diseño en HDL corresponde a la implementación en la FPGA de los módulos: conversión a escaneo progresivo; redimensionamiento; sincronización de entradas y salida; y posicionamiento de señales, que fueron mencionados en la sección de diseño conceptual.

La primera consideración tomada en esta etapa del proyecto fue la utilización del IPcore de Xilinx, Multi-Port Memory Controller (MPMC), para el manejo de escritura y lectura de la memoria. El MPMC puede controlar hasta un máximo de 8 puertos de entrada/salida, lo que es un punto importante en la implementación del diseño, ya que restringe la utilización de los buffers de cuadro.

Teniendo en consideración restricción anterior, se plantea la estructura que se muestra en la Figura 3-9, donde cada módulo tiene una función específica, que será descrita en la sección 3.4.2.

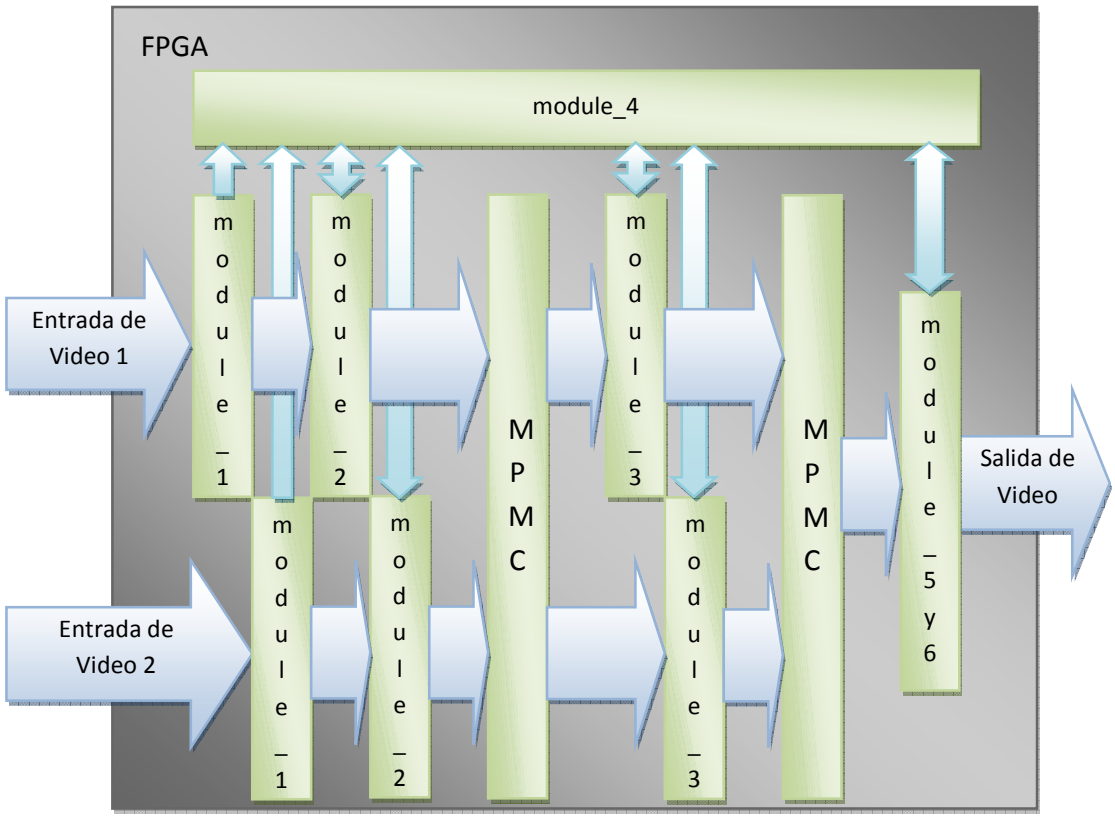


Figura 3-9: Diseño de sistema en HDL.

Como se puede apreciar en total existen 7 accesos a memoria que corresponden a las flechas que entran y salen del modulo MPMC. Estos accesos a memoria tienen una gran relevancia en el funcionamiento de la plataforma de prueba, debido a que cada uno consume una cierta cantidad de ancho de banda de la memoria, que de ser muy alto puede llevar a resultados indeseados.

3.4.1 Cálculo de ancho de banda y requerimiento de memoria del sistema planteado para plataforma de prueba

La plataforma Spartan-3A Starter Kit cuenta con una memoria DDR2 SDRAM MH47H32M16-3. Esta memoria tiene una configuración de 32 Meg x 16 x 4 banks y alcanza tasas de transferencia máximas que se muestran en la Tabla 3-4.

Data Rate (MTransiciones/s)				tRC
CL =3	CL = 4	CL = 5	CL = 6	(ns)
400	533	667	n/s	55

Tabla 3-4: Ancho de banda máximo y tiempo de refresco de memoria DDR2 SDRAM de plataforma Starter Kit.

A pesar de tener estas tasas de transferencia máxima, la memoria DDR2 de la plataforma está limitada por la frecuencia máxima a la que puede funcionar el módulo MPMC en la FPGA que trae la plataforma, cuyo valor es de 133MHz, lo cual se traduciría en una tasa máxima teórica de 266 MT/s, 532 Mbytes/s o 4256 Mbits/s.

La mayor parte del ancho de banda de la memoria estaría destinado a la escritura y lectura de los cuadros, por esta razón se calcula el uso de ancho de banda de los estándares que puede manejar la plataforma de pruebas y además el espacio utilizado en memoria por estos estándares, los cuales se muestran en la siguiente tabla, considerando una codificación YCbCr 4:2:2 de 8 bits.

Estándar	Pixeles activos por línea	Líneas activas por cuadro	Pixeles por cuadro	Tamaño buffer de cuadro (bytes)	Tasa de cuadro (Hz)	Ancho de banda (Mbit/s)
480i29.27	720	480	345.600	691.200	30	165,722
576i25	720	576	414.720	829.440	25	165,888
720p50	1280	720	921.600	1.843.200	50	737,28
1080i25	1920	1080	2.073.600	4.147.200	25	829,44

Tabla 3-5: Requerimiento de memoria para un frame y ancho de banda para estándares soportados por plataforma de prueba.

Para el cálculo total del ancho de banda utilizado por el sistema, se considera que cada proceso de escritura de module_2 y lectura de module_3 utiliza un ancho de banda equivalente al requerido por el estándar de la entrada de video respectiva. module_3 como se explicará más adelante corresponde al módulo de redimensionamiento, por esta razón su salida puede tener en el caso de máxima utilización de ancho de banda las mismas característica que la salida de video del sistema, por lo que en el cálculo se considera que cada proceso de escritura de module_3 y lectura de module_4, utiliza un ancho de banda equivalente al requerido por el estándar de la salida de video. Finalmente el cálculo del ancho de banda usado por el sistema puede ser realizado con la siguiente ecuación.

$$AnchoDeBandaTotal = 2 \cdot (AnchoDeBanda_{video0} + AnchoDeBanda_{video1} + AnchoDeBanda_{salida})$$

Ecuación 3-4: Calculo de ancho de banda del sistema.

Utilizando la ecuación anterior se calcula el ancho de banda de distintas combinaciones de estándares, para las entradas y la salida, lo cual se muestra en la Tabla 3-6. Además se calcula la utilización del ancho de banda de la DDR2 en la plataforma de pruebas.

Combinación	$Standard_{video0}$	$Standard_{video1}$	$Standard_{salida}$	Ancho de Banda total del sistema (Mbit/s)	Utilización de ancho de banda de memoria (%)
1	576i25	576i25	576i25	995,328	23,39
2	720p50 576i25	576i25 576i25	576i25 720p50	2138,112	50,24
3	576i25 720p50	720p50 720p50	720p50 576i25	3280,896	77,09
4	720p50	720p50	720p50	4423,68	103,93
5	576i25 576i25	1080i25 576i25	576i25 1080i25	2322,432	54,57
6	576i25 1080i25	1080i25 1080i25	1080i25 576i25	3649,536	85,75
7	1080i25	1080i25	1080i25	4976,64	116,93
8	720p50 576i25 720p50	1080i25 1080i25 576i25	576i25 720p50 1080i25	3465,216	81,42

Tabla 3-6: Utilización de ancho de plataforma de prueba Starter Kit para distintas combinaciones de entradas y salida.

Si se considera que el ancho de banda real de la memoria es un 80% del teórico, la plataforma Starter Kit soportaría las combinaciones: entradas SD con cualquier tipo de salida; entradas 720p50 con salida SD; una entrada SD y otra 720p50 con salida SD; y una entrada HD y otra SD con salida SD. En el caso de la plataforma de pruebas solo la primera combinación es soportada.

Considerando que la utilización máxima de memoria se produce con la combinación de entradas 576i25 y salida 1080i25, usando doble buffer cuadro en el almacenamiento, la memoria usada por esta combinación es de 19.906.560 bytes, lo que corresponde a un 30% del espacio total de la memoria de la plataforma Starter Kit. Por lo que la plataforma de pruebas no tiene ninguna complicación para soportar esta combinación.

3.4.2 Descripción de los módulos del sistema

3.4.2.1 Módulo module_1

Este módulo es el encargado de recopilar información sobre señal de video. En las Tabla 3-7 y Tabla 3-8 se muestra las señales de entrada y salida y los parámetros de este módulo.

Nombre de Señal	Dirección	Estado de Inicio	Descripción
clk	entrada	x	Reloj
rst	entrada	x	Reset
data_in	entrada	x	Bus de datos
data_out	salida	0	Bus de datos de entrada retrasado en un periodo de reloj
hs_n	salida	1	Señal de sincronización
vs_n	salida	1	Señal de sincronización
field	salida	1	Señal de sincronización
is_interlaced	salida	0	Si es 1 indica que la señal es entrelazada, 0 que es progresiva.
locked	salida	0	Si es 1 indica que la señal ha estado estable por 2 frames.
active_components_per_line	salida	0	Corresponde al número de componentes activo de la señal de video
field_0_active_lines_per_frame	salida	0	En caso de ser una señal entrelazada corresponde a la cantidad de líneas activas del field 0. En caso de ser progresiva corresponde a la cantidad líneas activas del frame
field_1_active_lines_per_frame	salida	0	En caso de ser una señal entrelazada corresponde a la cantidad de lineas activas del field 1. En caso de ser progresiva corresponde a la cantidad líneas activas del frame.

Tabla 3-7: Señales entrada y salida de module_1.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
DATA_WIDTH	8	8	Corresponde al ancho del bus de datos de entrada
VIDEO_DEFINITION	"SD"	"SD", "HD"	Permite determinar el tamaño de los contadores.

Tabla 3-8: Parámetros Globales de molule_1.

Este módulo tiene cuatro partes fundamentales en su funcionamiento. La primera de ellas es identificar los códigos de temporización que vienen embebidos en data_in, y con estos generar las señales de sincronismo acordes a la recomendación ITU-R BT.656. La implementación de esta parte es descrita asemejando el esquema resaltado en el recuadro rojo de la Figura 3-10. Esta sección del módulo parte identificando los códigos de temporización a través de la señal is_timing_signal, la cual se activa cuando data_in tiene todos sus bits en 1 y se desactiva cuando data_in es distinta de 0. La activación de is_timing_signal y data_in distinto de 0 corresponde al código XYZ, por lo que en ese instante de tiempo los flipflops hs_n_0, vs_n_0 y field_0, toman el valor correspondiente a los bits de información del código. Para que las señales de sincronización cambien cuando se termina el código de temporización se añaden flipflops en cascada. Dado que esto no sirve para le señal hs_n la cual debe pasar de 0 a 1, cuando se detecta el código de temporización. Se le incorpora esta característica en un flipflop distinto. Como se puede apreciar no se realiza verificación o corrección de errores al código, esto porque se supone que la señal data_in es poco propensa a tener errores, al ser generada en la misma PCB.

La segunda parte es determinar las características de la imagen en cuanto a dimensiones, esto se encuentra esquematizado en la Figura 3-10 en la parte resaltada con azul. El proceso se inicia con un contador de componentes el cual siempre está en funcionamiento y que se reinicia cuando termina el código de temporización SAV. Para obtener un valor fijo se usa un banco de flipflops, los cuales se activan cuando reinicia el contador de componentes, obteniendo así el número total de componentes por línea. El reinicio del contador de componentes además sirve para aumentar la cuenta de un contador de líneas el que se reinicia con la finalización del EAV de la primera línea activa en caso de ser video progresivo o solo en la primera línea activa del campo 0 en caso de ser entrelazado. La señal de reinicio del contador de líneas habilita un banco de flipflops, lo que almacenan el total de líneas por cuadro. Para obtener el resto de características se usa un método similar activando un banco de flipflops cuando ocurre algún evento. En el caso de los componentes activos por líneas, el evento es cuando todos los bits data_in son 1 y hs_n es 0. La señales field_0_active_lines_per_frame y field_1_active_lines_per_frame tienen un doble comportamiento dependiendo del tipo de video. En caso de ser progresivo estas dos señales son activadas por el cambio de vs_n de 0 a 1, por lo que tendrán el mismo valor el cual corresponde al número de líneas activas del cuadro. Cuando es entrelazado, el evento también depende del campo.

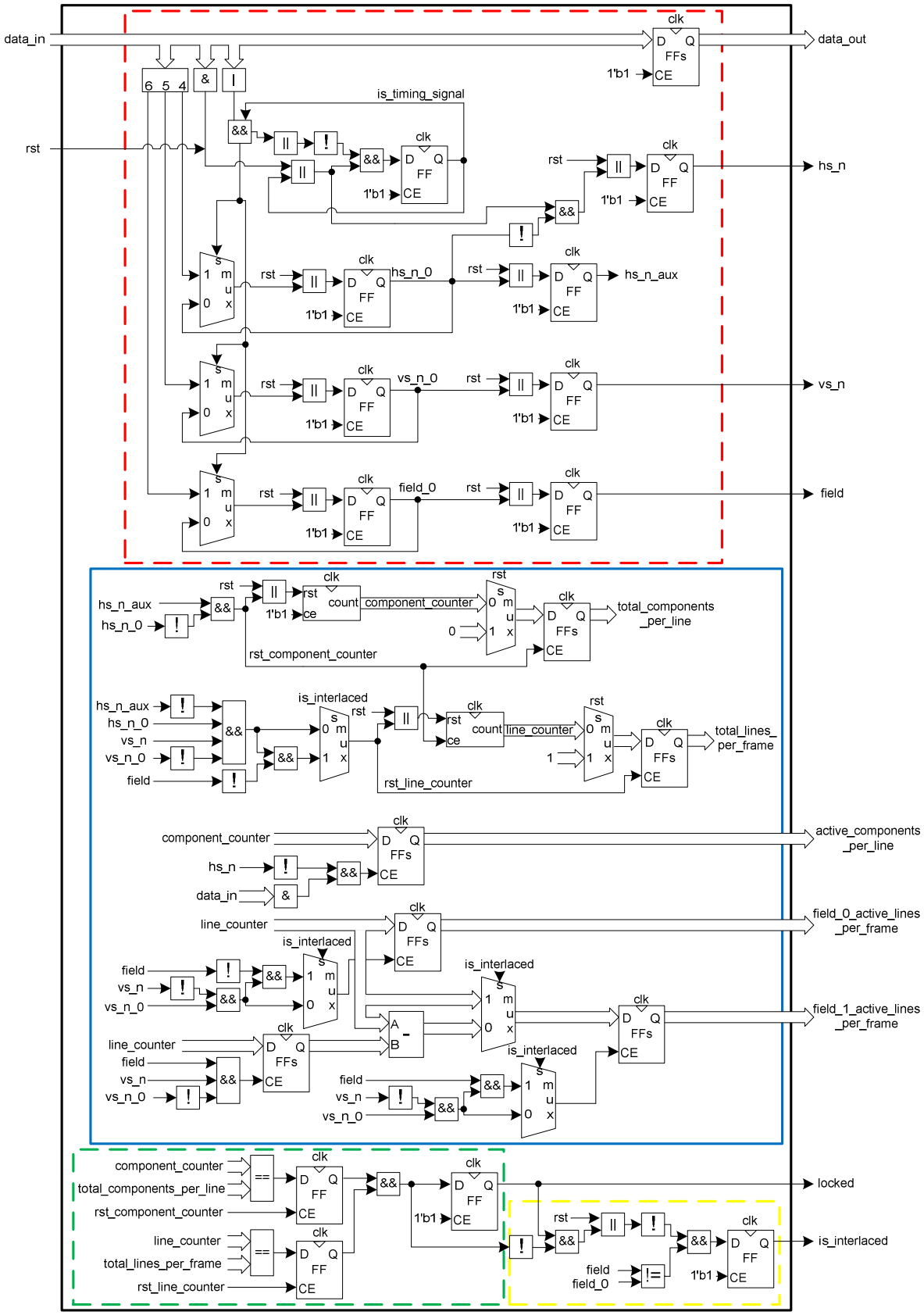


Figura 3-10: Esquema module_1.

Una tercera parte es la encargada de generar la señal locked, la que se ve destacada con un recuadro verde en la Figura 3-10. Esta sección compara los valores del contador de componentes y el total de componentes cada vez que se reinicia el primero, con lo que se obtiene un flipflop que indica si la cantidad total de componentes por línea se mantiene constante. Un proceso similar se realiza con el contador de líneas, obteniendo un flipflop que indica que el número total de líneas del cuadro se mantiene fijo. Cuando estos dos flipflop están activos entonces la señal locked es 1, en caso de que cualquiera de los flipflops este inactivo, la señal locked es 0.

Finalmente existe una cuarta parte que se encarga de determinar si la señal es entrelazada o progresiva, esta se indica en el recuadro amarillo de la Figura 3-10. La señal is_interlaced se activa cuando la señal field cambia y se desactiva cuando la señal locked se desactiva.

3.4.2.2 Módulo module_2

Este módulo es el encargado de generar un interfaz de conexión con el modulo MPMC para la escritura de los datos de video en memoria. Además en este modulo se descartan los campos impares para las señales de video entrelazado. En la Tabla 3-9 se muestran las señales de entrada y salida de este modulo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
Interfaz con module_4			
mem_start_addr	Entrada	32	Indica la dirección del inicio del buffer de cuadro donde se almacenara la imagen activa.
mem_end_addr	Entrada	32	Indica la dirección del término del buffer de cuadro donde se almacenara la imagen activa.
mem_working		-	Indica que el modulo está escribiendo en memoria.
Interfaz con modulo MPMC			
clk_mem	Entrada	-	Reloj que sincroniza los flipflop relacionados con entradas y salidas de modulo MPMC.
InitDone	Entrada	-	Revisar Tabla 2-17.
AddrAck	Entrada	-	Revisar Tabla 2-17.
WrFIFO_Empty	Entrada	-	Revisar Tabla 2-17.
Size	Salida	4	Revisar Tabla 2-17.
AddrReq	Salida	-	Revisar Tabla 2-17.
RNW	Salida	-	Revisar Tabla 2-17.
Addr	Salida	32	Revisar Tabla 2-17.
WrFIFO_Push	Salida	-	Revisar Tabla 2-17.
WrFIFO_Data	Salida	-	Revisar Tabla 2-17.
WrFIFO_BE	Salida	4	Revisar Tabla 2-17.
WrFIFO_Flush	Salida	-	Revisar Tabla 2-17.

Tabla 3-9: Señales de entrada y salida de module_2.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
Interfaz con module_1			
clk_video	Entrada	-	Señal de reloj de entrada de video.
video_data	Entrada	8	Datos de video.
hs_n	Entrada	-	Señal de sincronización horizontal.
vs_n	Entrada	-	Señal de sincronización vertical.
field	Entrada	-	Señal de sincronización de campo.
locked	Entrada	-	Señal de estabilidad de señal de video.
is_interlaced	Entrada	-	Indicador de tipo de video.

Tabla 3-9: Señales de entrada y salida de module_2 (Continuación).

Este módulo se inicia describiendo las señales RNW, WrFIFO_BE y WrFIFO_Flush, que son señales fijas y mantienen los valores 1'b0, 4'hf y 1'b0 respectivamente. La señal RNW se deja fija dado que este modulo solo escribe en memoria, en cuanto la señal WrFIFO_BE se debe a que este modulo funciona solo con transferencias de datos de 32 bits. La última señal es por recomendación de datasheet de utilizar esta señal en 0. La descripción de estas señales es destacada por un recuadro amarillo en la Figura 3-11.

Luego se generan dos señales de control, la primera de ellas llamada rst_or_unlocked que mantiene en un estado de inactividad al módulo mientras la señal rst este activa o si el video entrante no es estable. La segunda señal llamada InitWrDone impide que el módulo empiece las transferencias de escritura hasta que el modulo MPMC termine su inicialización y la señal de video sea estable. La generación de estas señales es esquematizada en la Figura 3-11 en la zona demarcada por el recuadro naranjo.

El recuadro rojo de la Figura 3-11, contiene los elementos para la generación de las señales WrFIFO_Data, WrFIFO_Push y AddrReq. El funcionamiento de esta sección parte con la generación de la señal wr_en la cual corresponde a un indicador de video activo que es obtenido de las señales de sincronización hs_n y vs_n, y además en el caso de video entrelazado indica que es video activo del campo 0. La principal función de esta señal es habilitar la escritura en módulo fifo_8_to_32_bits, el cual como su nombre lo dice es una memoria organizada en forma de cola que además de almacenar agrupa cuatro bytes de información en una palabra de 32 bits. La segunda función es habilitar el contador counter_in, el cual tiene el cometido de contar la cantidad de palabras incorporadas a fifo_8_to_32_bits y cuando este valor es igual valor de palabras que se van a transmitir reinicia su conteo e inicia el conteo del contador counter_out. Este contador funciona de forma continua hasta que su valor se equipara a la cantidad de palabras que se van a transmitir. Mientras esté trabajando counter_out la cola deberá estar transmitiendo los datos almacenados en ella, por esto se genera la señal rd_en que habilita la lectura de la cola. La obtención de los datos de la cola tiene una latencia de un ciclo de clk_mem, por esta razón se retrasa en un ciclo la señal rd_en y se obtiene la señal

WrFIFO_Push. Finalmente cuando counter_out es igual al número de palabras transmitidas se activa el flipflop que genera la señal AddrReq, y este se mantiene activo hasta la activación de la señal AddrAck.

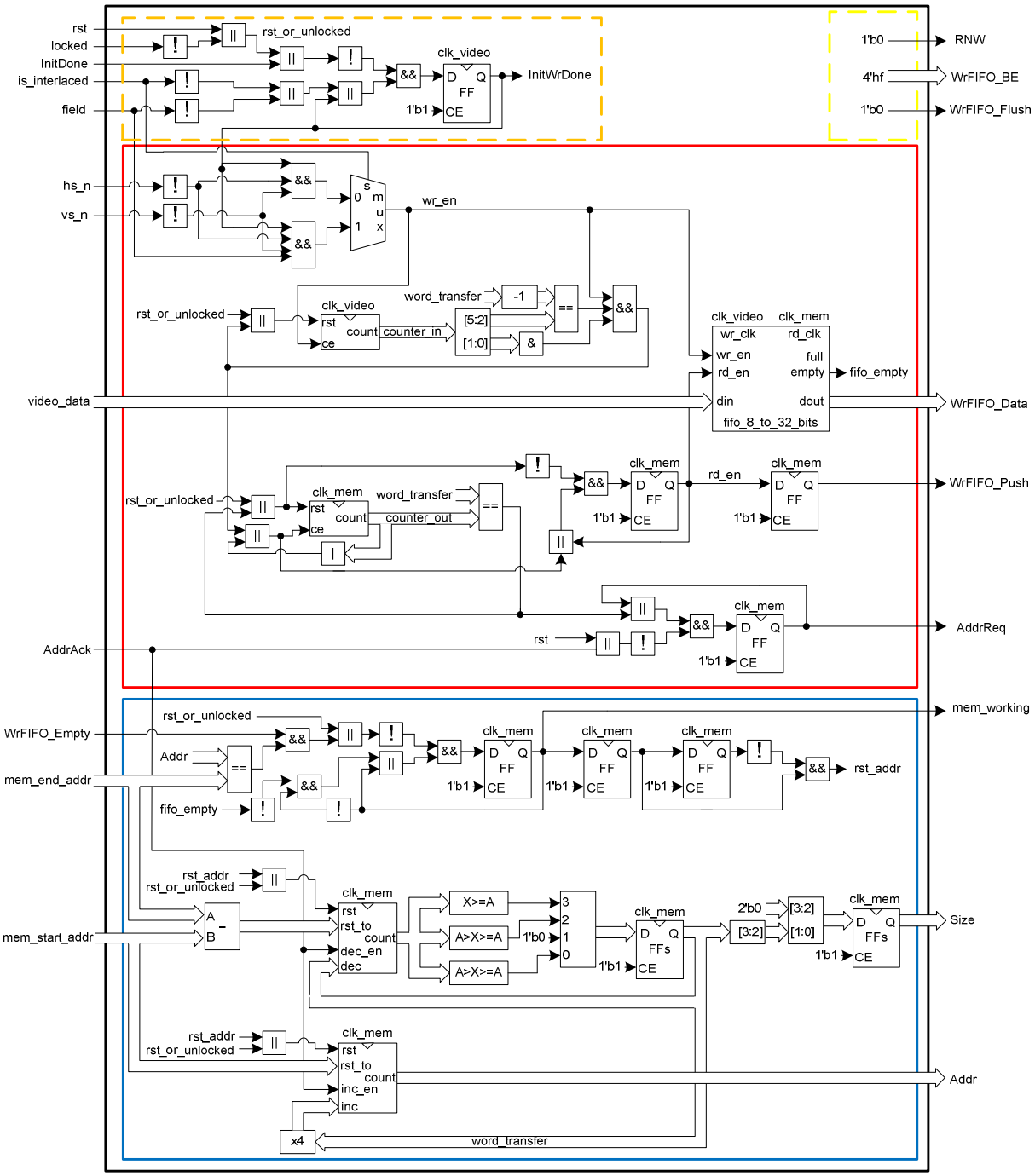


Figura 3-11: Esquema module_2.

La última sección de este módulo señalada en la Figura 3-11 con el cuadrado azul, se inicia con la generación de la señal mem_working, la cual es activada cuando la cola fifo_8_to_32_bits deja de estar vacía y se desactiva cuando la dirección de escritura es igual mem_end_addr y ademas la cola del modulo MPMC está vacía. La señal mem_working es utilizada para generar la señal interna rst_addr que, como más adelante se mencionará, reinicia la cuenta del contador encargado de generar Addr, al valor de mem_start_addr. Esta señal generada en el module_4 tiene la característica de no poder ser modificada mientras se esté trabajando en alguno de los buffers de cuadro que utiliza este módulo, tomando en consideración el caso que el cambio de dirección es realizado en el mismo instante que la activación de la señal mem_working se diseña la señal rst_addr para que después de un ciclo de reloj ocurrida la activación de mem_working esta señal se active por un ciclo de reloj, y así evitar una mala adquisición de la señal de inicio por la señal Addr. La señal rst_addr también es utilizada por otro contador encargado de indicar el número de palabras que faltan por transferir, el cual al activarse rst_addr calcula el número total de palabras del campo o cuadro (dependiendo del tipo de video), a través de la resta de las direcciones de inicio y termino del buffer. Este valor disminuye cada vez que se realiza una transferencia, lo que se ve reflejado en la activación de la señal AddrAck, en un valor equivalente al número de palabras transmitidas. El número de palabras transmitidas es calculado como la transferencia máxima que se puede realizar dado la cantidad de elementos faltantes por transmitir, en el diseño este valor es representado por la señal word_transfer que posteriormente es traducido para la obtención de la señal Size. Finalmente esta sección genera la señal Addr reiniciando un contador a la señal mem_start_addr cuando se activa la señal rst_addr y se incrementa su valor cada vez que se realiza una transferencia, en cuatro veces el tamaño de la transferencia, esto debido a la estructura de las direcciones del modulo MPMC.

3.4.2.2.1 Módulo fifo_8_to_32_bits

Este módulo como ya fue comentado cumple la función de almacenar datos usando el algoritmo de cola o primero en llegar, primero en irse (First In First Out, FIFO). Ademas incorpora la capacidad de manejar dos relojes, uno encargado de sincronizar la escritura y el otro la lectura. En la Tabla 3-10 se muestran las señales de entrada y salida de este modulo y en la Figura 3-12 un esquema de la descripción realizada en HDL. En el esquema se puede apreciar que el almacenamiento de memoria se separó en cuatro diferentes memorias, esto se debe a que se utiliza memoria distribuida para el almacenamiento, debido que este módulo tiene como tarea principal pasar de 8 a 32 bits los datos y no su almacenamiento, el que se le encomienda a la cola del módulo MPMC. Este tipo de memoria es menos eficiente por lo que para que cumpla las restricciones de velocidad impuesta por el reloj del MPMC, se debe describir de esta forma el módulo. Otro elemento a tener en consideración son los bancos de flipflops que generan la señal dout, estos incorporan una latencia de un ciclo de reloj a la entrega de los datos.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
wr_clk	Entrada	-	Reloj de escritura.
rd_clk	Entrada	-	Reloj de lectura.
wr_en	Entrada	-	Habilita la escritura de la cola.
din	Entrada	8	Datos de entrada.
rd_en	Entrada	-	Habilita la lectura de la cola.
dout	Salida	32	Datos de salida.
empty	Salida	-	Indica que no existen elementos en la cola.
full	salida	-	Indica que no se pueden añadir más elementos a la cola.

Tabla 3-10: Entradas y salidas de modulo fifo_8_to_32_bits.

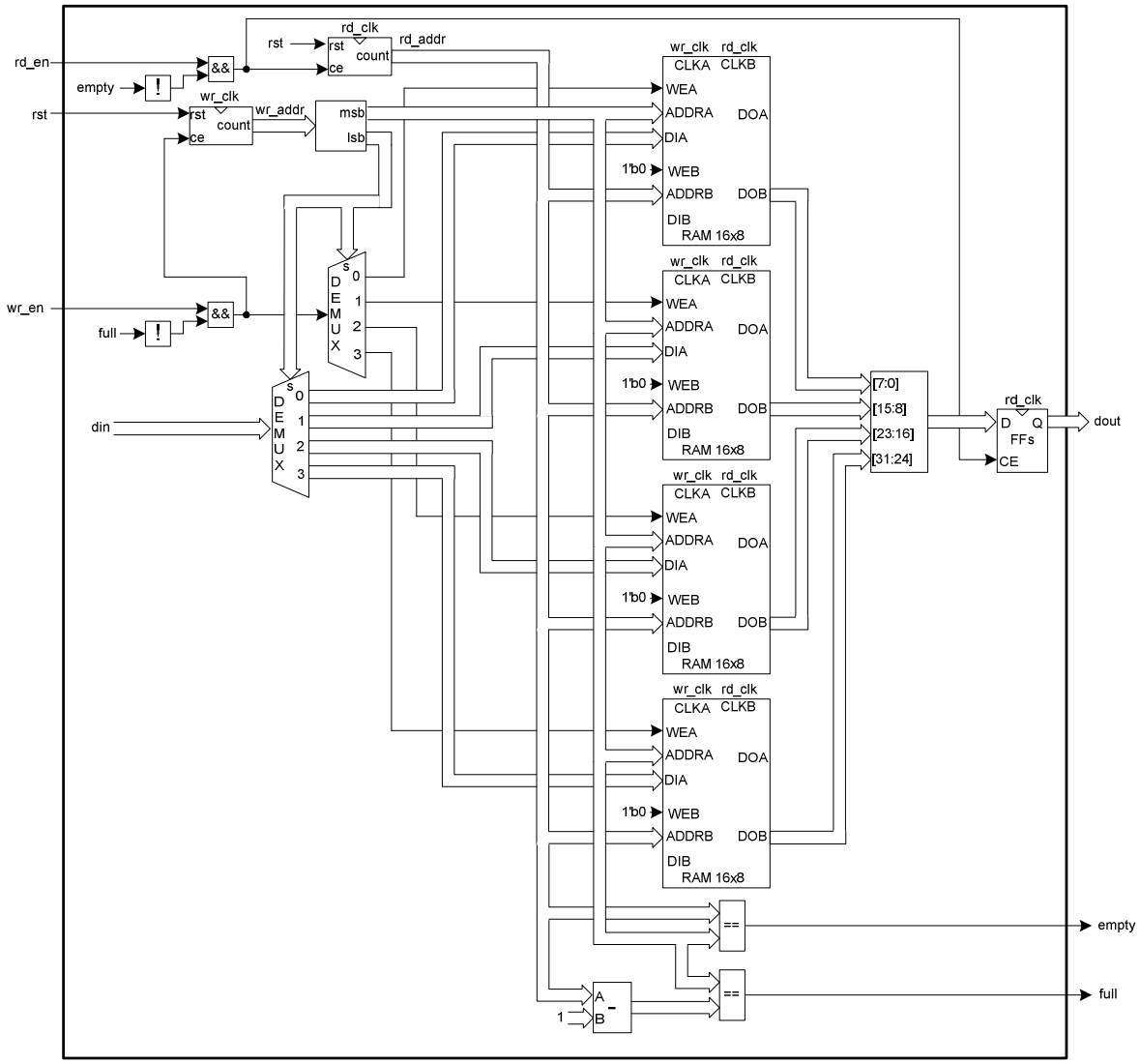


Figura 3-12: Esquema modulo fifo_8_to_32_bits.

3.4.2.3 Módulo module_3

Este módulo es el encargado de redimensionar el video, en la Tabla 3-11 se muestran las señales de entrada y salida de este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
Interfaz con module_4			
hm	Entrada	4	Multiplicador de dimensión horizontal.
hd	Entrada	4	Divisor de dimensión horizontal.
vm	Entrada	4	Multiplicador de dimensión vertical.
vd	Entrada	4	Divisor de dimensión vertical.
si_start_addr	Entrada	32	Dirección de inicio de buffer de entrada al redimensionador.
si_end_addr	Entrada	32	Dirección de término de buffer de salida al redimensionador.
si_with_contents	Entrada	-	Indica si buffer de entrada al redimensionador tiene datos validos.
si_working_on	Entrada	-	Indica de que buffer deben adquirirse los datos de para el redimensionador.
si_working	Entrada	-	Indica si el buffer de entrada al redimensionador está siendo leído.
si_acpl	Entrada	12	Número de componentes activos por línea de entrada de video
si_alpf	Entrada	12	Número de líneas activos por cuadro (o campo) de entrada video.
so_start_addr	Entrada	32	Dirección de inicio de buffer de salida a redimensionado.
so_end_addr	Entrada	32	Dirección de término de buffer de salida a redimensionado.
so_working	Entrada	-	Indica si el buffer de salida al redimensionador está siendo escrito.
Interfaz con módulo MPMC			
si_InitDone	Entrada	-	Conexión InitDone de entrada al redimensionador. Ver Tabla 2-17.
si_AddrAck	Entrada	-	Conexión AddrAck de entrada al redimensionador. Ver Tabla 2-17.
si_WrFIFO_Empty	Entrada	-	Conexión WrFIFO_Empty de entrada al redimensionador. Ver Tabla 2-17.
si_Size	Salida	4	Conexión Size de entrada al redimensionador. Ver Tabla 2-17.
si_AddrReq	Salida	-	Conexión AddrReq de entrada al redimensionador. Ver Tabla 2-17.
si_RNW	Salida	-	Conexión RNW de entrada al redimensionador. Ver Tabla 2-17.
si_Addr	Salida	32	Conexión Addr de entrada al redimensionador. Ver Tabla 2-17.
si_WrFIFO_Push	Salida	-	Conexión WrFIFO_Push de entrada al redimensionador. Ver Tabla 2-17.
si_WrFIFO_Data	Salida	-	Conexión WrFIFO_Data de entrada al redimensionador. Ver Tabla 2-17.
si_WrFIFO_BE	Salida	4	Conexión WrFIFO_BE de entrada al redimensionador. Ver Tabla 2-17.
si_WrFIFO_Flush	Salida	-	Conexión WrFIFO_Flush de entrada al redimensionador. Ver Tabla 2-17.
so_AddrAck	Entrada	-	Conexión AddrAck de salida al redimensionador. Ver Tabla 2-17.
so_WrFIFO_Empty	Entrada	-	Conexión WrFIFO_Empty de salida al redimensionador. Ver Tabla 2-17.
so_Size	Salida	4	Conexión Size de salida al redimensionador. Ver Tabla 2-17.
so_WrFIFO_BE	Salida	4	Conexión WrFIFO_BE de salida al redimensionador. Ver Tabla 2-17.
so_WrFIFO_Flush	Salida	-	Conexión WrFIFO_Flush de salida al redimensionador. Ver Tabla 2-17.
so_AddrReq	Salida	-	Conexión AddrReq de salida al redimensionador. Ver Tabla 2-17.
so_RNW	Salida	-	Conexión RNW de salida al redimensionador. Ver Tabla 2-17.
so_WrFIFO_Push	Salida	-	Conexión WrFIFO_Push de salida al redimensionador. Ver Tabla 2-17.
so_WrFIFO_Data	Salida	32	Conexión WrFIFO_Data de salida al redimensionador. Ver Tabla 2-17.
so_Addr	Salida	32	Conexión Addr de salida al redimensionador. Ver Tabla 2-17.

Tabla 3-11: Señales de entrada y salida de module_3.

Las funciones de este módulo son separadas en cuatro módulos separados, llamados scaler_in, scaler_control, scaler y scaler_out. El primero de esto se encarga de obtener los datos de video activos desde la memoria utilizando una interfaz de lectura con el módulo MPMC. El bloque scaler_control es el encargado de general las señales de administración para realizar el redimensionamiento. El bloque scaler contiene la lógica matemática que permite el procesamiento de los datos. Y finalmente scaler_out se encarga de escribir en memoria la información de la imagen redimensionada. En la Figura 3-13 se muestra la interconexión entre los cuatro bloques.

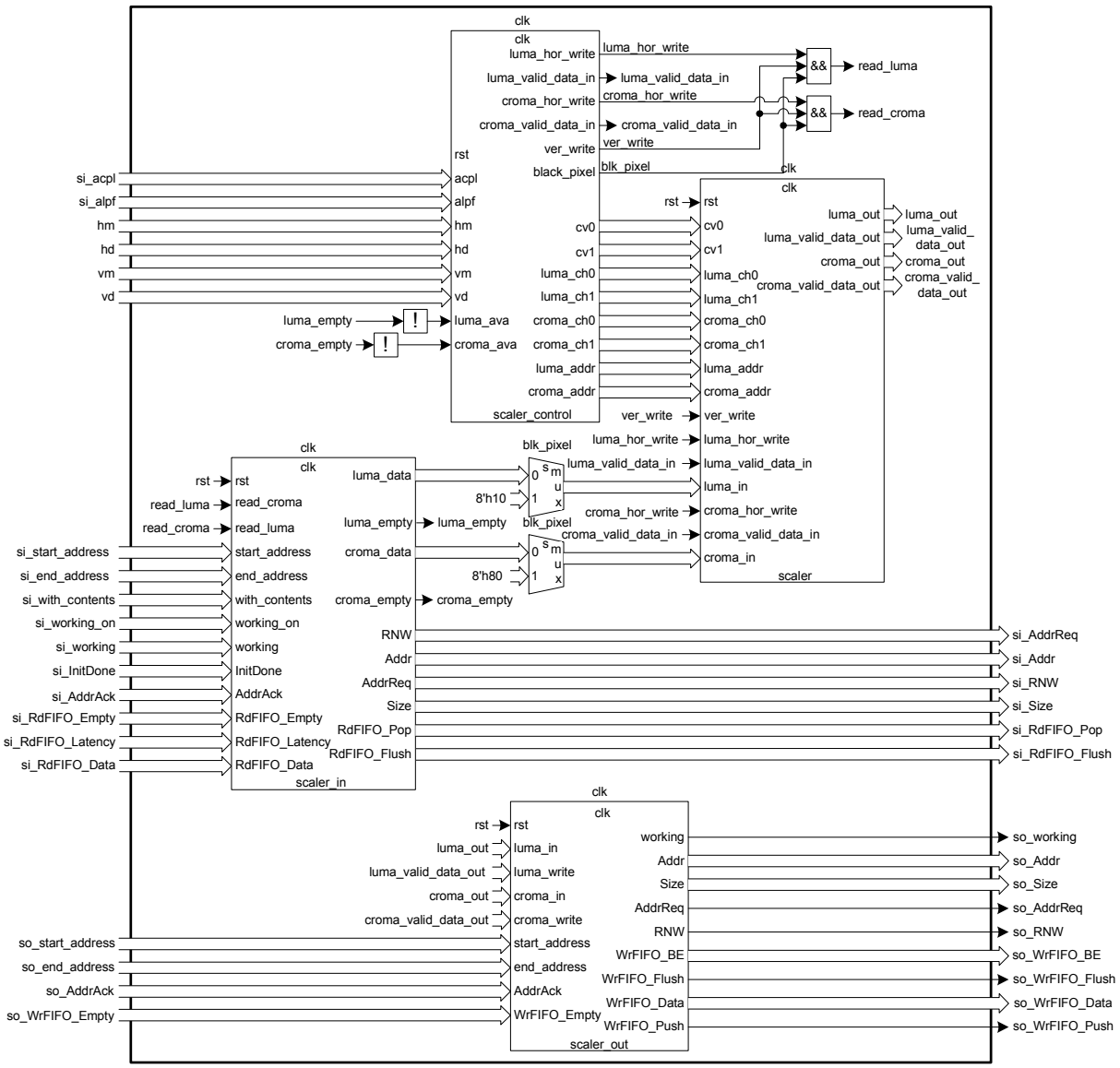


Figura 3-13: Esquema module_3.

3.4.2.3.1 Módulo scaler_in

Este módulo tiene como principal función la obtención de datos de video desde la memoria, por lo que es implementado utilizando una interfaz NPI. Además tiene la función de separar los datos de memoria en datos de luma y croma. En la Tabla 3-11 se muestran las señales de entrada y salida de este modulo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
Interfaz con module_4			
start_addr	Entrada	32	Dirección de inicio de buffer.
end_addr	Entrada	32	Dirección de término de buffer.
with_contents	Entrada	-	Indica si buffer el tiene datos validos.
working_on	Entrada	-	Indica el buffer desde el donde se deben adquirir los datos.
working	Entrada	-	Indica si el buffer está siendo leído.
Interfaz con módulo MPMC			
InitDone	Entrada	-	Ver Tabla 2-17.
AddrAck	Entrada	-	Ver Tabla 2-17.
WrFIFO_Empty	Entrada	-	Ver Tabla 2-17.
Size	Salida	4	Ver Tabla 2-17.
AddrReq	Salida	-	Ver Tabla 2-17.
RNW	Salida	-	Ver Tabla 2-17.
Addr	Salida	32	Ver Tabla 2-17.
WrFIFO_Push	Salida	-	Ver Tabla 2-17.
WrFIFO_Data	Salida	-	Ver Tabla 2-17.
WrFIFO_BE	Salida	4	Ver Tabla 2-17.
WrFIFO_Flush	Salida	-	Ver Tabla 2-17.
Interfaz con módulo scaler_control			
read_luma	Entrada	-	Señal de activación en alto, que indica la lectura de un componente de luma.
read_croma	Entrada	-	Señal de activación en alto, que indica la lectura de un componente de croma.
luma_empty	Salida	-	Indica si no existen componentes luma para la lectura.
croma_empty	Salida	-	Indica si no existen componentes croma para la lectura.
Interfaz con módulo scaler			
luma_data	Salida	8	Datos de luma.
croma_data	Salida	8	Datos de croma.

Tabla 3-12: Entradas y salidas de scaler_in.

La lectura de la memoria parte cuando existe un cambio en el buffer de lectura, lo que equivale a que existen un nuevo cuadro (o campo), con esto se genera una señal de inicio llamada start, la que tiene varias funciones, la primera de ella es activar la señal working y la segunda es reiniciar el valor de los contadores que contienen la dirección de transferencia y los elementos a transferir. En la Figura 3-14 se muestra la implementación de esta señal, que además es controlada por InitDone y with_contents para que esta señal solo se active cuando MPMC este inicializado y el buffer tenga datos validos.

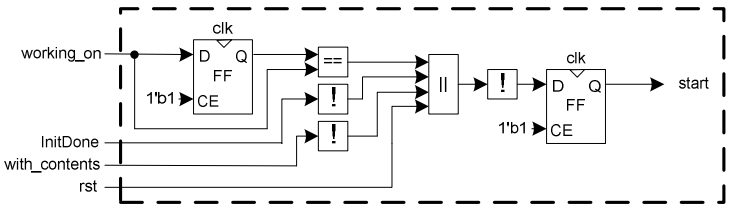


Figura 3-14: Esquema scaler_in, generación señal start.

Las señales Addr y Size son generadas de manera similar a la realizada por module_2. La señal Addr es generada por un contador que es reiniciado al valor start_address cuando se activa la señal start y incrementa su valor en cuatro veces el número de palabras transferidas cuando se activa la señal AddrAck. Para la señal Size el proceso se inicia con un contador que contiene el número de elementos restantes a transferir, que es reiniciado con start a la cantidad de elementos almacenados en el buffer de video. Este contador a diferencia de Addr tiene un decremento igual número de elementos transferidos cada vez que se activa AddrAck. Con el valor de los elementos restantes por transferir es calculado el máximo número de elementos que se pueden transferir, el que es guardado por el registro word_transfer. Finalmente este valor es codificado obteniendo el valor de Size. En la Figura 3-15 se muestra el esquema de la generación de las señales Size y Addr.

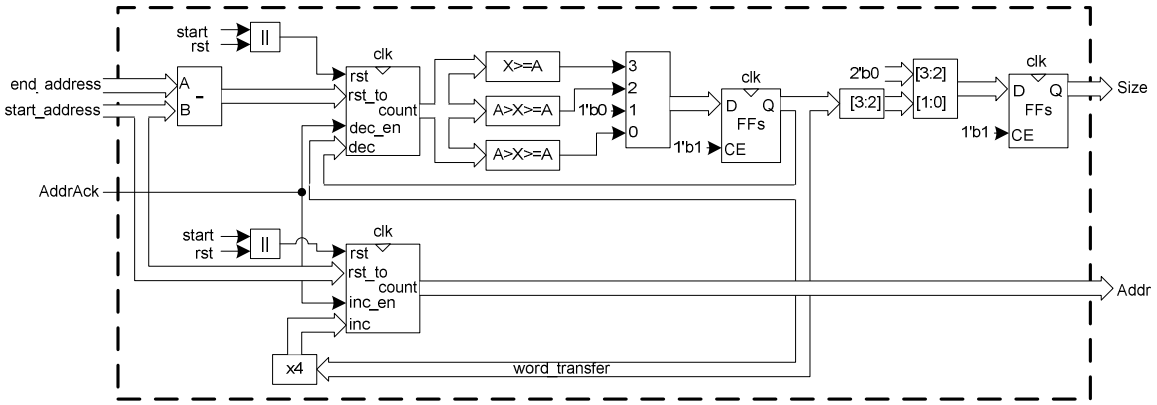


Figura 3-15: Esquema scaler_in, generación señales Size y Addr.

La separación de datos de luma y croma a partir de las palabras obtenidas del módulo MPMC, es realizada por la cola fifo_32_to_16_bits. La petición de datos a MPMC usando RdFIFO_Pop es diseñada para que se realice siempre que existan datos en la cola del dispositivo MPMC y fifo_32_to_16_bits no esté llena. Debido a la latencia en la entrega de datos por parte de bloque MPMC se le asigna un número máximo de elementos almacenados por la cola de este modulo antes de detener la transferencia. Además la señal de escritura de la cola es obtenida como un retraso de la señal RdFIFO_Pop igual a RdFIFO_Latency. En la Figura 3-16 se muestra la implementación de esta sección.

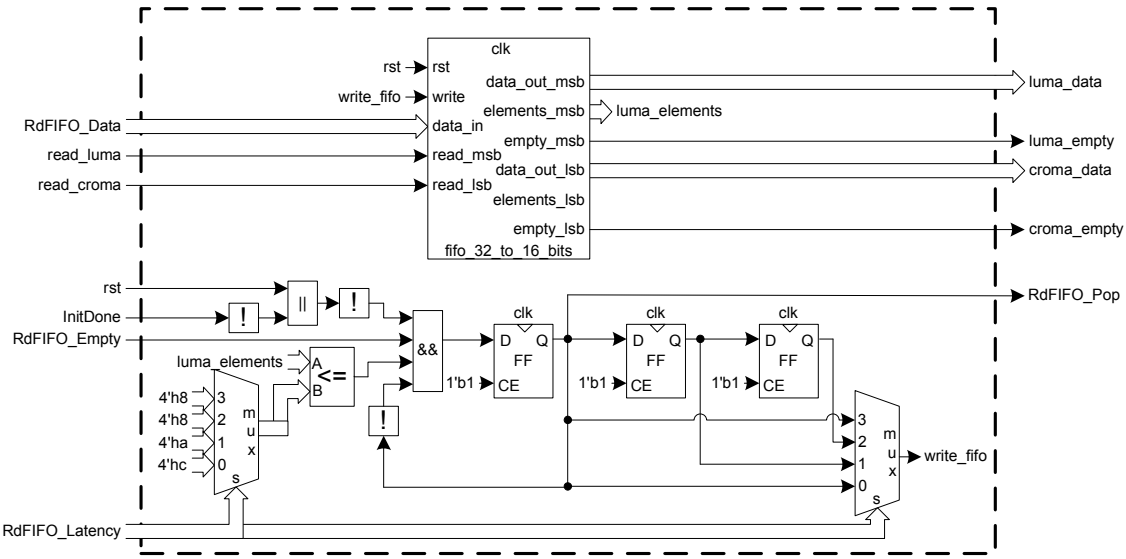


Figura 3-16: Esquema scaler_in, generación RdFIFO_Pop y almacenamiento de datos.

El comportamiento de la señal AddrReq es controlado por un contador que tiene el número de palabras que están almacenadas en la cola del módulo MPMC, que incrementa su valor en el número de palabras transferidas cuando se activa AddrAck y disminuye en uno cada ciclo de reloj que esta activa la señal RdFIFO_Pop. La principal función de este contador es impedir que se hagan peticiones de transferencia cuando la cola RdFIFO está llena. AddrReq también es controlado por la señal AddrReq_lag que tiene por cometido impedir que se active la señal antes de tener estable la señal Size. En la Figura 3-17 se muestra el esquema utilizado para la generación de AddrReq.

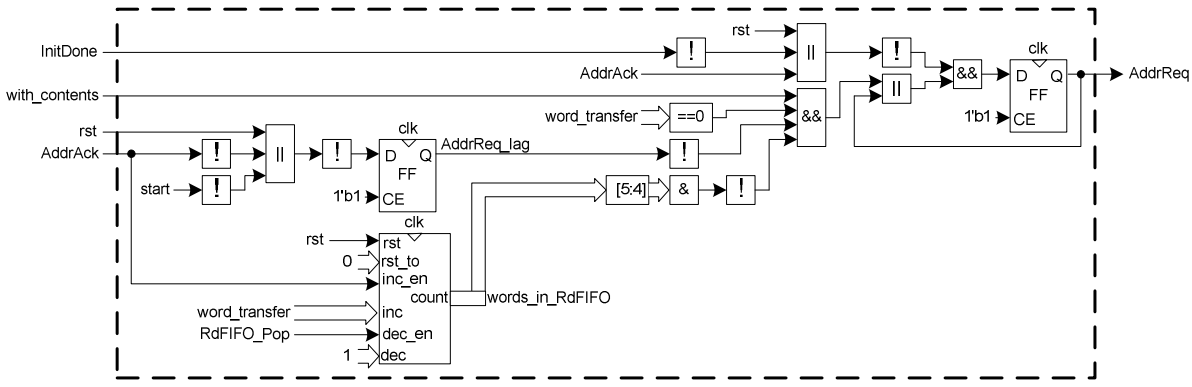


Figura 3-17: Esquema scaler_in, generación de AddrReq.

La señal working se activa cuando start es 1 y se mantiene activa hasta que el número de palabras por transferir sea cero y no existan datos en RdFIFO. Además se incorpora como señal de control a AddrReq_lag para compensar el ciclo de reloj que hay entre la activación de

working y el cálculo de word_transfer. En la Figura 3-18, se muestra el circuito usado para señal working.

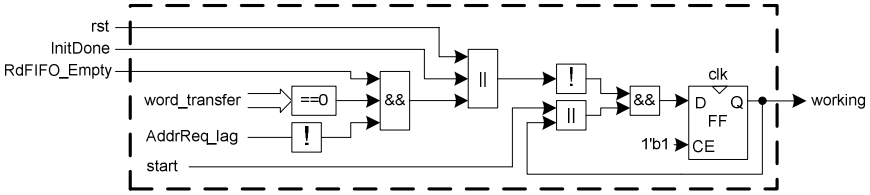


Figura 3-18: Esquema scaler_in, generación de working.

La señal RdFIFO_Flush es usada solo en caso de reinicio del sistema, en caso que existan elementos en RdFIFO y la señal RNW siempre esta activa para indicar que este módulo lee la memoria, en la Figura 3-19 se muestra la generación de estas dos señales.

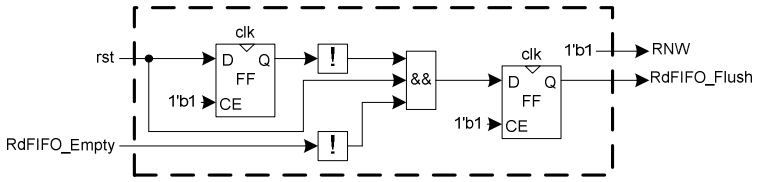


Figura 3-19: Esquema scaler_in, generación RdFIFO_Flush y RNW.

3.4.2.3.1.1 Módulo fifo_32_to_16_bits

Este dispositivo ordena los bytes de data_in para que sean multiplexados en el tiempo por el módulo fifo_32_to_16_bits. En la Figura 3-20 se muestra un esquema de este dispositivo.

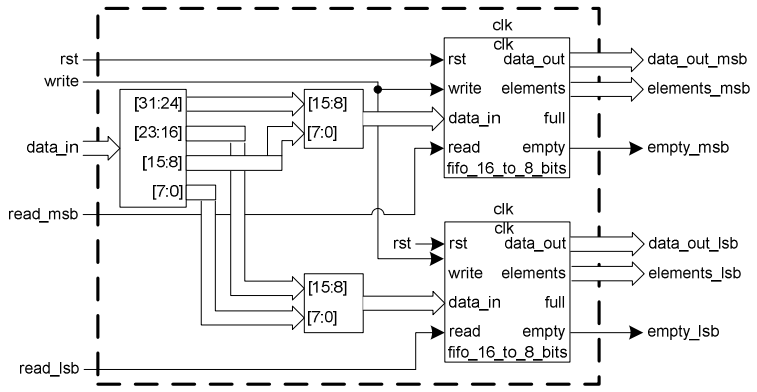


Figura 3-20 Esquema fifo_32_to_16_bits.

3.4.2.3.1.1.1 Módulo fifo_16_to_8_bits

Este módulo almacena y multiplexa en el tiempo los bytes de la información de entrada, en la Figura 3-21 se muestra los dispositivos que contiene este módulo.

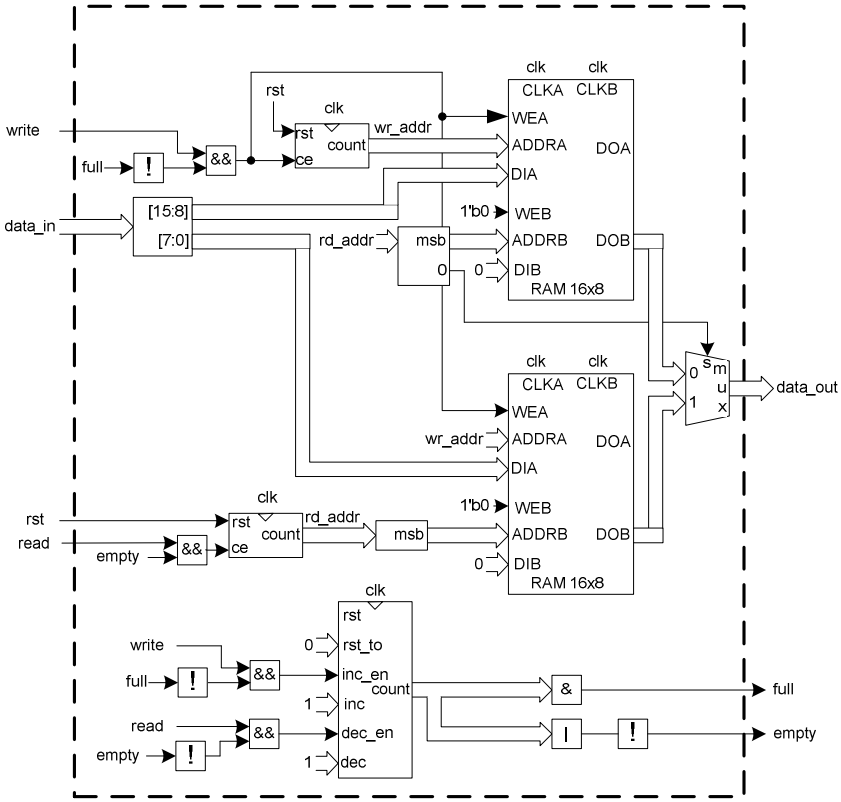


Figura 3-21: Esquema fifo_16_to_8_bits.

3.4.2.3.2 Módulo scaler_control

Este módulo es el encargado de implementar el redimensionamiento del cuadro (o campo) de video, entregando señales de control para obtener los datos desde scaler_in, generar los datos los nuevos datos en scaler y filtrar los datos antes de almacenarlos en memoria. En la Tabla 3-13 se muestran las señales de entrada y salida de este módulo.

Para la implementación del redimensionado explicado en el capitulo 3.2.3 se separa el proceso en dos fases, la primera consiste en el aumento de dimensiones en una cantidad de veces dado por el multiplicador de dimensión y la segunda en realizar una reducción en la dimensiones en una cantidad de veces dado por el divisor de dimensión a la imagen obtenida en la primera fase.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
acpl	Entrada	12	Numero de componentes activos por línea de imagen a redimensionar.
alpf	Entrada	12	Numero de lineas activas por cuadro (o campo) de imagen a redimensionar.
hm	Entrada	4	Multiplicador de dimensión horizontal.
hd	Entrada	4	Divisor de dimensión horizontal.
vm	Entrada	4	Multiplicador de dimensión vertical.
vd	Entrada	4	Divisor de dimensión vertical.
luma_ava	Entrada	-	Indica la existencia de componentes de luma, en cola de scaler_in.
croma_ava	Entrada	-	Indica la existencia de componentes de croma, en cola de scaler_in.
ver_write	Salida	-	Indica a modulo scaler, la escritura de una línea en memoria
luma_valid_data_in	Salida	-	Indica validez de componente de luma.
croma_valid_data_in	Salida	-	Indica validez de componente de croma.
luma_addr	Salida	11	Dirección de escritura y lectura de retraso de línea.
croma_addr	Salida	11	Dirección de escritura y lectura de retraso de línea.
cv0	Salida	8	Coeficiente de filtro vertical.
cv1	Salida	8	Coeficiente de filtro vertical.
luma_ch0	Salida	8	Coeficiente de filtro horizontal, para generación de componentes de luma.
luma_ch1	Salida	8	Coeficiente de filtro horizontal, para generación de componentes de luma.
croma_ch0	Salida	8	Coeficiente de filtro horizontal, para generación de componentes de croma.
croma_ch1	Salida	8	Coeficiente de filtro horizontal, para generación de componentes de croma.

Tabla 3-13: Entradas y salidas de scaler_control.

La primera fase al aumentar el número de elementos de la imagen debe interpolar los nuevos componentes a partir de los datos iníciales, para esto se utiliza la Ecuación 3-1, la cual se hace recorrer por toda la imagen para obtener los nuevos datos. Considerando un filtro de dos taps horizontales y dos taps verticales, cada elemento interpolado seria generado a partir cuatro elementos originales, por lo que para aumentar en un número de veces el tamaño de la imagen es necesario que cada cuarteto de datos se obtenga una cantidad de elementos igual al número de veces que se desea aumentar el tamaño de la imagen. El cálculo de los nuevos elementos, implica un cambio en los coeficientes utilizado por la ecuación para obtener pixeles correspondientes a distintas posiciones dentro del cuarteto.

En la Figura 3-22 se muestra el proceso que se debe realizar para obtener una imagen de 6x6 a partir de una de 3x3, con un filtro de dos taps verticales y dos horizontales. El proceso se parte calculando el pixel superior izquierdo y así sucesivamente de izquierda a derecha y de arriba a abajo, hasta llegar al pixel inferior derecho.

En la imagen los cuadros de colores identifican a la posición del filtro, que corresponde al cuarteto de elementos con los que se calcula el nuevo pixel. El movimiento del filtro seria de izquierda a derecha manteniéndose en una posición fija para el cálculo de parejas de pixeles, así los dos primeros puntos son calculados por el cuarteto de elementos encerrados por el cuadrado naranjo, los siguientes dos por el rojo y los últimos dos de la línea por el café. En esta última posición no existen suficientes pixeles originales para el cálculo de los nuevos puntos, por lo que se usan pixeles negros para el cómputo de los nuevos pixeles. La siguiente pareja de elementos es calculada nuevamente en el cuadrado naranjo y así se repite la secuencia vista en los primeros seis puntos.

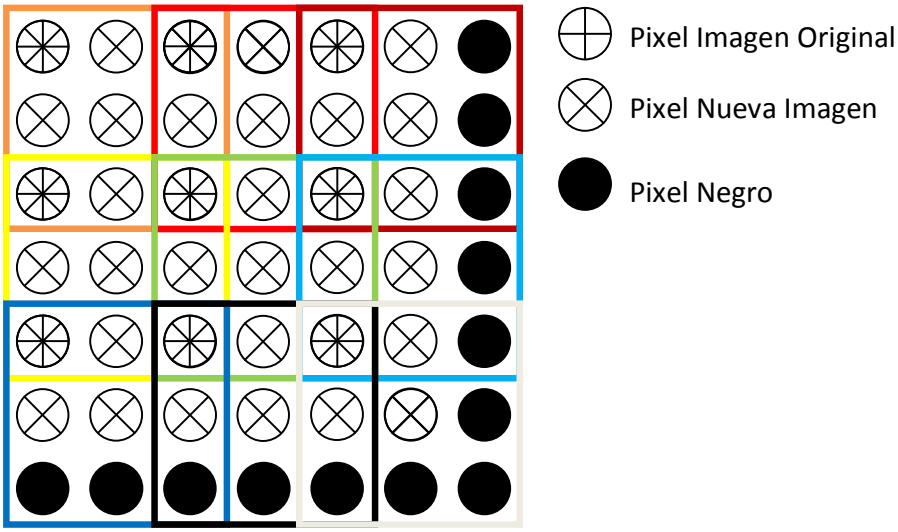


Figura 3-22: Redimensionamiento de imagen de 3x3 a 6x6.

De lo explicado anteriormente se puede inferir el comportamiento de la posición del filtro para un redimensionamiento donde se aumenta la dimensión horizontal en n veces y la dimensión vertical en m veces, este se moverá de izquierda a derecha y de arriba a abajo manteniéndose en el mismo cuarteto de datos para el cálculo de n puntos y en el mismo par de líneas para el cálculo de m líneas.

En cuanto a los coeficientes se consideran dos filtros uno vertical y otro horizontal, cada uno con sus coeficientes. Se puede apreciar en la Figura 3-22 que los coeficientes usados por el filtro horizontal varían en el cálculo de cada componente para una posición fija del filtro, así se tiene un conjunto de coeficientes para el computo del primer elemento y otro grupo para el segundo. Para el filtro vertical los coeficientes varían por línea. Lo último permite llegar a la conclusión que el filtro horizontal tiene n conjuntos de coeficientes y el vertical m, los cuales tienen un orden de aparición fijo.

En la segunda fase para disminuir las dimensiones de la imagen se deben eliminar datos obtenido en la fase anterior. Una imagen a la cual se le quiere disminuir su dimensión horizontal n veces y su tamaño vertical m veces, debe dejar un pixel de cada n pixeles consecutivos y una línea cada m líneas consecutivas, así eliminando los pixeles y líneas restantes del conjunto de datos. Este procedimiento se puede apreciar en el ejemplo de la Figura 3-23, donde la imagen del ejemplo anterior es reducida a una de 2x2.

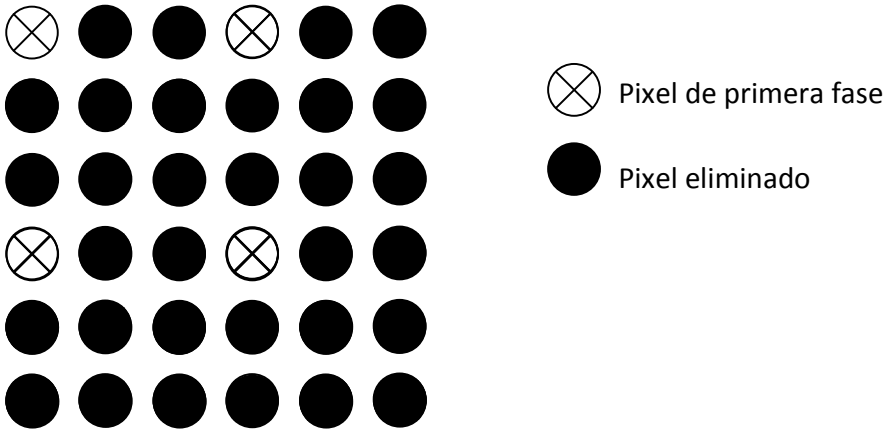


Figura 3-23: Ejemplo de diseminación de fase 1.

Considerando la limitación de multiplicadores y bloques RAM que tiene la FPGA de la plataforma Starter Kit, el filtro se diseña con 2 *taps* verticales y 2 *taps* horizontales, y considerando los coeficientes de una interpolación lineal.

Esta configuración en su implementación más simple requiere el uso de un retraso de línea y cuatro multiplicadores por plano de color, lo que implicaría el uso de 12 multiplicadores para el redimensionamiento de una entrada de video. Considerando la limitantes de 20 multiplicadores de la Spartan-3A 700k, solo se podría implementar una sola entrada de video. Para solucionar esto se plantea una implementación donde se procesa las componentes de luma por si sola y las componentes de croma de color rojo y azul son multiplexadas para usar solo cuatro multiplicadores, llevándose a 8 la cantidad de multiplicadores necesarios para la implementación total. Un punto a tenerse en cuenta es que esta implementación no disminuye la eficiencia del redimensionamiento, cuando se tiene video con muestreo 4:2:2 como es el caso de este diseño. En caso de tenerse video con muestro 4:4:4 el redimensionamiento con dos canales multiplexados se demoraría el doble del tiempo en comparación a un redimensionamiento por canal.

Otra consideración tomada en la implementación fue agregar un retraso de línea extra, que permite que se lea desde la memoria solo una vez por componente, mejorando así el rendimiento del redimensionamiento.

Como se explicó en antes el funcionamiento del módulo se divide en dos partes, la primera encarga de aumentar las dimensiones de la imagen y la segunda de disminuirlas. La primera parte además se subdivide en las tareas de determinar la posición del filtro, la determinación de la fase de los coeficientes y la adquisición de los datos. Esta sección es descrita en HDL como se esquematiza en la Figura 3-24.

La implementación de esta sección se parte con la descripción de tres contadores, los que identifican la fase del filtro vertical y la horizontal, llamados luma_hm_counter, croma_hm_counter y vm_counter. Estos contadores son usados también para generar señales de escritura para los retrasos de línea y determinar si la lectura del dato más reciente se realiza desde el primer retraso de línea o desde la cola de scaler_in. Luego de esto se definen dos contadores que identifican la posición del filtro, llamados luma_addr y line_counter, el primero identifica la posición horizontal para el canal de luma y el segundo la vertical para ambos canales. Como se vio previamente la posición de estos filtros se desplaza cada vez que sus fases vuelven a cero, por lo que los contadores incrementan su valor cuando se reinician los contadores luma_hm_counter y vm_counter. En el caso de line_counter la última posición indicada corresponde a la línea de pixeles negros usada para calcular las últimas línea de datos, para luma_addr las dos últimas posiciones corresponden a pixeles negros, esto se debe a que el cálculo de los últimos componentes Cb requieren del penúltimo pixel negro y los componentes Cr del último. La posición del filtro horizontal para los canales de croma es la mitad de luma_addr.

Entrando en detalle de la descripción en verilog, el contador luma_hm_counter no incrementa su valor solo en el caso que se necesite leer un dato desde la cola de scaler_in, estando está vacía, y su condición de reinicio es la de incrementar su valor estando hm-1. El contador que genera la señal croma_hm_counter, además produce la señal croma_addr0 en su bit menos significativo, esta señal es usada para identificar si el componente es Cb o Cr. Las condiciones usadas para el control de este contador son las mismas que luma_hm_counter. Las señales luma_hor_write y croma_hor_write son activadas cuando los contadores respectivos son iguales a cero y se cumple la condición de incremento de estos contadores. La señal croma_addr se forma a partir los bits más significativos de luma_addr, los que son llamados addr y croma_addr0.

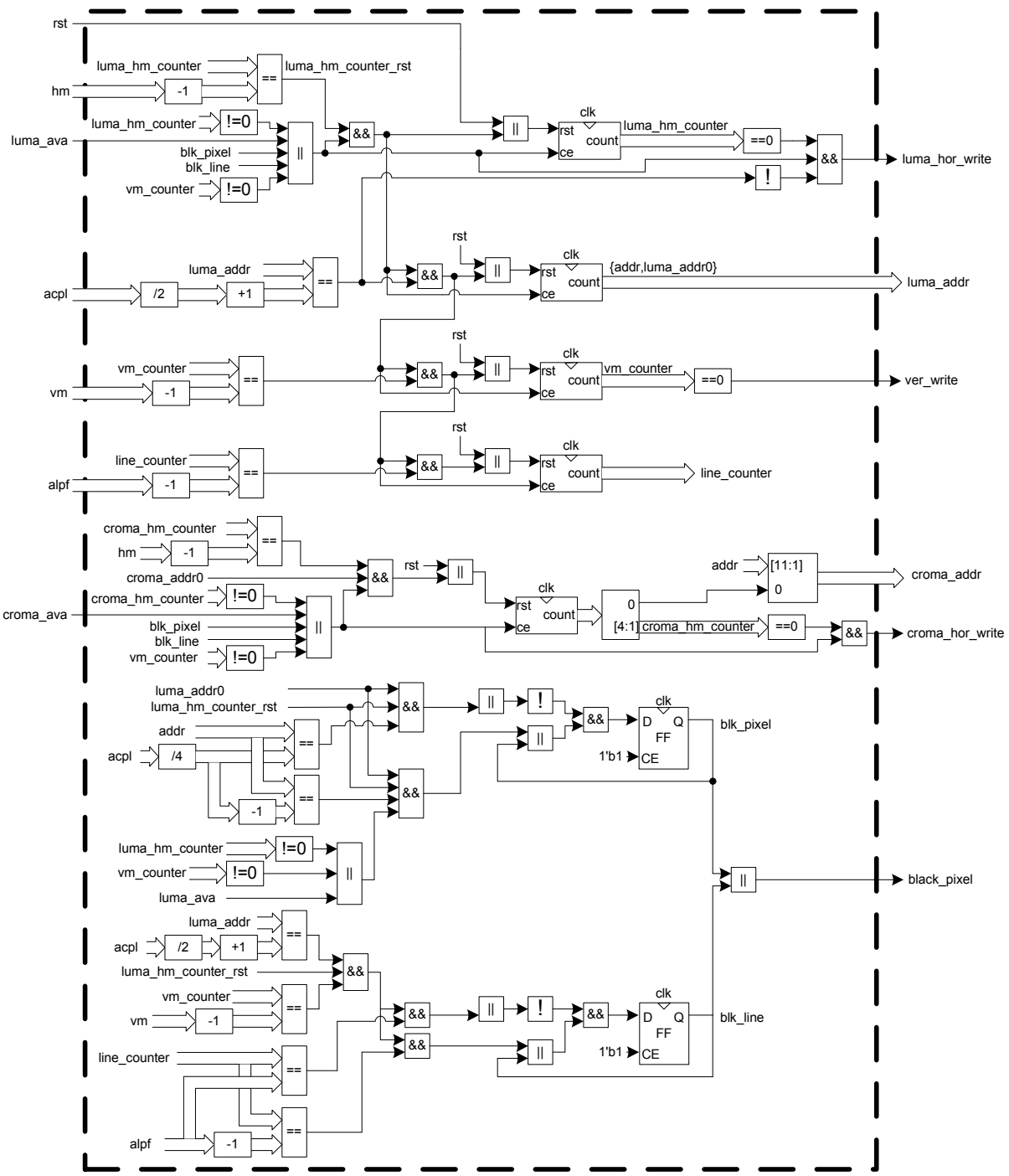


Figura 3-24: Esquema scaler_control, generación de señales de control para aumento de dimensiones.

La generación de los coeficientes de los filtros vertical y horizontal es realizando con memorias ROM que contienen el valor de los coeficientes a usar para distintas fase. En la Figura 3-25 se muestra el esquema utilizado para la generación de los coeficientes, donde los multiplicadores de dimensión y la fase donde se encuentra el filtro son usados como dirección de las ROM para obtener los coeficientes de cada tap. El contenido de estas ROM es el mismo y es indicado en la Tabla 3-14.

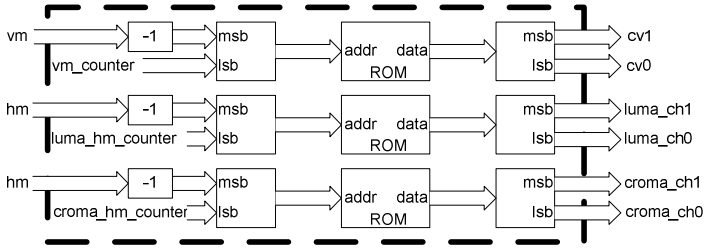


Figura 3-25: Esquema scaler_control, generación de coeficientes de filtros.

Dirección	Contenido de dirección	$Coeficiente1$	$\frac{Coeficiente1}{128}$	$Coeficiente2$	$\frac{Coeficiente2}{128}$
8'h00	16'h0080	0	0	128	1
8'h10	16'h0080	0	0	128	1
8'h11	16'h4040	64	0,5	64	0,5
8'h20	16'h0080	0	0	128	1
8'h21	16'h2b55	43	0,34	55	0,66
8'h22	16'h552b	55	0,66	43	0,34
8'h30	16'h0080	0	0	128	1
8'h31	16'h2060	32	0,25	96	0,75
8'h32	16'h4040	64	0,5	64	0,5
8'h33	16'h6020	96	0,75	32	0,25
8'h40	16'h0080	0	0	128	1
8'h41	16'h1a66	26	0,2	102	0,8
8'h42	16'h334d	51	0,4	77	0,6
8'h43	16'h4d33	77	0,6	51	0,4
8'h44	16'h661a	102	0,8	26	0,2
8'h50	16'h0080	0	0	128	1
8'h51	16'h156b	21	0,16	107	0,84
8'h52	16'h2b55	43	0,34	55	0,66
8'h53	16'h4040	64	0,5	64	0,5
8'h54	16'h552b	55	0,66	43	0,34
8'h55	16'h6b15	107	0,84	21	0,16
8'h60	16'h0080	0	0	128	1
8'h61	16'h126e	18	0,14	110	0,86
8'h62	16'h255b	37	0,29	91	0,71
8'h63	16'h3749	55	0,43	73	0,57
8'h64	16'h4937	73	0,57	55	0,43
8'h65	16'h5b25	91	0,71	37	0,29
8'h66	16'h6e12	110	0,86	18	0,14
8'h70	16'h0080	0	0	128	1
8'h71	16'h1070	16	0,13	112	0,87
8'h72	16'h2060	32	0,25	96	0,75
8'h73	16'h3050	48	0,38	80	0,62
8'h74	16'h4040	64	0,5	64	0,5
8'h75	16'h5030	80	0,62	48	0,38
8'h76	16'h6020	96	0,75	32	0,25
8'h77	16'h7010	112	0,87	16	0,13

Tabla 3-14: Contenido de ROM de coeficientes de filtros.

La reducción de las dimensiones de la imagen está a cargo de las señales luma_valid_data_in y croma_valid_data_in las cuales son manejadas a través de los contadores luma_hd_counter, croma_hd_counter y vd_counter. Los contadores horizontales incrementan su valor cada vez que lo hacen los contadores hm_counter, con la excepción del primer componente entregado el cual inhabilita el incremento. En el caso de luma_hm_counter también se inhabilita su incremento cuando se entrega al modulo scaler el segundo pixel negro. Para vd_counter ocurre algo similar incrementándose cuando vm_counter se incrementa e inhabilitando este incremento para la primera línea. Finalmente la generación de las señales esta diseñada como se esquematiza en la Figura 3-26.

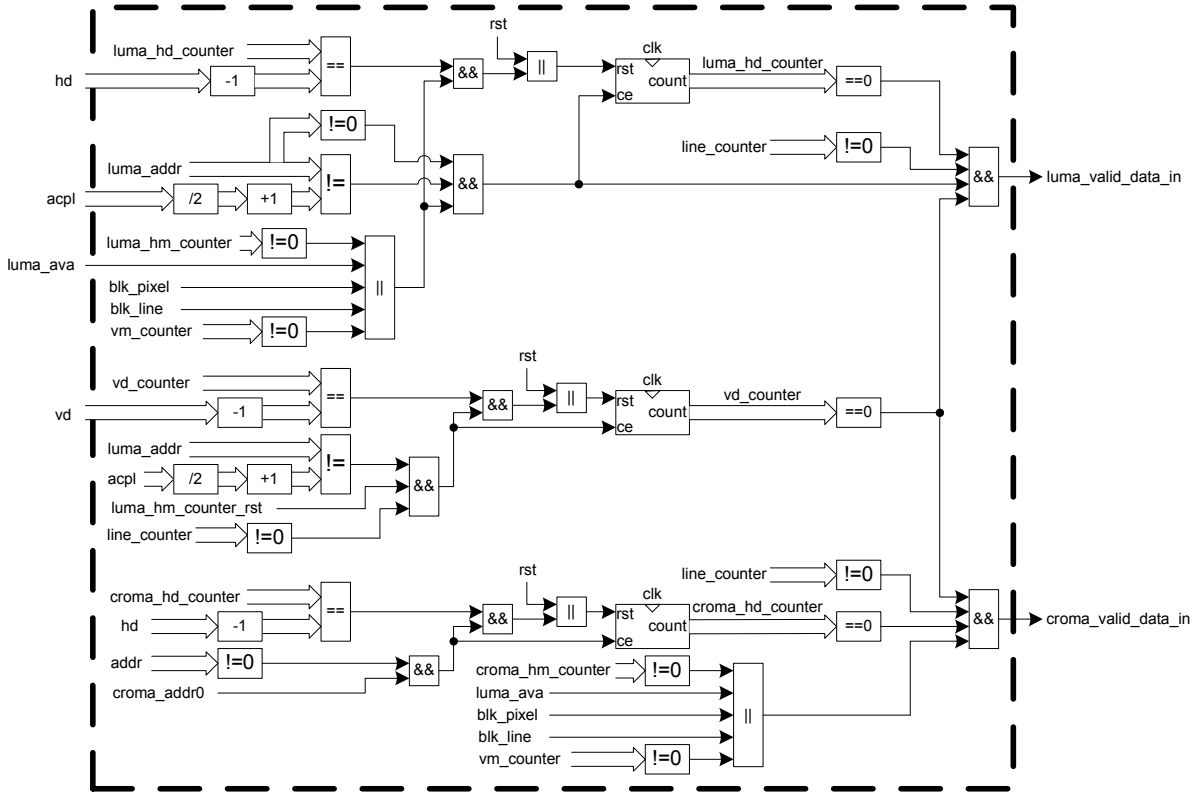


Figura 3-26: Esquema scaler_control, generación de luma_valid_data_in y croma_valid_data_in.

3.4.2.3.3 Módulo scaler

Este módulo es el encargado de realizar las operaciones matemáticas indicadas por la Ecuación 3-2 y la Ecuación 3-3. En la Tabla 3-15 se muestran las señales de entra y salida que tiene este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
ver_write	Entrada	-	Indica la escritura de la línea actual en el retraso de línea.
luma_hor_write	Entrada	-	Indica la escritura del componente de luma en el retraso de línea.
croma_hor_write	Entrada	-	Indica la escritura del componente de croma en el retraso de línea.
luma_valid_data_in	Entrada	-	Indica la valides de la posición y fase del filtro para generar componentes luma de la nueva imagen.
croma_valid_data_in	Entrada	-	Indica la valides de la posición y fase del para generar componentes croma de la nueva imagen.
luma_in	Entrada	8	Datos de entrada de los componentes luma de imagen original.
croma_in	Entrada	8	Datos de entrada de los componentes croma de imagen original.
luma_addr	Entrada	11	Dirección de escritura y lectura de los retrasos de línea.
croma_addr	Entrada	11	Dirección de escritura y lectura de los retrasos de línea.
cv0	Entrada	8	Coeficiente de filtro vertical.
cv1	Entrada	8	Coeficiente de filtro vertical.
luma_ch0	Entrada	8	Coeficiente de filtro horizontal para canal de luma.
luma_ch1	Entrada	8	Coeficiente de filtro horizontal para canal de luma.
croma_ch0	Entrada	8	Coeficiente de filtro horizontal para canal de croma.
croma_ch1	Entrada	8	Coeficiente de filtro horizontal para canal de croma.
luma_out	Salida	8	Datos de salida con los componentes luma de la imagen con sus dimensiones aumentadas.
croma_out	Salida	8	Datos de salida con los componentes croma de la imagen con sus dimensiones aumentadas.
luma_valid_data_out	Salida	-	Indica que componentes luma, pertenecen a la nueva imagen.
croma_valid_data_out	Salida	-	Indica que componentes croma, pertenecen a la nueva imagen.

Tabla 3-15: Entrada y salida de scaler.

En la Figura 3-27 se muestra la implementación de los retrasos de línea y el filtro vertical para las componentes de luma. Esta implementación es idéntica para el croma, por lo que solo se explicará para los componentes luma.

El proceso parte con la generación de una señal de escritura de los retrasos de línea que se activa cuando luma_hor_write y ver_write están activas, luego esta señal y todas las demás señales de entrada son almacenadas en flipflops, para cumplir con las restricciones de reloj. Luego de esto las señales write, luma_addr y luma_in son usadas por el dispositivo line_buffers para generar los componentes de la línea actual y de la línea previa para una posición horizontal dada por luma_addr. En el caso de que write sea 1, el valor de luma_in es almacenado en uno de los retrasos de línea y dos ciclos de reloj después el mismo valor de luma_in se obtiene desde data_out0 y en data_out1 se obtiene el valor para el componente de la misma posición de la línea previa. Para el caso donde write es 0, el valor de luma_in es desechado y después de dos ciclos de reloj se obtiene desde data_out1 el componente de la línea actual y desde data_out2 el de la línea previa. Para la correcta elección de las salidas se retrasa en dos ciclos de reloj la señal write, para usarla como seleccionador del par de entradas usada para la multiplicación con los coeficientes verticales. El resultado de la multiplicación es sumado obteniéndose la señal luma_vert_sum.

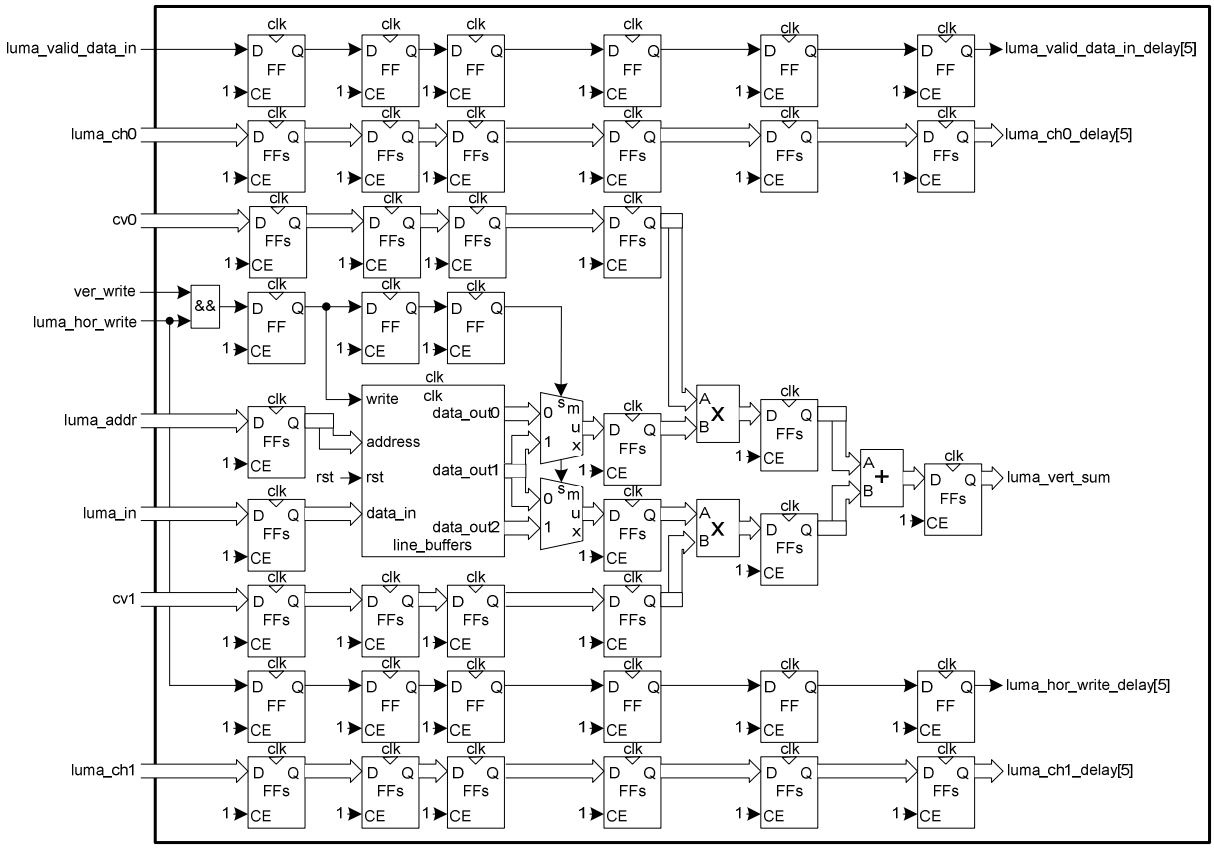


Figura 3-27: Esquema scaler, filtro vertical.

A partir de este punto se diferencian los procesos para las componentes de luma y croma. En la Figura 3-28, se muestra el filtrado horizontal para las componentes de luma. luma_ver_sum es divido por 128, si el retraso de la señal luma_hor_write es 1 este valor es guardado por un banco fliflops y el valor de este en otro, generando así el componente actual y previo de una misma línea. En caso de ser 0 se usan los valores previamente guardados por estos flipflops. Luego de esto se realiza la multiplicación con los coeficientes de luma horizontales y la posterior suma entre sus resultados. Finalmente el resultado es divido por 128 y almacenado en un flipflop, que es usado para igualar la latencia que tiene la sección de croma.

En la Figura 3-29, se muestra el filtrado horizontal para las componentes de croma. En este después de la división por 128, el valor obtenido puede ser guardado en dos bancos de flipflops, la elección de banco se basa en el componente al que corresponde, siendo un banco para almacenar los componentes Cb y el otro los Cr. Cuando alguno de estos bancos es escrito su valor pasa a otro banco produciéndose así los componentes actual y previo de la misma línea. Luego de esto se selecciona desde que banco de flipflops se leen los datos para su previa multiplicación, suma y división obteniéndose así el resultado final del filtro.

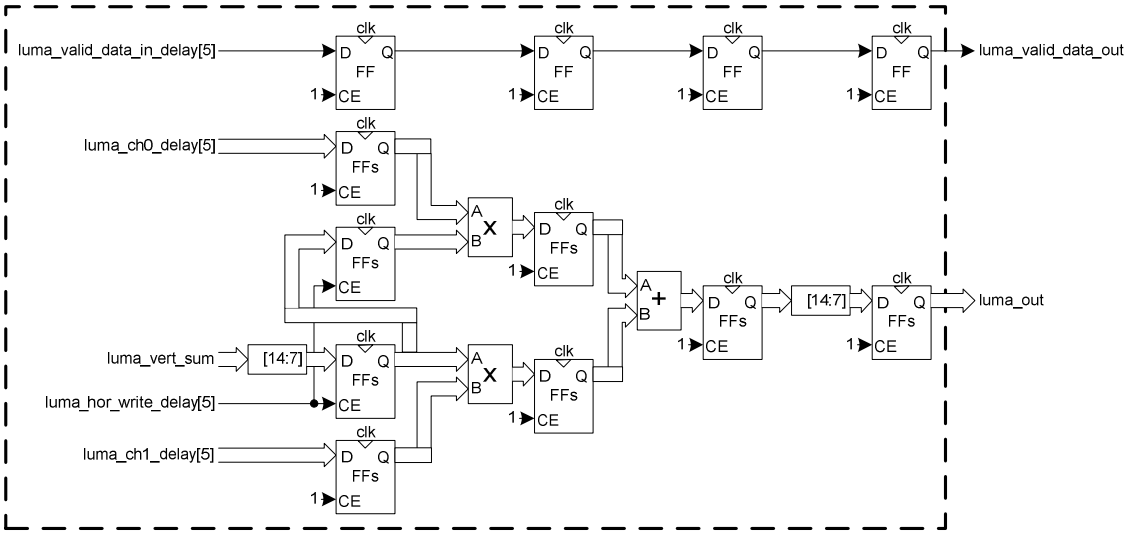


Figura 3-28: Esquema scaler, filtro horizontal para componentes de luma.

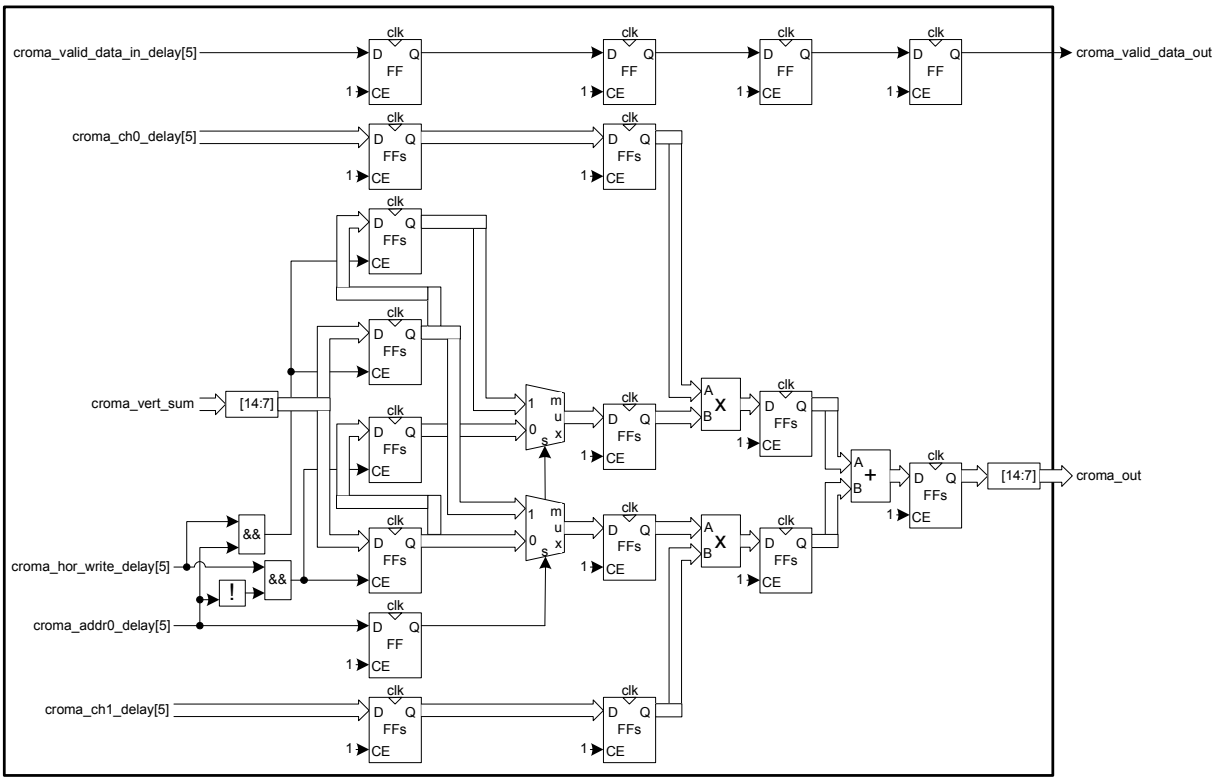


Figura 3-29: Esquema scaler, filtro horizontal para componentes de croma.

Las señales luma_valid_data_out y croma_valid_data_out, son simples retrasos de las señales luma_valid_data_in y croma_valid_data_in correspondientemente.

3.4.2.3.3.1 Módulo line_buffers

Este módulo contiene los retrasos de línea usado por el scaler, en la Tabla 3-16 se muestran la entradas y salidas de este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
write	Entrada	-	Señal de escritura.
addr	Entrada	10	Dirección de escritura y lectura.
data_in	Entrada	8	Dato de escritura.
data_out0	Salida	8	data_in retrasado en dos ciclos de reloj.
data_out1	Salida	8	Dato guardado en primer retraso de línea, en la dirección addr, retrasado en un ciclo.
data_out2	Salida	8	Dato guardado en segundo retraso de línea, en la dirección addr.

Tabla 3-16: Entradas y salidas de line_buffers.

La implementación de este módulo se muestra en el esquema de la Figura 3-30, donde se puede apreciar que se implementa los retrasos de línea utilizando los puertos A y B de un bloque memoria de RAM. La salida de datos del puerto A esta conectada a la entrada de datos del puerto B, la dirección del puerto B es el retraso en un ciclo de la dirección del puerto A, la señal de escritura del puerto B retrasa en un ciclo de reloj en comparación del puerto A y la configuración de los dos puertos en "read first" permite que cuando se escribe un dato en el primer retraso, los datos previamente guardados pasan a ser almacenados por el segundo retraso. Finalmente para obtener los datos para una misma dirección en el mismo ciclo de reloj, se retrasan en dos ciclos la señal data_in y en un ciclo la salida de datos del puerto A.

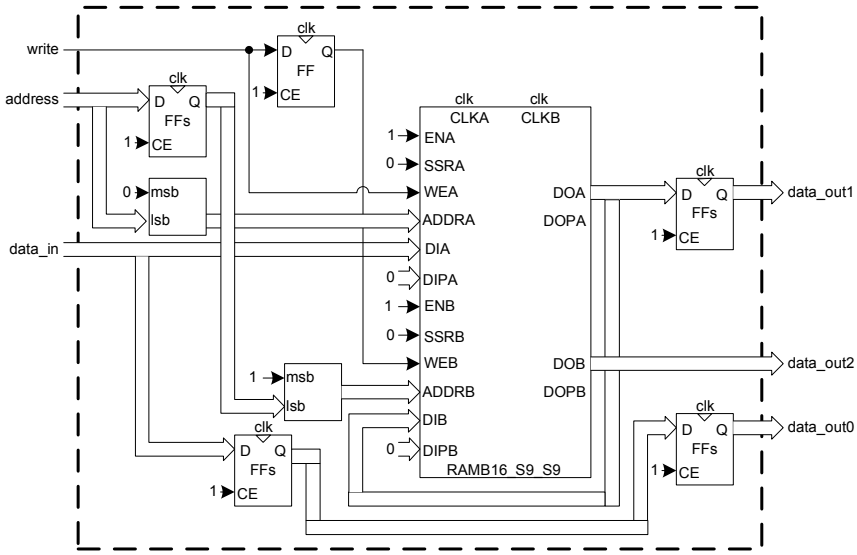


Figura 3-30: Esquema line_buffers.

3.4.2.3.4 Módulo scaler_out

Este módulo es el encargado de almacenar los datos de la imagen redimensionada en memoria, para esto se conecta usando una interfaz NPI con el modulo MPMC. En la Tabla 3-17 se muestran las señales de entrada y salida de este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
Interfaz con modulo 4			
start_address	Entrada	32	Dirección de inicio de buffer de video.
end_address	Entrada	32	Dirección de término de buffer de video.
working	Salida	-	Indica si el módulo está escribiendo en el buffer de video.
Interfaz con scaler			
luma_write	Entrada	-	Señal de escritura de componente de luma en cola.
croma_write	Entrada	-	Señal de escritura de componente de croma en cola.
luma_in	Entrada	8	Datos de entrada de componentes de luma.
croma_in	Entrada	8	Datos de entrada de componentes de croma.
Interfaz con PIM MPMC			
AddrAck	Salida	-	Ver Tabla 2-17.
WrFIFO_Empty	Entrada	-	Ver Tabla 2-17.
Size	Salida	4	Ver Tabla 2-17.
WrFIFO_BE	Salida	4	Ver Tabla 2-17.
WrFIFO_Flush	Salida	-	Ver Tabla 2-17.
AddrReq	Salida	-	Ver Tabla 2-17.
RNW	Salida	-	Ver Tabla 2-17.
WrFIFO_Push	Salida	-	Ver Tabla 2-17.
WrFIFO_Data	Salida	32	Ver Tabla 2-17.
Addr	Salida	32	Ver Tabla 2-17.

Tabla 3-17: Entradas y salidas de scaler_out.

Este módulo se implementa de manera similar a module_2. Las señales RNW, WrFIFO_BE y WrFIFO_Flush tienen valores fijos iguales a 1'b0, 4'hf y 1'b0 respectivamente. Los datos de entrada son guardados en la cola fifo_16_to_32_bits que tiene por función generar palabras de 32. Para leer los datos de la colar se puede usar la señal WrFIFO_Push, ya que esta no tiene latencia en la entrega de datos. Estos elementos son descritos como se muestra en la Figura 3-31.

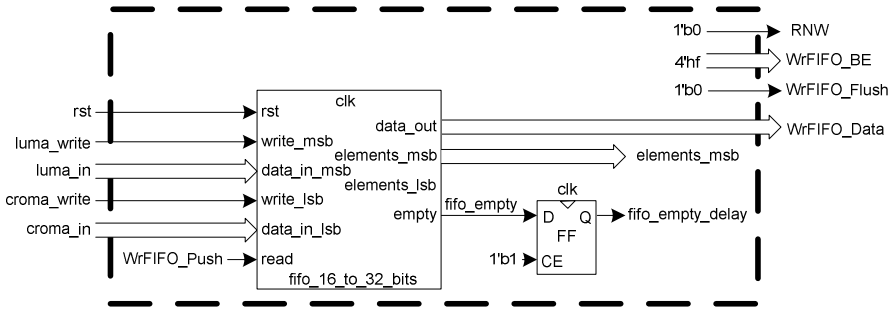


Figura 3-31: Esquema scaler_out, generación de señales RNW, WrFIFO_BE, WrFIFO_Flush y WrFIFO_Data.

La señal working se activa cuando la cola fifo_16_to_32_bits deja de estar vacía estando está señal en 0 y se desactiva cuando no hay más elementos por transferir y ademas la cola del módulo MPMC está vacía. A partir de la señal working se obtiene la señal start, que señala la activación de la señal working retrasada en un ciclo de reloj. Esta señal es usada para reiniciar los contadores elements_to_transfer y Addr. En la Figura 3-32 se muestra la implementación de las señales working y start.

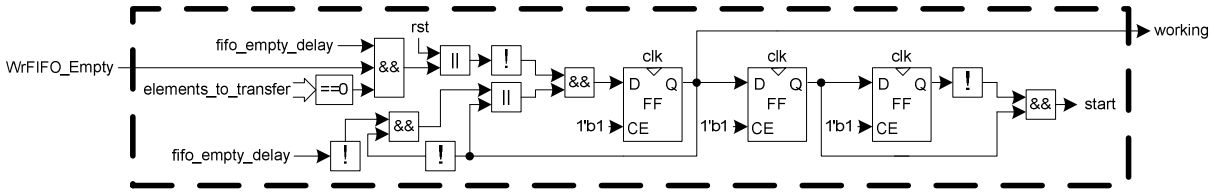


Figura 3-32: Esquema scaler_out, generación de señales mem_working y start.

La señales Addr y Size se generar de forma identica al module_2, en la Figura 3-33 se muestra su implementación.

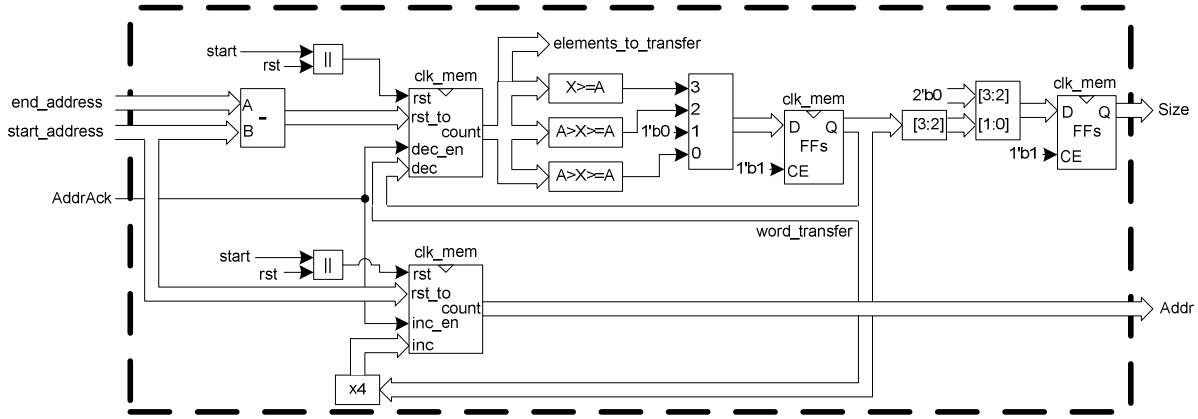


Figura 3-33: Esquema scaler_out, generación de Addr y Size.

La señal WrFIFO_Push se activa cuando la cola fifo_16_to_32_bits, tiene una cantidad de palabras igual o superior al número de palabras a enviar, dado que existe el caso en que word_transfer es 0 y la cola esta vacía, se elimina este caso para la activación haciendo que esta se realice solo cuando el número de palabras a transferir es distinto de 0. Para que la transferencia de palabras sea igual al número deseado se utiliza el contador push_counter, el que incrementa su valor mientras WrFIFO_Push este activada y se reinicia luego de transmitir la última palabra, condición que es usada para desactivar WrFIFO_Push. La señal AddrReq tambien hace uso de este contador activándose cuando se termina de enviar el penúltimo dato. Debido a que esta condición no considera las transferencias de un elemento, se incorpora este caso activando la señal cuando word_transfer es uno y se activa WrFIFO_Push. En la Figura 3-34 se muestra el diagrama usado para implementación de las señales WrFIFO_Push y AddrReq.

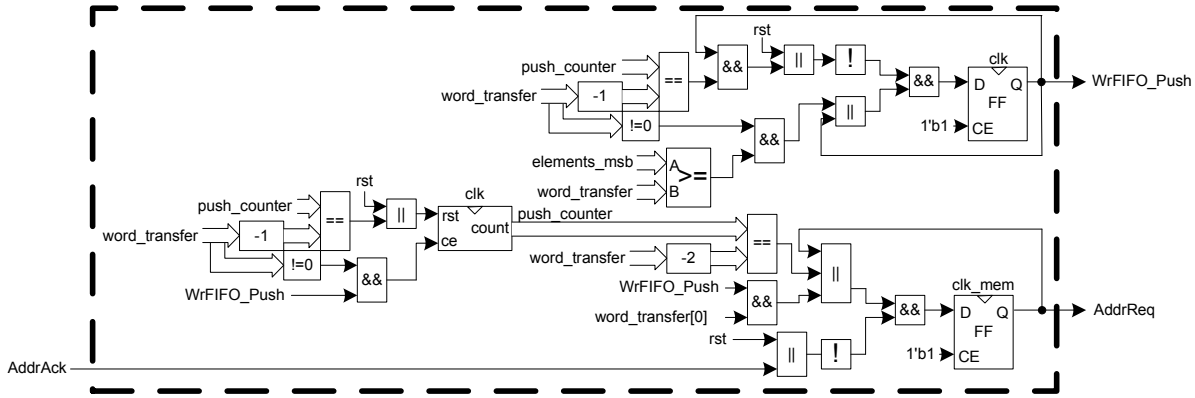


Figura 3-34: Esquema scaler_out, generación de WrFIFO_Push y AddrReq

3.4.2.4 Módulo module_4

Este módulo es el encargado de calcular las direcciones de inicio y término de los espacios de memoria utilizados en la SDRAM DDR2 y administrar su uso en los otros módulos del sistema. En Tabla 3-18 se muestran las señales de entrada y salida de este modulo y en la Tabla 3-19 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
Entrada de Video 0			
Interfaz con module_1			
Locked_v0	Entrada	-	Indica estabilidad de la entrada de video.
acpl_v0	Entrada	12	Número de componentes activos por línea.
alpf_v0	Entrada	12	Número de líneas activas por campo (o cuadro).
Interfaz con module_2			
m2_v0_mw	Entrada	-	Indicador de trabajo en memoria.
m2_v0_mwo	Salida	-	Indica el buffer a utilizar por modulo.
m2_v0_sa	Salida	32	Dirección de inicio de buffer.
m2_v0_ea	Salida	32	Dirección de término de buffer.
Interfaz con module_3			
m3_si_v0_mw	Entrada	-	Indicador de trabajo en memoria en entrada del redimensionado.
m3_si_v0_mwo	Salida	-	Indica el buffer a utilizar por modulo en entrada de redimensionado.
m3_si_v0_wc	Salida	-	Indica si el buffer de entrada al redimensionado tiene datos validos.
m3_si_v0_sa	Salida	32	Dirección de inicio de buffer de entrada de redimensionado.
m3_si_v0_ea	Salida	32	Dirección de término de buffer de entrada de redimensionado.
m3_so_v0_mw	Entrada	-	Indicador de trabajo en memoria en salida del redimensionado.
m3_so_v0_mwo	Salida	-	Indica el buffer a utilizar por modulo en salida del redimensionado.
m3_so_v0_sa	Salida	32	Dirección de inicio de buffer en salida del redimensionado.
m3_so_v0_ea	Salida	32	Dirección de término de buffer en salida del redimensionado.
Interfaz con module_5			
m5_v0_mw	Entrada	-	Indicador de trabajo en memoria.
m5_v0_mwo	Salida	-	Indica el buffer a utilizar por modulo.
m5_v0_wc	Salida	-	Indica si el buffer tiene datos validos.
m5_v0_sa	Salida	32	Dirección de inicio de buffer.
Información de imagen redimensionada			
v0_so_acpl	Salida	16	Número de componentes activos por línea.
v0_so_alpf	Salida	16	Número de líneas activas por campo (o cuadro).

Tabla 3-18: Entradas y salidas de module_4.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
Entrada de Video 1			
Interfaz con module_1			
locked_v1	Entrada	-	Indica estabilidad de la entrada de video.
acpl_v1	Entrada	12	Número de componentes activos por línea.
alpf_v1	Entrada	12	Número de líneas activas por campo (o cuadro).
Interfaz con module_2			
m2_v1_mw	Entrada	-	Indicador de trabajo en memoria.
m2_v1_mwo	Salida	-	Indica el buffer a utilizar por modulo.
m2_v1_sa	Salida	32	Dirección de inicio de buffer.
m2_v1_ea	Salida	32	Dirección de término de buffer.
Interfaz con module_3			
m3_si_v1_mw	Entrada	-	Indicador de trabajo en memoria en entrada del redimensionado.
m3_si_v1_mwo	Salida	-	Indica el buffer a utilizar por modulo en entrada de redimensionado.
m3_si_v1_wc	Salida	-	Indica si el buffer de entrada al redimensionado tiene datos validos.
m3_si_v1_sa	Salida	32	Dirección de inicio de buffer de entrada de redimensionado.
m3_si_v1_ea	Salida	32	Dirección de término de buffer de entrada de redimensionado.
m3_so_v1_mw	Entrada	-	Indicador de trabajo en memoria en salida del redimensionado.
m3_so_v1_mwo	Salida	-	Indica el buffer a utilizar por modulo en salida del redimensionado.
m3_so_v1_sa	Salida	32	Dirección de inicio de buffer en salida del redimensionado.
m3_so_v1_ea	Salida	32	Dirección de término de buffer en salida del redimensionado.
Interfaz con module_5			
m5_v1_mw	Entrada	-	Indicador de trabajo en memoria.
m5_v1_mwo	Salida	-	Indica el buffer a utilizar por modulo.
m5_v1_wc	Salida	-	Indica si el buffer tiene datos validos.
m5_v1_sa	Salida	32	Dirección de inicio de buffer.
Información de imagen redimensionada			
v1_so_acpl	Salida	16	Número de componentes activos por línea.
v1_so_alpf	Salida	16	Número de líneas activas por campo (o cuadro).

Tabla 3-18: Entradas y salidas de module_4 (continuación).

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
Señales de datos para entrada de video 0			
HORIZONTAL_MULTIPLIER_SCALING_VIDEO_0	4'd1	1 a 8	Multiplicador de dimensión horizontal.
HORIZONTAL_DIVISOR_SCALING_VIDEO_0	4'd1	1 a 8	Divisor de dimensión horizontal.
VERTICAL_MULTIPLIER_SCALING_VIDEO_0	4'd1	1 a 8	Multiplicador de dimensión vertical.
VERTICAL_DIVISOR_SCALING_VIDEO_0	4'd1	1 a 8	Multiplicador de dimensión vertical.
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_0	32'h00000000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	32'h00400000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_0	32'h00800000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	32'h00c00000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.

Tabla 3-19: Parámetros de modulo module_4.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
Señales de datos para entrada de video 1			
HORIZONTAL_MULTIPLIER_SCALING_VIDEO_1	4'd1	1 a 8	Multiplicador de dimensión horizontal.
HORIZONTAL_DIVISOR_SCALING_VIDEO_1	4'd1	1 a 8	Divisor de dimensión horizontal.
VERTICAL_MULTIPLIER_SCALING_VIDEO_1	4'd1	1 a 8	Multiplicador de dimensión vertical.
VERTICAL_DIVISOR_SCALING_VIDEO_1	4'd1	1 a 8	Multiplicador de dimensión vertical.
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	32'h01000000	Números de 32 bits	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	32'h01400000	Números de 32 bits	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	32'h01800000	Números de 32 bits	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	32'h01c00000	Números de 32 bits	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.

Tabla 3-19: Parámetros de modulo module_4 (continuación).

El funcionamiento de este módulo parte con el cálculo de las nuevas dimensiones de las entradas de video luego de ser procesadas por module_3, las cuales son calculadas por el módulo addr_gen. Este modulo además es el encargado de calcular las direcciones de inicio y término de los espacios de memoria utilizados por los módulos que tienen una interfaz de trabajo con el modulo MPMC. En la Figura 3-35 se ilustran las conexiones del modulo addr_gen, también en esta imagen se puede apreciar que este módulo obtiene la información de los factores de redimensionamiento por parte de bancos de flipflops los cuales tienen un valor fijo dado por parámetros del módulo. Estos flipflop son definidos para que en futuras mejoras al sistema, estos parámetros sean modificables de forma externa al módulo.

La otra función principal de este módulo es la administración de los espacios de memoria utilizados. Para la realización de esto se utilizan las señales memory_working_on o abreviado en los nombres de las señales como mwo. Todas las señales que tienen incorporada esta palabra en su nombre indican el espacio de memoria donde debe trabajar el módulo. El cambio del espacio de memoria usado por un módulo se dará siempre que el modulo que lee de los buffers no esté trabajando y el módulo que escribe en el buffer deja de trabajar con la memoria. Otro uso dado a estas señales es de definir una señal de existencia de contenido, la que es activada con el primer cambio de la mwo y es desactivada si se pierde la estabilidad de la señal. Esta señal es caracterizada por las letras wc, en los nombres de las señales.

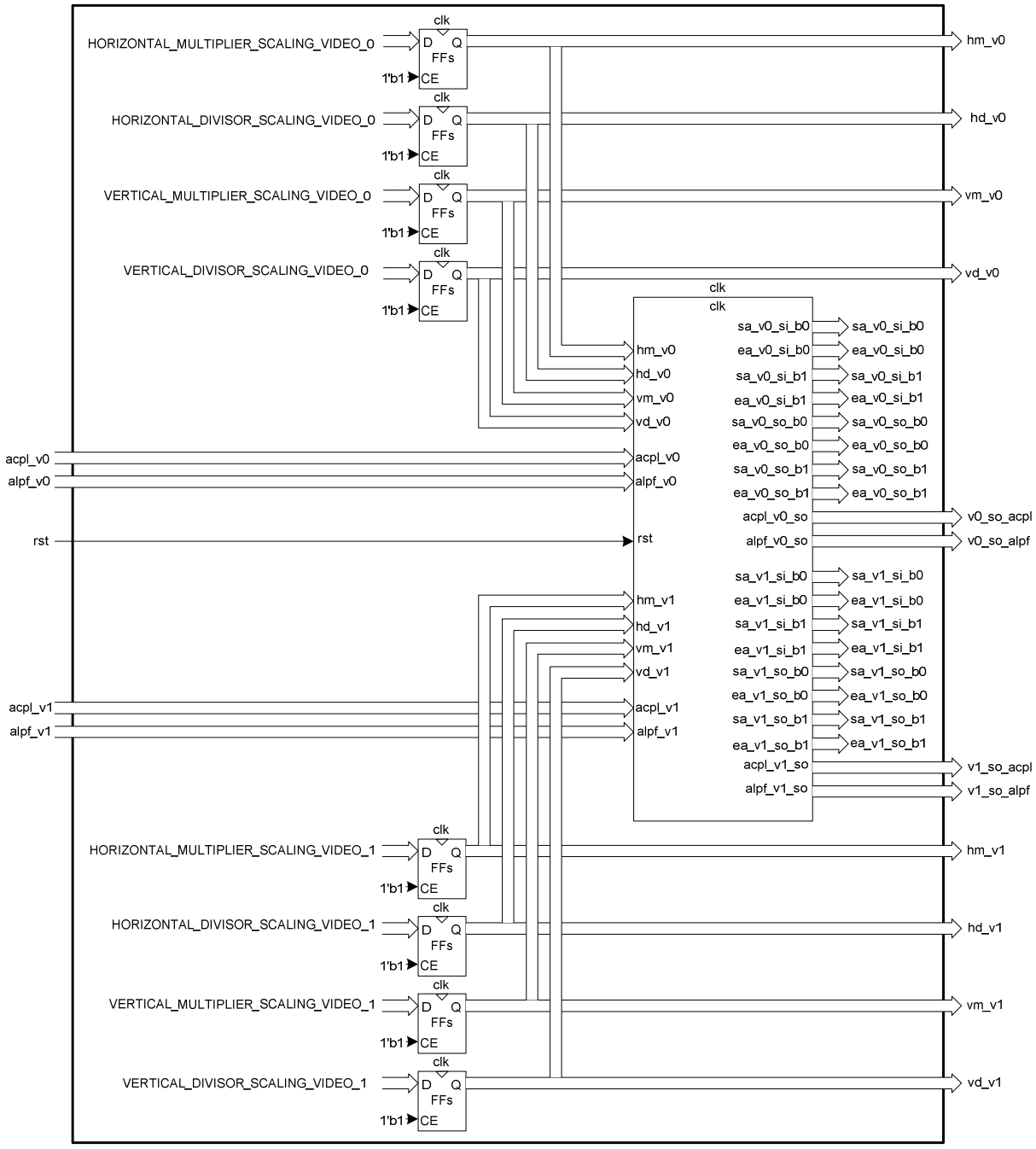


Figura 3-35: Esquema modulo module_4 parte 1.

En la Figura 3-36 se muestra el esquema de los elementos utilizados para la generación de las señales mwo y wc, y la selección de las direcciones de inicio y término de los módulos con una interfaz NPI encargados de procesar la entrada de video 0. La misma configuración es utilizada para las señales de la entrada de video 1.

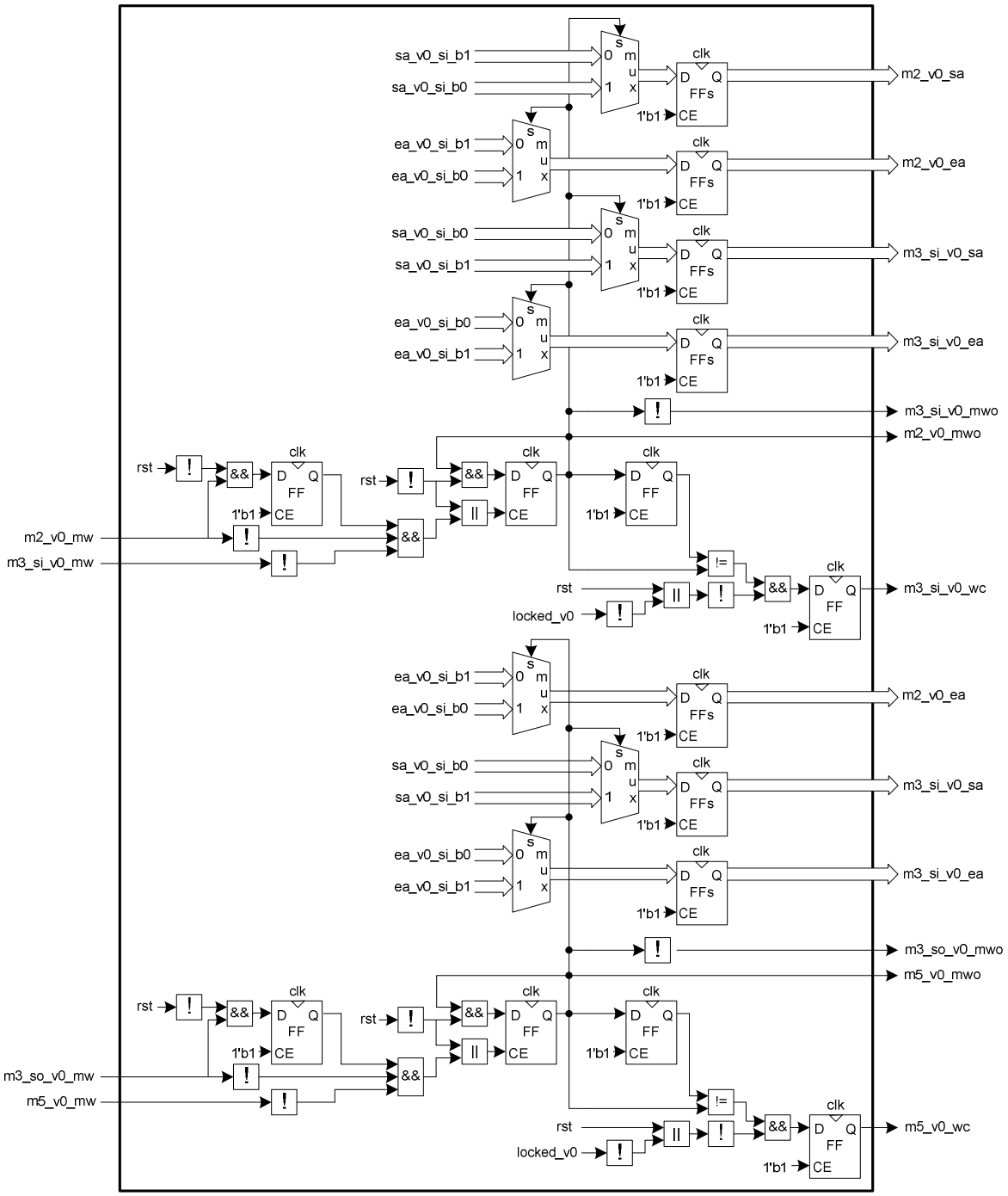


Figura 3-36: Esquema modulo module_4 parte 2.

3.4.2.4.1 Módulo addr_gen

Este módulo es el encargado de calcular las nuevas dimensiones de las entradas de video tras ser procesadas por el modulo module_3 y a partir de estas generar la direcciones de inicio y término de los espacio de memoria utilizados por el modulo MPMC. En la Tabla 3-20 se indican las señales de entrada y salida de este módulo, y en la Tabla 3-21 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rst	Entrada	-	Señal de reinicio.
clk	Entrada	-	Señal de reloj.
Señales de datos de entrada de video 0			
acpl_v0	Entrada	12	Número de componentes activos por línea.
alpf_v0	Entrada	12	Número de líneas activas por campo (o cuadro).
hm_v0	Entrada	4	Multiplicador de dimensión horizontal.
hd_v0	Entrada	4	Divisor de dimensión horizontal.
vm_v0	Entrada	4	Multiplicador de dimensión vertical.
vd_v0	Entrada	4	Divisor de dimensión vertical.
sa_v0_si_b0	Salida	32	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
sa_v0_si_b1	Salida	32	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v0_si_b0	Salida	32	Dirección de término del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v0_si_b1	Salida	32	Dirección de término del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
sa_v0_so_b0	Salida	32	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
sa_v0_so_b1	Salida	32	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.
ea_v0_so_b0	Salida	32	Dirección de término del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v0_so_b1	Salida	32	Dirección de término del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
v0_so_acpl	Salida	16	Número de componentes activos por línea de imagen redimensionada.
v0_so_alpf	Salida	16	Número de líneas activas por campo (o cuadro) de imagen redimensionada.
Señales de datos de entrada de video 1			
acpl_v1	Entrada	12	Número de componentes activos por línea.
alpf_v1	Entrada	12	Número de líneas activas por campo (o cuadro).
hm_v1	Entrada	4	Multiplicador de dimensión horizontal.
hd_v1	Entrada	4	Divisor de dimensión horizontal.
vm_v1	Entrada	4	Multiplicador de dimensión vertical.
vd_v1	Entrada	4	Divisor de dimensión vertical.
sa_v1_si_b0	Salida	32	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
sa_v1_si_b1	Salida	32	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v1_si_b0	Salida	32	Dirección de término del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v1_si_b1	Salida	32	Dirección de término del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
sa_v1_so_b0	Salida	32	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
sa_v1_so_b1	Salida	32	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.
ea_v1_so_b0	Salida	32	Dirección de término del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
ea_v1_so_b1	Salida	32	Dirección de término del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
v1_so_acpl	Salida	16	Número de componentes activos por línea de imagen redimensionada.
v1_so_alpf	Salida	16	Número de líneas activas por campo (o cuadro) de imagen redimensionada.

Tabla 3-20: Entradas y salidas de modulo addr_gen.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
Señales de datos para entrada de video 0			
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_0	32'h021F0000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	32'h023B2000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_0	32'h02574000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	32'h02736000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.
Señales de video para entrada de video 1			
START_ADDR_VIDEO_1_SCALING_IN_BUFFER_0	32'h028F8000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_1_SCALING_IN_BUFFER_1	32'h02ABA000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como entrada al modulo de redimensionamiento.
START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_0	32'h02C7C000	Cualquiera	Dirección de inicio del primer espacio de memoria utilizado como salida al modulo de redimensionamiento.
START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_1	32'h02E3E000	Cualquiera	Dirección de inicio del segundo espacio de memoria utilizado como salida al modulo de redimensionamiento.

Tabla 3-21: Parámetros de modulo addr_gen.

En una señal de video, el módulo module_1 calcula las dimensiones de entrada pasado un cuadro del video y al siguiente cuadro la señal locked activa. El cálculo de las nuevas dimensiones debe realizarse antes que la señal locked sea activada, y luego estas se mantienen mientras la señal se mantenga estable. Por esta razón este módulo tiene un amplio margen tiempo para determinar las nuevas dimensiones, lo que lleva al diseño de este módulo el cual se esquematiza en la Figura 3-37, donde se prioriza disminuir la cantidad lógica usada sobre minimizar el tiempo de procesamiento. Esto se realiza limitando la lógica de procesamiento a 2 multiplicadores y un divisor, y multiplexando las señales que entran en estos dispositivos. Cada uno de los dispositivos es procesa cuatro buses distintos por lo que genera un latencia inicial de cuatro ciclos de reloj. Se añade a esta latencia, la incorporada por lo módulos divider0 y multiplier0, los que corresponden a IPcores de Xilinx que fueron explicados en el capítulo 0. En la Tabla 3-22, se muestra la configuración usada por estos módulos. Con estas configuraciones el multiplicador tiene una latencia de 4 ciclos y el divisor de 18 ciclos de reloj. Finalmente el proceso que realizan los datos parte con la multiplicación de las dimensiones iníciales de las entradas con los factores multiplicativos, esta etapa es realizada por un multiplicador generado con LUTs por lo que tiene latencia 0, por lo que considerando los bancos de flipflops usados y la multiplicación, esta etapa toma 6 ciclos de reloj en computar todos los valores post-multiplicación o pm.

Multiplier0		Divider0	
Parámetros de configuración	**Opción o valor**	**Parámetros de configuración**	**Opción o valor**
Tipo de Multiplicador	Paralelo	Tipo de Algoritmo	Radix2
Tipo de datos puerto A	Sin signo	Ancho de dividendo y cociente	16
Ancho de datos de puerto A	16	Ancho de divisor	4
Tipo de datos puerto B	Sin signo	Tipo de resto	Residuo
Ancho de datos de puerto B	16	Tipo de operandos	Sin signo
Construcción de multiplicador	LUTs	Ciclos por división	1
Etapas Pipeline	4		

Tabla 3-22: Configuración módulos multiplier0 y divider0.

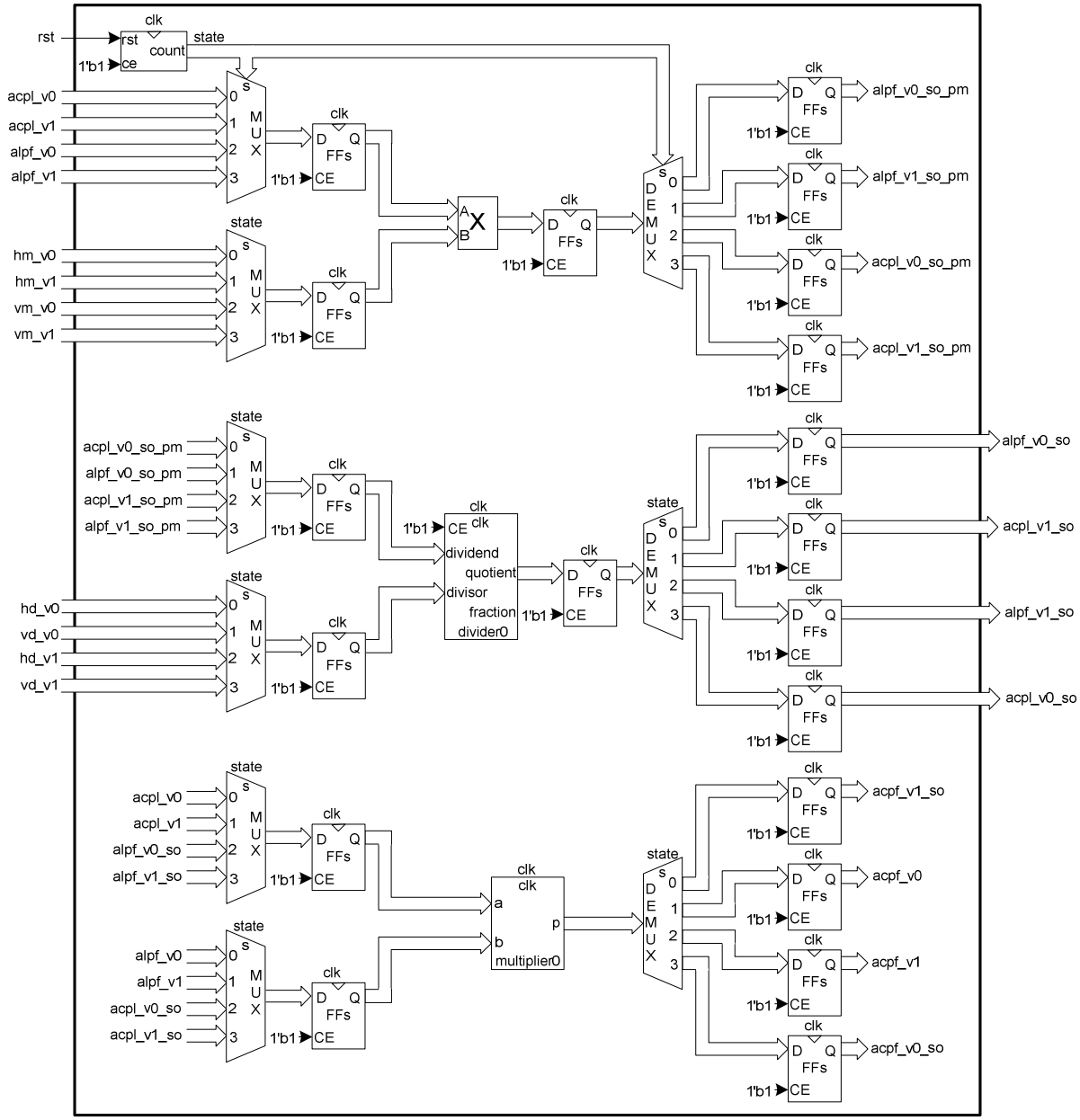


Figura 3-37: Esquema modulo addr_gen parte 1.

La siguiente fase se encarga de dividir los valores antes obtenidos por los divisores de dimensión, usando el IPcore divider0 con lo que esta fase se demora 24 ciclos de reloj en estar terminada. Los resultados de esta fase corresponden a las nuevas dimensiones horizontales y verticales de las entradas. La siguiente etapa consiste en la multiplicación entre las dimensiones horizontales y verticales de los tamaños nuevos y los originales de las imágenes, obteniéndose el número de componentes activos por cuadro (o campo). Esta última etapa toma un total de 9 ciclos en ser completada. Finalmente los valores obtenidos son sumados con los parámetros de la dirección inicial, obteniéndose las direcciones de término de los buffers, lo que se muestra en la Figura 3-38. Como se muestra en la imagen existe un último flipflop después de la suma, por lo que la latencia total para obtener las direcciones de término es de 40 ciclos de reloj. Esta cifra es insignificante considerando que una línea de video NTSC tiene 1716 componentes, por lo que no existirán problemas por la demora del cómputo.

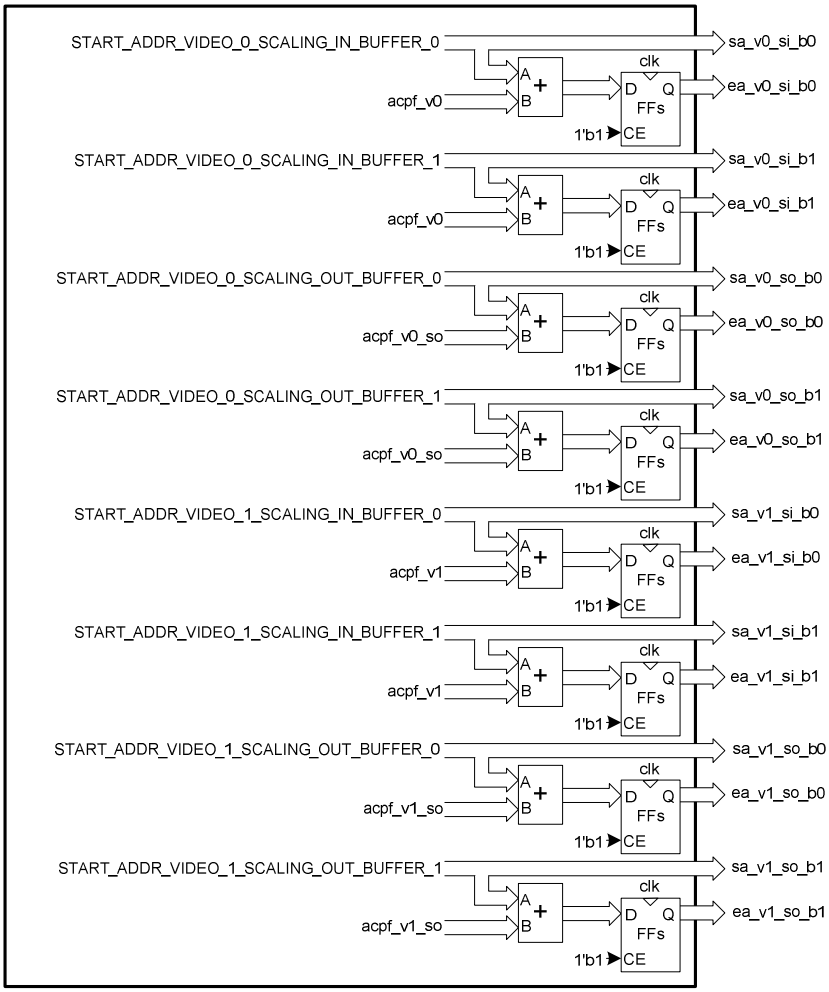


Figura 3-38: Esquema modulo addr_gen parte 2.

3.4.2.5 Módulo module_5

Este módulo junto con module_6 son los encargados de integrar las diferentes señales de video en una única señal de salida, a través de la obtención ordenada de los datos desde los distintos buffers de video. Este módulo tiene además la función de generar la señal de salida del sistema la cual contiene los datos de video y la codificación de temporización. En la Tabla 3-23 se muestran las entradas y salidas de este bloque y en la Tabla 3-24 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk_mem	Entrada	-	Señal de reloj para la sincronización de señales MPMC.
clk_video	Entrada	-	Señal de reloj para la sincronización de señales de video.
rst	Entrada	-	Señal de reinicio.
Interfaz con module_4			
v0_so_acpl	Entrada	16	Número de componentes activos por línea de imagen redimensionada de la entrada de video 0.
v0_so_alpf	Entrada	16	Número de líneas activas por cuadro de imagen redimensionada de la entrada de video 0.
v0_wc	Entrada	-	Indica la existencia de datos en el buffer de video, que contiene la imagen redimensionada de la entrada de video 0.
v1_so_acpl	Entrada	16	Número de componentes activos por línea de imagen redimensionada de la entrada de video 1.
v1_so_alpf	Entrada	16	Número de líneas activas por cuadro de imagen redimensionada de la entrada de video 1.
v1_wc	Entrada	-	Indica la existencia de datos en el buffer de video, que contiene la imagen redimensionada de la entrada de video 1.
Interfaz con module_6			
end_f	Salida	-	Indica término de campo o cuadro.
send	Salida	-	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
rst_f	Salida	-	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
rst_l	Salida	-	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
port	Salida	-	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
active	Salida	-	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
count	Salida	10	Interfaz de comando unidireccional con module_6. Ver Tabla 3-25.
Interfaz con modulo MPMC			
RdFIFO_Empty	Entrada	-	Ver Tabla 2-17.
RdFIFO_Latency	Entrada	2	Ver Tabla 2-17.
RdFIFO_Data	Salida	31	Ver Tabla 2-17.
RdFIFO_Pop	Salida	-	Ver Tabla 2-17.
RdFIFO_Flush	Salida	-	Ver Tabla 2-17.
Salida de sistema			
output_video	Salida	8	Salida de video del sistema.

Tabla 3-23: Entradas y salidas module_5.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
SCANNING	"interlaced"	"interlaced" , "progresive"	Caracteriza el tipo de escaneo que tiene la salida de video.
RASTER	"sim"	"sim" , "576",	Indica que raster de video es usado por la salida de video.
PICT0_ENABLE	1	0,1	Habilita el uso de imagen 0.
PICT0	"in0"	"in0","in1"	Identifica que entrada de video es usada para la imagen 1.
PICT0_TL_X	0	Números de 9 bits.	Posición horizontal dentro de la imagen de salida, donde se ubica el vértice superior izquierdo de la imagen 0.
PICT0_TL_Y	0	Números de 9 bits.	Posición vertical dentro de la imagen de salida, donde se ubica el vértice superior izquierdo de la imagen 0.
PICT0_PRIO	0	0,1	Identifica la prioridad de la imagen 0 al estar sobrepuesta con otras imágenes.
PICT1_ENABLE	1	0,1	Habilita el uso de imagen 1.
PICT1	"in1"	"in0","in1"	Identifica que entrada de video es usada para la imagen 1.
PICT1_TL_X	20	Números de 9 bits.	Posición horizontal dentro de la imagen de salida, donde se ubica el vértice superior izquierdo de la imagen 1.
PICT1_TL_Y	10	Números de 9 bits.	Posición vertical dentro de la imagen de salida, donde se ubica el vértice superior izquierdo de la imagen 1.
PICT1_PRIO	0	0,1	Identifica la prioridad de la imagen 1 al estar sobrepuesta con otras imágenes.

Tabla 3-24: Parámetros module_5.

El funcionamiento de este bloque se separa en dos partes, la primera encargada de realizar peticiones de transferencia al modulo MPMC de una forma tal que los elementos almacenados en la cola RdFIFO estén listos para ser usado por la salida de video. La segunda es la encargada de generar la salida de video usando la estructura de un *raster* común.

La generación de peticiones, se inicia con la definición de los intervalos de blanqueo (horizontales y verticales) y de imagen activa (tamaño de líneas activas y cuadro o campos activos) generando así la estructura de la señal de salida. Luego se definen imágenes dentro de los intervalos activos de la estructura ya definidos. Estas imágenes corresponden a marcos que pueden albergar a cualquiera de las entradas de video redimensionadas. Así se obtiene una estructura como la mostrada en la Figura 3-39. A continuación se define un puntero que recorrerá la estructura de izquierda a derecha y de arriba a abajo. Este puntero será usado para definir cada punto de la estructura, dentro de las categorías: inactivo y activo. Dentro de la categoría de puntos activos se define si el punto pertenece a alguna de las imágenes y en caso de pertenecer se especifica a cuales. Debido a que no se pueden mostrar dos imágenes simultáneamente en un mismo punto, se utiliza un sistema de prioridades para definir qué imagen es mostrada. A cada imagen se le asigna una prioridad, siendo 0 la más alta y disminuyendo al aumentar el número. Cuando varias imágenes se superponen teniendo la

misma prioridad, se otorga a la imagen de menor índice, la prioridad sobre las otras. Así una imagen puede tener dos tipos de puntos los mostrados en la salida de video y los que no.

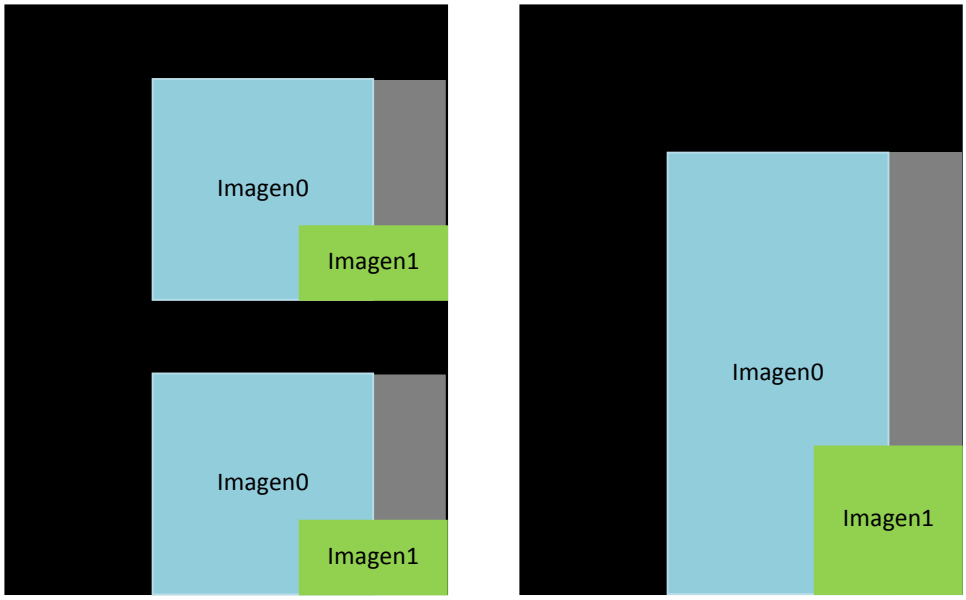


Figura 3-39: Estructura generada con para la petición de transferencia, para imagen a) entrelazada, b) progresiva.

Para realizar una petición de transferencia se debe manejar la señal Addr de la interfaz NPI, esta señal a sido implementada en otros módulos con un contador que incrementa su valor cada vez que se arbitra una petición, pero al existir intervalos de datos que no son usados en la salida, se debe incluir un incremento en la dirección igual al tamaño de estos intervalos, para que la dirección usada en las peticiones corresponda al conjunto de datos que se desea obtener. Para solucionar este hecho se plantea un procedimiento donde en una línea determinada que tiene superposición de imágenes se cuenta a partir del primer elemento de esta línea hasta el último elemento del mismo tipo (punto mostrado o no) y luego se encontrara elementos del otro tipo los cuales son contados desde su primera aparición hasta el último encontrado en la línea. Dependiendo del tipo de dato, se realizan distintas operaciones; en caso de ser un conjunto de puntos mostrados, se realiza el procedimiento de petición de transferencia, activando la señal AddrReq, hasta que se arbitra la petición y es incrementado Addr en el tamaño de la transferencia, y así sucesivamente hasta que se realiza la petición de todos los elementos contados. En caso de ser un conjunto de puntos no mostrados, solo se debe incrementar la dirección en el número de elementos del conjunto. Además existe la situación parte de la línea de la imagen no es completamente contenida en la parte activa de la salida, cuando ocurre esto se debe incrementar la señal Addr en un número igual a los elementos no contenidos en el cuadro. Para tratar esta situación se lleva un conteo de los elementos de la línea que no han

sido contabilizados como mostrados o no, al terminar la línea el valor de este contador es usado para incrementar Addr y luego es reiniciado a su máximo valor.

Hasta el momento no se ha hecho referencia si el video de salida es entrelazado o progresivo y las diferencias que provocaría en el proceso. Como se explicó en un capítulo previo el video entrelazado toma las líneas pares para generar el campo 0 y las líneas impares para generar el campo 1. Para incorporar esta característica a las peticiones de transferencia se deben realizar dos cosas, la primera es definir el máximo valor del contador de elementos no contabilizados como la cantidad de palabras contenida en dos líneas de la imagen. De esta forma cuando una línea acaba no se continuará con la línea contigua si no con la siguiente. Para video progresivo el valor máximo del contador será igual a los elementos contenidos por una línea de la imagen. La segunda diferencia en el proceso ocurre cuando se reinicia la dirección, en video progresivo se reinicia siempre al valor de la dirección de inicio del buffer de video cuando se inicia un nuevo cuadro de líneas activas, para video entrelazado la dirección de inicio depende del campo que se va escribir, en caso de ser el campo 0 se reinicia la dirección a la primera línea del buffer, cuando es el campo 1 se reinicia a la dirección de la segunda línea.

La implementación del proceso descrito en los párrafos anteriores es mostrada en la Figura 3-40, acá se parte con la definición del puntero por parte de los módulos module_5_p1 y module_5_p2, los que dan las posiciones horizontal y vertical respectivamente, utilizando las señales h_counter y v_counter_mem. Estos bloques tienen además la función de entregar la característica del punto de actividad o inactividad a través de las señales h_av y v_av_mem, de esta forma se genera la estructura básica del cuadro. La incorporación de las imágenes a la estructura del cuadro es realizada con el dispositivo module_5_p3, el cual con los datos de la posición del puntero y las posiciones de los vértices de una imagen, entrega si el punto corresponde a un elemento de la imagen. En el diseño de la Figura 3-40 se muestra la utilización de dos de estos módulos, debido a que se define una estructura con dos imágenes. El último paso para la definición completa de la estructura, es el tratamiento de los puntos que pertenecen a superposiciones de imágenes, esta tarea es realizada por el bloque module_5_p4.

Usando las características definidas por los bloques ya descritos, el dispositivo module_5_p5, realiza el recuento de los puntos mostrados y no mostrados, y la identificación de términos de línea para cada imagen, junto con la identificación del inicio de un cuadro o campo activo y el término de este. Esta información es multiplexada en el tiempo para formar una interfaz de comandos, compuesta por las señales send, rst_f, port, rst_l, active y count, generada por el bloque module_5_p6. Esta interfaz considera el envió de un comando con la activación de send, y los comandos enviados son los que se muestran en la Tabla 3-25. El proceso de decodificación y realización de comando es realizado posteriormente por module_6.

send	Rst_f	port	Rst_l	active	count	Comando
0	X	X	X	X	XXXX	Indica inactividad de interfaz.
1	1	X	X	X	XXXX	Comando de inicio de cuadro o campo.
1	0	0	1	X	XXXX	Comando de término de línea de imagen 0.
1	0	1	1	X	XXXX	Comando de término de línea de imagen 1.
1	0	0	0	0	COUNT	Comando de incremento de dirección de imagen 0 en un valor igual a COUNT.
1	0	1	0	0	COUNT	Comando de incremento de dirección de imagen 1 en un valor igual a COUNT.
1	0	0	0	1	COUNT	Comando de petición de transferencia de elementos de imagen 0 de COUNT palabras.
1	0	1	0	1	COUNT	Comando de petición de transferencia de elementos de imagen 1 de COUNT palabras.

Tabla 3-25: Comandos generados por module_5_p6.

La segunda parte de la funcionalidad de este módulo, correspondiente a la generación del video de salida, parte sobre la misma base de la definición de una estructura y un puntero que la recorre identificando tipos de puntos. Para esta parte se requieren las identificar características del punto: si es activo o inactivo, y en caso de ser activo si corresponde a un componente de alguna de las imagenes, el campo al que pertenece, si corresponde a código de temporización y el canal al que pertenece (Y, Cb, Cr).Utilizando estas características de los puntos se logra generar la señal de salida.

La implementación de esta sección también se muestra en la Figura 3-40. El puntero usado es generado por los bloques module_5_p1 y module_5_p2, siendo este puntero diferente al usado en la generación de peticiones, en su posición vertical. El uso de dos punteros diferentes se debe a la necesidad de almacenar con antelación los datos de las imágenes en la cola RdFIFO, para que estos estén disponibles cuando sean requeridos en la generación de la salida de video. En consecuencia, el puntero queda definido por las señales h_counter y v_counter_output.

La caracterización del puntero es realizada por los dispositivos module_5_p1, module_5_p2, module_5_p3 y module_5_p8. module_5_p1 tiene la función de caracterizar el componente como activo o inactivo dado su posición horizontal, además identifica si el componente corresponde a código de temporización SAV o EAV. module_5_p2 cataloga el punto como activo o inactivo dado su posición vertical, esta información junto con la entregada por module_5_p1 permite determinar si el componente es activo o inactivo. Los bloques module_5_p3 determinan la pertenencia del punto a alguna de las imágenes, si alguno de estos identifica el punto como imagen. Por último module_5_p8 es el encargado de determinar a qué campo pertenece el elemento.

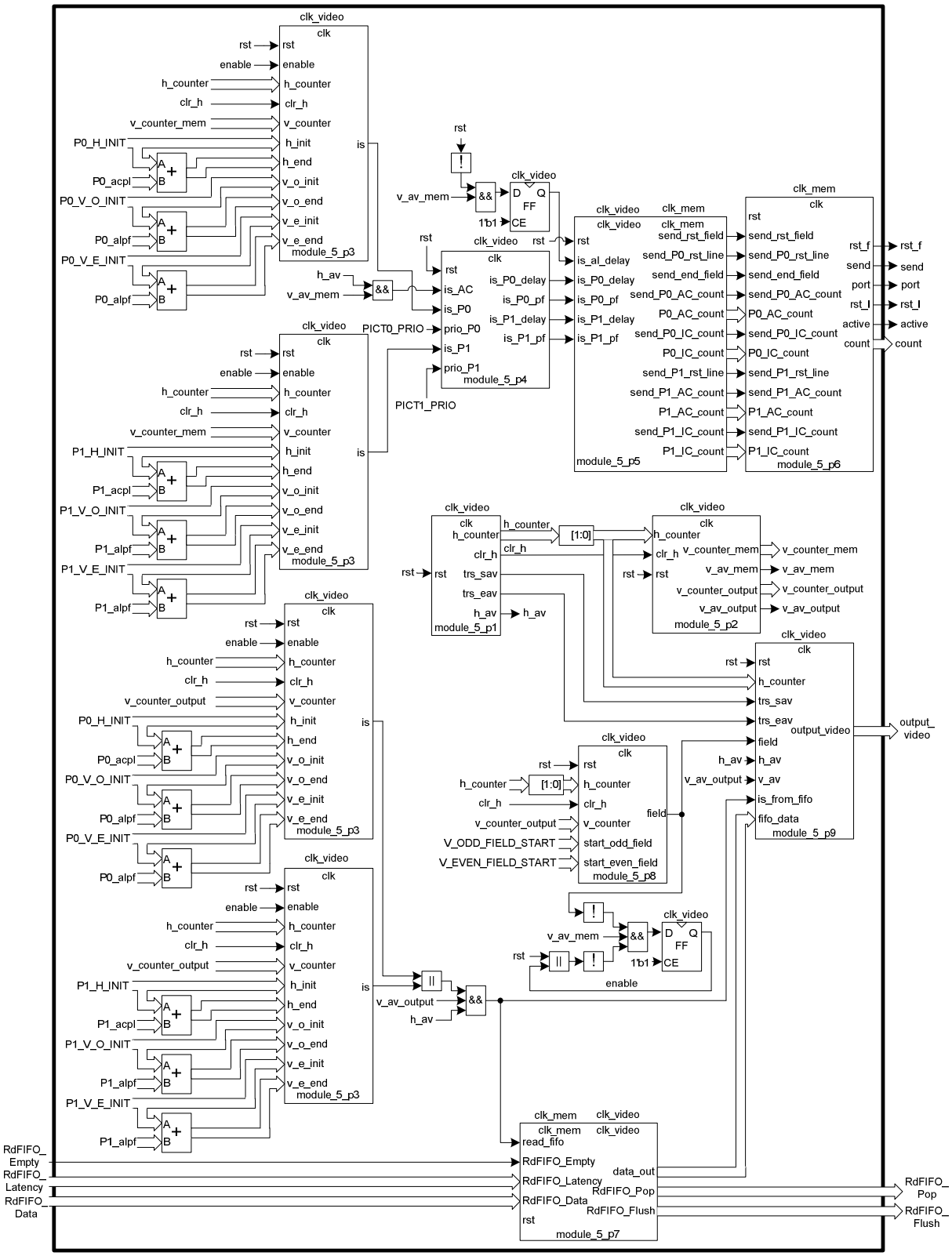


Figura 3-40: Esquema de module_5.

En el proceso es usado el dispositivo module_5_p7 para leer la información almacenada en RdFIFO, este bloque permite guardar esta información y transformarla de palabras de 4 bytes a componentes de 1 byte.

Finalmente haciendo usado de todas las características del puntero, module_5_p9 genera la salida de video.

3.4.2.5.1 Módulo module_5_p1

Este módulo es el encargado de generar la posición horizontal del componente, determinar si corresponde a un elemento activo o inactivo, y pertenece a uno de los códigos de temporización. En la Tabla 3-26 se indican las entradas y salidas de este módulo y en la Tabla 3-27 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
h_counter	Salida	12	Indica la posición horizontal del componente.
clr_h	Salida	-	Señal de control que identifica el reinicio de h_counter.
trs_eav	Salida	-	Indica si el componente pertenece al código de temporización EAV.
trs_sav	Salida	-	Indica si el componente pertenece al código de temporización SAV.
h_av	Salida	-	Indica si el componente es activo.

Tabla 3-26: Entradas y salidas module_5_p1.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
H_TOTAL	1716	Números de 12 bits	Número total de componentes por línea.
H_EAV	0	Números de 12 bits	Posición de inicio de código de temporización EAV.
H_SAV	272	Números de 12 bits	Posición de inicio de código de temporización SAV.

Tabla 3-27: Parámetros module_5_p1.

Este bloque es implementado usando un sistema de regiones que permite definir las señales de control. En la Figura 3-42 se muestran las regiones que se plantea para este módulo, en este se puede ver la definición de los intervalos de video inactivo y activo, de los códigos de temporización SAV y EAV, y CLR_H que indica los últimos elementos de la línea. Este sistema es implementado por dos contadores y una memoria ROM.

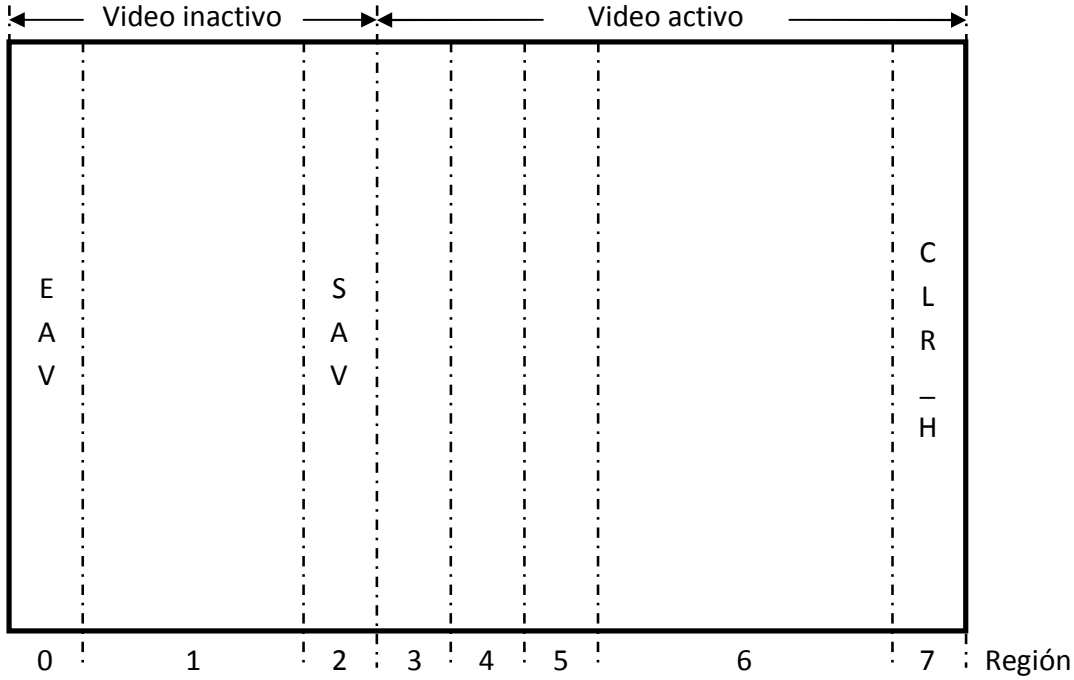


Figura 3-41: Regiones utilizadas por module_5_p1.

Los contadores son usados para almacenar la posición del componente y la región a la que pertenece el componente. La memoria ROM recibe como dirección la región en que se está trabajando y almacena en sus datos, la información del tipo componentes de la región (activo o inactivo), si la región corresponde a SAV, EAV y CLR_H, y el valor que debe tener la posición del componente para cambiar a la siguiente región. En la Tabla 3-28 se muestran los datos contenidos por la memoria ROM. La Figura 3-42 se muestra como son conectados los contadores y la memoria ROM para obtener el comportamiento deceado.

Dirección	Datos				
h_region	next_event	clr_h	eav	sav	av
000	(H_EAV/4)	0	1	0	0
001	(H_SAV/4)-1	0	0	0	0
010	(H_SAV/4)	0	0	1	0
011	(H_SAV/4)+1	0	0	0	1
100	(H_SAV/4)+2	0	0	0	1
101	(H_SAV/4)+3	0	0	0	1
110	(H_TOTAL/4)-2	0	0	0	1
111	(H_TOTAL/4)-1	1	0	0	1

Tabla 3-28: Datos almacenados en la memoria ROM usada en module_5_p1.

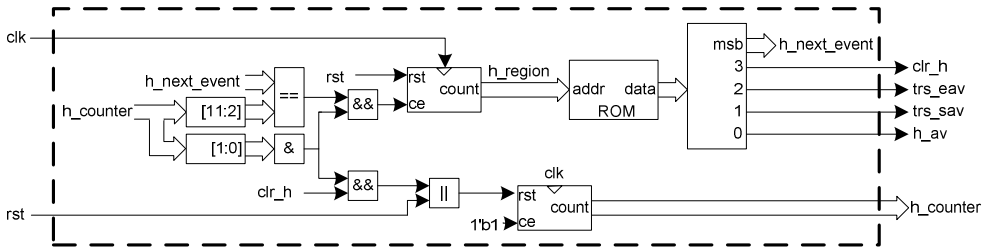


Figura 3-42: Esquema module_5_p1.

3.4.2.5.2 Módulo module_5_p2

Este dispositivo se encarga de generar la posición vertical de los componentes usados para las peticiones de transferencia y la construcción de salida de video. Además indica si la posición vertical de estos componentes corresponde a una línea activa. En Tabla 3-29 se muestran las entradas y salidas de este módulo y en la Tabla 3-30 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
h_counter	Entrada	2	Bits menos significativos de posición horizontal.
clr_h	Entrada	-	Señal de reinicio de línea.
v_counter_output	Salida	10	Posición vertical de puntero para la generación de salida de video.
v_counter_mem	Salida	10	Posición vertical de puntero para la generación de peticiones.
v_av_mem	Salida	-	Indica si la posición vertical del puntero para la generación de peticiones, corresponde a una línea activa.
v_av_output	Salida	-	Indica si la posición vertical del puntero para la generación de salida de video, corresponde a una línea activa.

Tabla 3-29: Entradas y salidas module_5_p2.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
INTERLACED	1	0,1	Indica el tipo escaneo, siendo 0 progresivo y 1 entrelazado
V_TOTAL	525	Números de 10 bits.	Número total de líneas del cuadro
V_O_AVS	20	Números de 10 bits.	Posición de inicio de líneas activas del campo impar.
V_O_BVS	264	Números de 10 bits.	Posición de término de líneas activas del campo impar.
V_E_AVS	283	Números de 10 bits.	Posición de inicio de líneas activas del campo par.
V_E_BVS	0	Números de 10 bits.	Posición de término de líneas activas del campo par.

Tabla 3-30: Parámetros module_5_p2.

Este bloque es implementado usando un sistema de regiones similar al usado en module_5_p1. En la Figura 3-42 se muestran las regiones que se plantea para este módulo, como se puede apreciar la estructura usada corresponde a un imagen entrelazada, pero esta estructura se aprovecha también para video progresivo al usarse solo las regiones desde la 0 a la 3. Esta

estructura de regiones permite generar tres señales que indican actividad de la línea, la última línea del campo par y la última línea del campo impar.

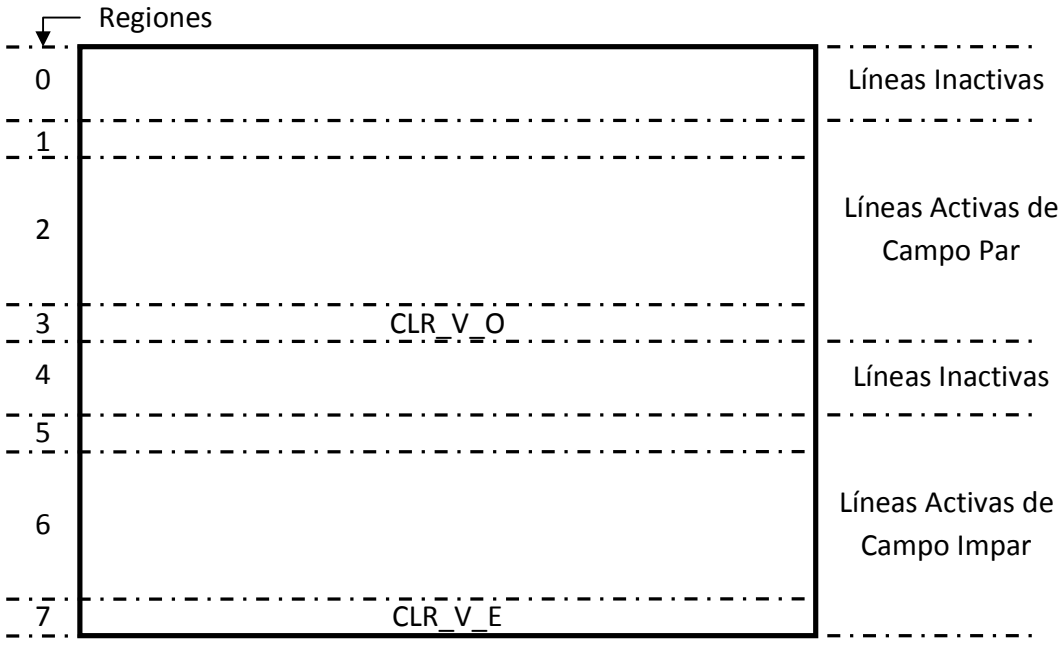


Figura 3-43: Regiones usadas por module_5_p2.

La implementación de este sistema utiliza para cada puntero dos contadores encargados de tener la posición vertical de la línea y la región en que se encuentra, y una memoria ROM que almacena en sus datos, la información del tipo de la región (activo o inactivo), si la región corresponde a CLR_V_O y CLR_V_E, y el valor que debe tener la posición de la línea para cambiar a la siguiente región. Estos datos se muestran en la Tabla 3-31, con la identificación a que señal pertenecen.

Dirección	Datos			
v_region	next_event	clr_vo	clr_ve	v_av
000	V_O_AVS-1	0	0	0
001	V_O_AVS	0	0	1
010	V_O_BVS-2	0	0	1
011	V_O_BVS-1	1	0	1
100	V_E_AVS-1	0	0	0
101	V_E_AVS	0	0	1
110	V_TOTAL-2	0	0	1
111	V_TOTAL-1	0	1	1

Tabla 3-31: Datos almacenados en la memoria ROM usada en module_5_p2.

En la Figura 3-44, se puede ver la implementación final de este módulo, en esta se puede apreciar un sistema de reinicio de los contadores dependiente del parámetro INTERLACED, para diferenciar el funcionamiento cuando se desea generar un salida de video entrelazada y una progresiva. Además se puede observar el valor de reinicio del contador v_counter_mem, puede tener dos valores distintos, esto ocurre por la necesidad de diferenciar cuando el reinicio ocurre por la señal rst o por el termino del cuadro. En la primera situación el contador debe ser llevado al valor 3 para generar una diferencia de dos líneas entre el puntero usado para la petición de transferencias y el usado para la generación de la salida de video. En la segunda situación el puntero debe ser llevado a la primera línea del cuadro.

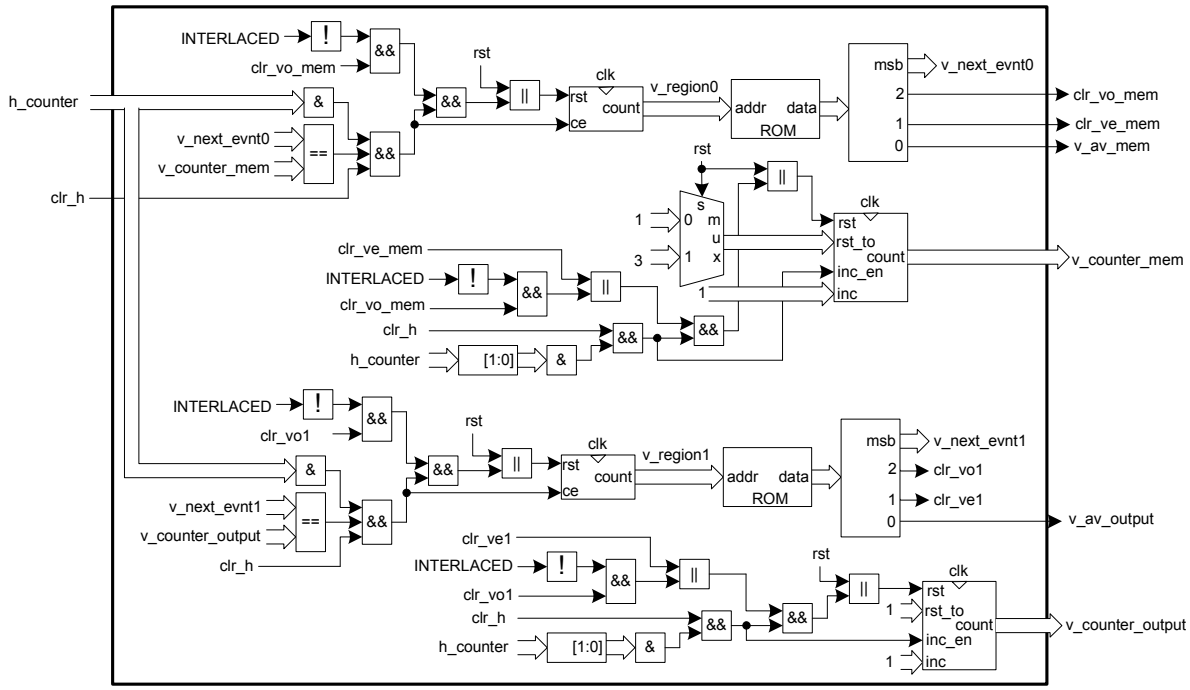


Figura 3-44: Esquema module_5_p2.

3.4.2.5.3 Módulo module_5_p3

Este módulo se encarga de determinar si el componente de una posición dada corresponde a un elemento de una imagen.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
enable	Entrada	-	Habilita el funcionamiento del modulo
h_counter	Entrada	12	Posición horizontal del componente.
clr_h	Entrada	-	Señal de reinicio de línea.
v_counter	Entrada	10	Posición vertical del componente.
h_init	Entrada	12	Posición horizontal de inicio de imagen.
v_o_init	Entrada	10	Posición vertical de inicio de imagen en el campo impar.
v_e_init	Entrada	10	Posición vertical de inicio de imagen en el campo par.
h_end	Entrada	12	Posición horizontal de término de imagen.
v_o_end	Entrada	10	Posición vertical de término de imagen en el campo impar.
v_e_end	Entrada	10	Posición vertical de término de imagen en el campo par.
is	Salida	-	Indica si el componente de la posición especificada pertenece a la imagen.

Tabla 3-32: Entradas y salidas module_5_p3.

Este módulo utiliza tres flipflops para la generación de la señal is. El primero de ellos llamado h_is es el encargado de determinar si la posición horizontal del componente pertenece al intervalo de posiciones entres h_init y h_end, los otros dos llamados v_is_o y v_is_e se preocupan de revisar si la posición vertical del componente pertenece al intervalo generado por v_o_init y v_o_end, y v_e_init y v_e_end respectivamente. Finalmente la señal is sera 1 si v_is_o o v_is_e es 1, y h_is es 1. En la Figura 3-45 se muestra el esquema de como es diseñado este bloque.

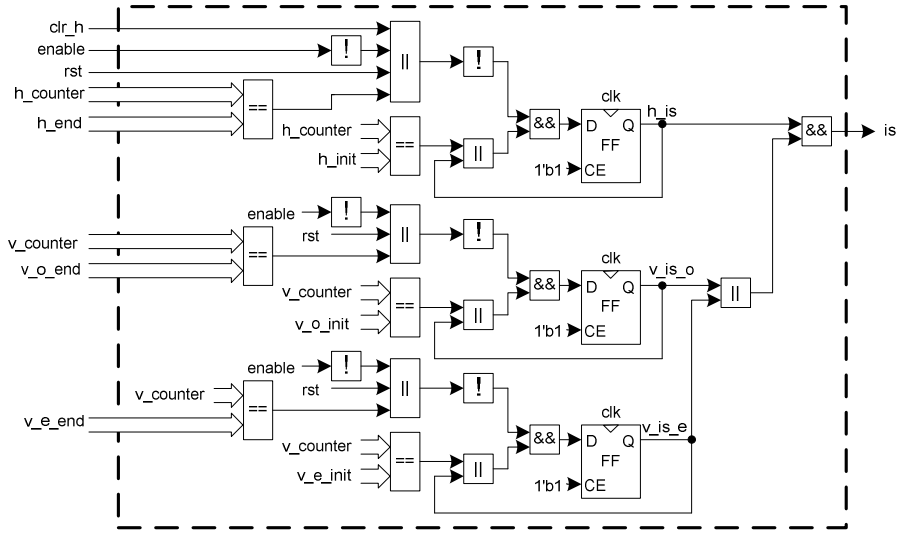


Figura 3-45: Esquema module_5_p3.

3.4.2.5.4 Módulo module_5_p4

Este módulo implementa el sistema de prioridades para la elección de la imagen mostrada en la salida de video cuando existe superposición de imágenes. En la Tabla 3-33 se muestran las entras y salidas de este bloque.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
is_AC	Entrada	-	Señal que indica si el componente es activo.
is_P0	Entrada	-	Señal que indica si el componente pertenece a la imagen 0.
is_P1	Entrada	-	Señal que indica si el componente pertenece a la imagen 1.
prio_P0	Entrada	-	Prioridad de la imagen 0.
prio_P1	Entrada	-	Prioridad de la imagen 1.
is_P0_delay	Salida	-	Retraso de señal is_P0.
is_P1_delay	Salida	-	Retraso de señal is_P1.
is_P0_pf	Salida	-	Indica si el componente perteneciente a la imagen 0 es mostrado en la salida de video.
is_P1_pf	Salida	-	Indica si el componente perteneciente a la imagen 1 es mostrado en la salida de video.

Tabla 3-33: Entradas y salidas module_5_p4.

En la implementación de la lógica de prioridad es realizada a través de una memoria ROM que funciona como decodificador, en la Tabla 3-34 se muestra el contenido de la memoria ROM usada y la correspondiente codificación usada para la generación de las señales is_P0_pf e is_P1_pf.

Dirección				Dato	
is_AC && is_P0	prio_P0	Is_AC && is_P1	Prio_P1	Is_P0_pf	Is_P1_pf
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	0	1
1	1	1	1	1	0

Tabla 3-34: Contenido de memoria ROM usada en module_5_p4.

En la Figura 3-46, se muestra la implementación completa de este módulo. En esta se puede apreciar la existencia de un flipflop después de la memoria ROM, que es utilizado para cumplir la restricción de frecuencia impuesta a todos los módulos. Este flipflop tiene por consecuencia el retraso en un ciclo de reloj de la información en comparación los datos de entrada, es por esta razón que se añaden retrasos a las entradas is_P0 y is_P1, para que la información mostrada por las señales de salida de este módulo correspondan a un mismo componente.

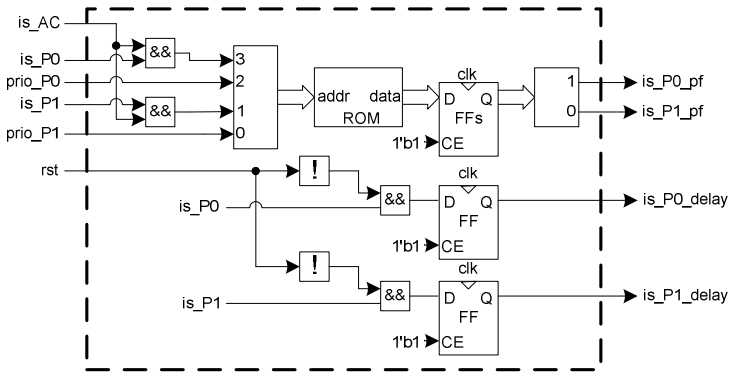


Figura 3-46: Esquema module_5_p4.

3.4.2.5.5 Módulo module_5_p5

Este modulo se encarga de realizar el conteo de los elementos mostrados y no mostrados de cada imagen. Además indica la ocurrencia del inicio de un campo, el término de este y la finalización de las líneas de cada imagen.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk_video	Entrada	-	Señal de reloj.
clk_mem	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
Interfaz con module_5_p4			
is_al_delay	Entrada	-	Señal que indica si la línea a la que pertenece el componente es activa.
is_P0_delay	Entrada	-	Señal que indica si el componente pertenece a la imagen 0.
is_P1_delay	Entrada	-	Señal que indica si el componente pertenece a la imagen 1.
is_P0_pf	Entrada	-	Señal que indica si el componente perteneciente a la imagen 0 es mostrado en la salida de video.
is_P1_pf	Entrada	-	Señal que indica si el componente perteneciente a la imagen 1 es mostrado en la salida de video.

Tabla 3-35: Entradas y salidas module_5_p5.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
Interfaz con module_5_p6			
P0_AC_count	Salida	10	Conteo de los elementos mostrados de una línea de la imagen 0.
P0_IC_count	Salida	10	Conteo de los elementos no mostrados de una línea de la imagen 0.
P1_AC_count	Salida	10	Conteo de los elementos mostrados de una línea de la imagen 1.
P1_IC_count	Salida	10	Conteo de los elementos no mostrados de una línea de la imagen 1.
send_rst_field	Salida	-	Indica inicio del campo o cuadro.
send_end_field	Salida	-	Indica término del campo o cuadro.
send_P0_rst_line	Salida	-	Indica el término de la línea de la imagen 0.
send_P0_AC_count	Salida	-	Indica el término del conteo de los elementos mostrados de una línea de la imagen 0.
send_P0_IC_count	Salida	-	Indica el término del conteo de los elementos no mostrados de una línea de la imagen 0.
send_P1_rst_line	Salida	-	Indica el término de la línea de la imagen 1.
send_P1_AC_count	Salida	-	Indica el término del conteo de los elementos mostrados de una línea de la imagen 1.
send_P1_IC_count	Salida	-	Indica el término del conteo de los elementos no mostrados de una línea de la imagen 1.

Tabla 3-35: Entradas y salidas module_5_p5 (continuación).

Para indicar el inicio o término de un campo o del cuadro se utilizan los cambios de estado de la señal is_al_delay, siendo su activación el inicio de un campo (o cuadro) y su desactivación el término de este. El término de la línea de una imagen es obtenido a partir de las señales is_P0_delay y is_P1_delay cuando estas pasan alto a bajo. El conteo de los elementos mostrados y no mostrados es realizado por dos contadores, cada uno encargado de un tipo de elemento. El comportamiento de estos contadores es controlado por su señal de reinicio, estando esta en 1 cuando no existe la condición de conteo. Tomando las señales para la imagen0, la condición de conteo de elementos mostrados ocurre cuando is_P0_delay e is_P0_pf son 1 y la de los elementos no mostrados cuando is_P0_delay es 1 estando is_P0_pf en 0. Un tema a considerar es que el reloj de los contadores es controlado por clk_video y las salidas de este módulo deben estar sincronizadas con clk_mem, esta razón lleva a colocar bancos de flipflops después de los contadores. Otro punto que debe ser tomado en cuenta es el hecho de que las peticiones de transferencia son realizadas con palabras de 32 bits, mientras que los contadores obtienen el número de elementos de 8 bits, para solucionar esto se añade un divisor por 4 entre el contador y los bancos de flipflops. Por último se debe tener presente que las señales generadas por los contadores, no permiten saber el número de elementos por sí solo, por lo que se utilizan señales auxiliares para indicar la validez de las señales generadas por los contadores. Así estas señales auxiliares corresponden a la desactivación de las condiciones de los contadores. En la Figura 3-47 se muestra la implementación de lo explicado.

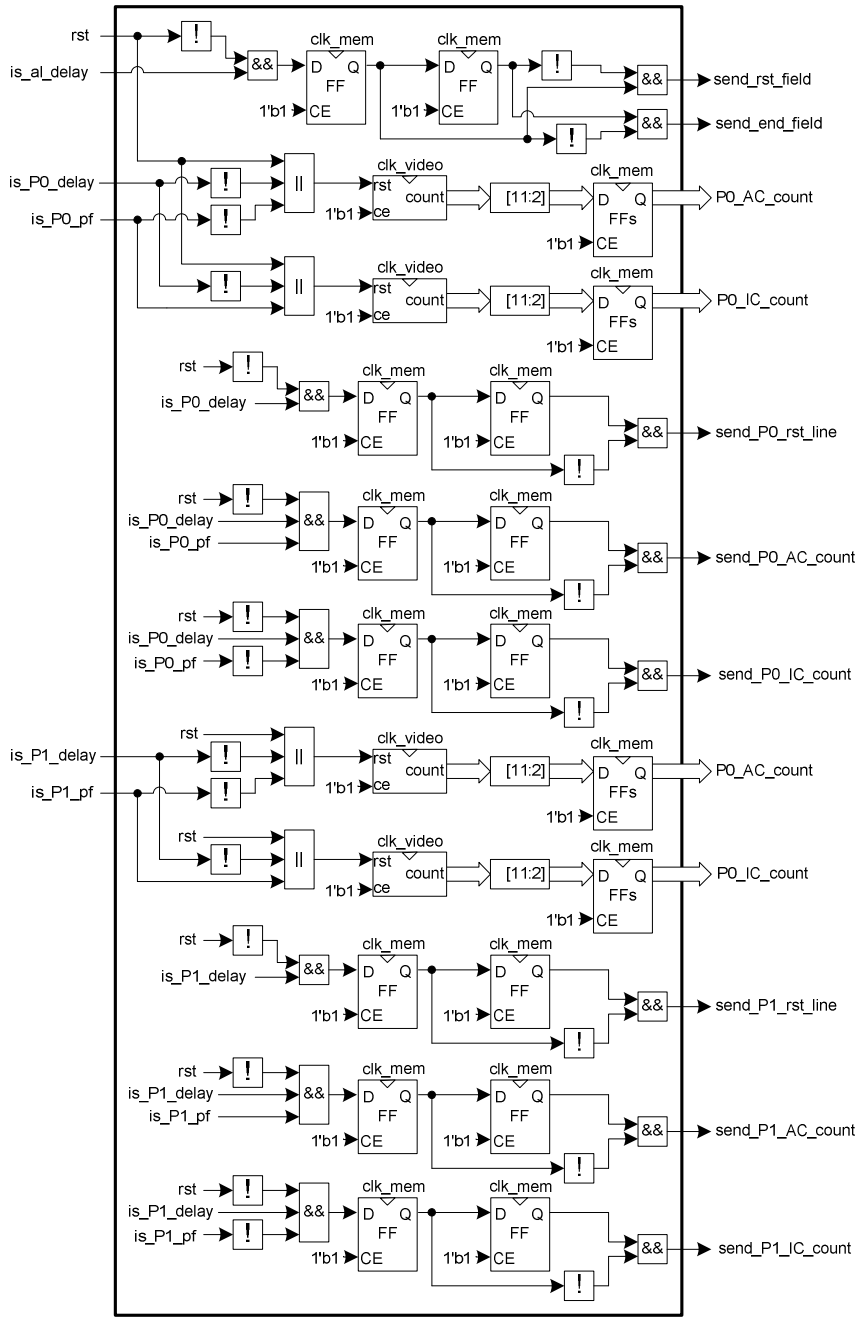


Figura 3-47: Esquema module_5_p5.

3.4.2.5.6 Módulo module_5_p6

Este bloque es el encargado de generar la interfaz de comando que permite la conexión con el dispositivo module_6. En la Tabla 3-36 se muestran las señales de entrada y salida de este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio de módulo.
Interfaz con module_5_p5			
P0_AC_count	Entrada	10	Conteo de los elementos mostrados de una línea de la imagen 0.
P0_IC_count	Entrada	10	Conteo de los elementos no mostrados de una línea de la imagen 0.
P1_AC_count	Entrada	10	Conteo de los elementos mostrados de una línea de la imagen 1.
P1_IC_count	Entrada	10	Conteo de los elementos no mostrados de una línea de la imagen 1.
send_rst_field	Entrada	-	Indica inicio del campo o cuadro.
send_end_field	Entrada	-	Indica término del campo o cuadro.
send_P0_rst_line	Entrada	-	Indica el término de la línea de la imagen 0.
send_P0_AC_count	Entrada	-	Indica el término del conteo de los elementos mostrados de una línea de la imagen 0.
send_P0_IC_count	Entrada	-	Indica el término del conteo de los elementos no mostrados de una línea de la imagen 0.
send_P1_rst_line	Entrada	-	Indica el término de la línea de la imagen 1.
send_P1_AC_count	Entrada	-	Indica el término del conteo de los elementos mostrados de una línea de la imagen 1.
send_P1_IC_count	Entrada	-	Indica el término del conteo de los elementos no mostrados de una línea de la imagen 1.
Interfaz con module_6			
send	Salida	-	Interfaz de comando. Ver Tabla 3-25.
rst_f	Salida	-	Interfaz de comando. Ver Tabla 3-25.
port	Salida	-	Interfaz de comando. Ver Tabla 3-25.
rst_l	Salida	-	Interfaz de comando. Ver Tabla 3-25.
active	Salida	-	Interfaz de comando. Ver Tabla 3-25.
count	Salida	10	Interfaz de comando. Ver Tabla 3-25.

Tabla 3-36: Entradas y salidas module_5_p6.

La construcción de una interfaz de comando tiene por función codificar las señales de entrada de este módulo para obtener una menor cantidad de señales de salida.

La existencia de sucesos como el conteo de elementos mostrados de la imagen0 y el conteo de los elementos no mostrados de la imagen1 que pueden ocurrir en un mismo instante de tiempo lleva a una dificultad mayor la codificación de las señales. Para solucionar esto se identifican los sucesos que pueden tener lugar al mismo tiempo y estos son retrasados de manera conveniente así evitando que ocurran en el mismo instante. La implementación de esto se realiza retrasando todas las señales relacionadas a la imagen1 con respecto a la misma señal para la imagen0. Además se retrasan en dos ciclos los sucesos de reinicio de línea con respecto a los sucesos de conteo.

En la Figura 3-48 se muestra la implementación final de este módulo. En la parte superior del esquema se muestra el proceso que tienen las señales send_P0_AC_count y send_P0_IC_count, estas, al no existir la posibilidad de que se activen al mismo tiempo, son unidas para formar la señal send_P0_count, que indica el suceso de conteo de elementos en la imagen0. El mismo trato se tiene con las señales send_P1_AC__count y send_P1_IC__count, pero la unión de esa señal es retrasada en ciclo de reloj con respecto a send_P0_count. Un poco más abajo de esta sección se muestra la generación de la señal active, que corresponde a los retrasos correspondientes de las señales send_P0_AC_count y send_P1_AC_count.

Las señales P0_AC_count y P0_IC_count son unidas a través de un multiplexor que selecciona la señal P0_AC_count cuando se trata de un conteo de elementos mostrados. La unión de estas señales es nuevamente multiplexada con una señal cero con el fin de tener el conteo solo cuando este está habilitado por alguna de las señales send_P0_AC_count o send_P0_IC_count. El mismo tratamiento se les da a las señales de conteo de la imagen1, para ser luego retrasadas y posteriormente unidas a través de la lógica OR bit a bit a la unión de los conteos de la imagen0, así generando la señal de salida count.

La señal rst_l corresponde a la unión de los retrasos correspondientes de las señales send_P0_rst_line y send_P1_rst_line. El retraso de esta última señal además es usado en conjunto con send_P1_count_delay, permiten activar la señal port para identificar que el término de línea o el conteo corresponde a la imagen1.

Finalmente la señal send es generada a partir de las señales send_rst_field, send_P0_count, send_P1_count_delay, send_P0_rst_line_delay[2] y send_P1_rst_line_delay[3], las que de existir una que se active, activara a la señal send.

3.4.2.5.7 Módulo module_5_p7

Este módulo es el encargado obtener los datos almacenados en RdFIFO, almacenarlos y transformarlos a elementos de 8 bites. En la Tabla 3-37 se muestran las entradas y salidas de este bloque.

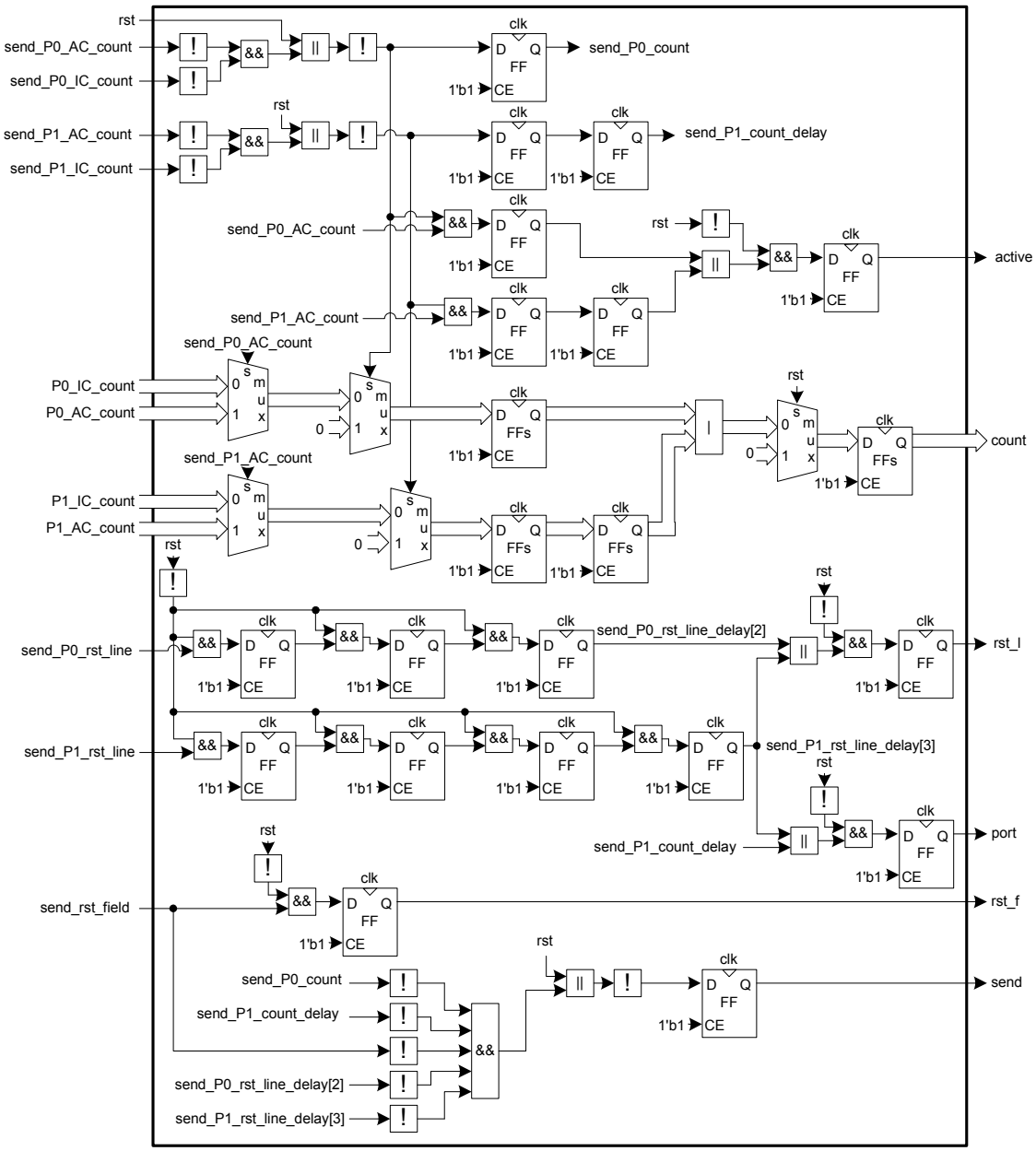


Figura 3-48: Esquema module_5_p6.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk_mem	Entrada	-	Señal de reloj.
clk_video	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
RdFIFO_Empty	Entrada	-	Ver Tabla 2-17.
RdFIFO_Latency	Entrada	2	Ver Tabla 2-17.
RdFIFO_Data	Entrada	32	Ver Tabla 2-17.
RdFIFO_Pop	Salida	-	Ver Tabla 2-17.
RdFIFO_Flush	Salida	-	Ver Tabla 2-17.
read_fifo	Entrada	-	Señal de lectura de salida de dato.
data_out	Salida	8	Dato de salida.

Tabla 3-37: Entradas y salidas module_5_p7.

La obtención de elementos desde RdFIFO es realizada siempre que esta cola tenga elementos disponibles y la cola de almacenamiento de este dispositivo no esté llena. En la Figura 3-49 se muestra la implementación de este módulo. La señal RdFIFO_Pop incorpora el comportamiento explicado al principio del párrafo. Debido a la existencia de latencia en la entrega de datos por parte de del módulo MPMC, se genera la señal write_fifo a partir del retraso de la señal RdFIFO_Pop en una cantidad de veces correspondiente a RdFIFO_Latency. Finalmente se utiliza el bloque module_5_fifo para el almacenamiento y transformación de los datos, la escritura de datos en este dispositivo es controlada por la señal write_fifo y la lectura por read_fifo.

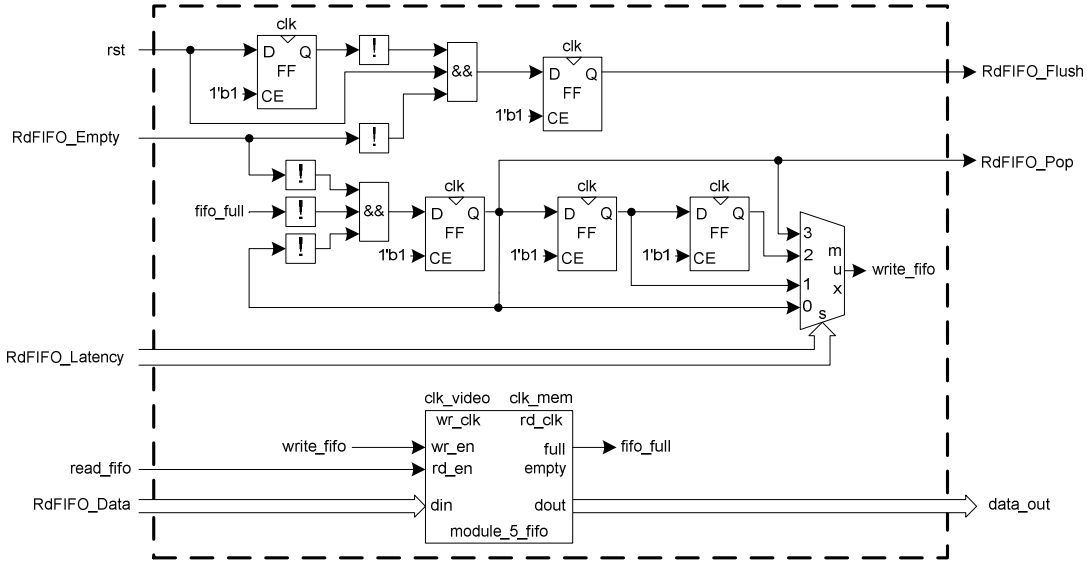


Figura 3-49: Esquema module_5_p7.

3.4.2.5.7.1 Módulo module_5_fifo

Esta cola es la encargada del almacenamiento de datos y su posterior transformación. En la Tabla 3-38 se muestran las entradas y salidas de este módulo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
rd_clk	Entrada	32	Señal de reloj de lectura.
wr_clk	Entrada	-	Señal de reloj de escritura.
rst	Entrada	-	Señal de reinicio.
din	Entrada	-	Datos de entrada.
wr_en	Entrada	-	Habilita la escritura en la cola.
rd_en	Entrada	-	Habilita la lectura de la cola.
dout	Salida	8	Datos de salida.
empty	Salida	-	Indica que no existen elementos para leer.
full	Salida	-	Indica que no se pueden escribir mas elementos.

Tabla 3-38: Entradas y salidas module_5_fifo.

La utilización de este módulo como memoria de almacenamiento lleva a utilizar un bloque de RAM para la implementación de este módulo, utilizando uno de sus puertos para la escritura de datos y el otro la lectura. El resto de la implementación de este módulo es similar a la de otras colas, usando contadores para la generación de las direcciones de escritura y lectura, que además son usadas para la determinación de las señales empty y full. Como incorporación a este modulo se le añade un contador que indica la dirección de escritura más tres, el que es usado para la construcción de la señal full, haciendo que esta se active cuando faltan dos elementos para llenar la memoria.

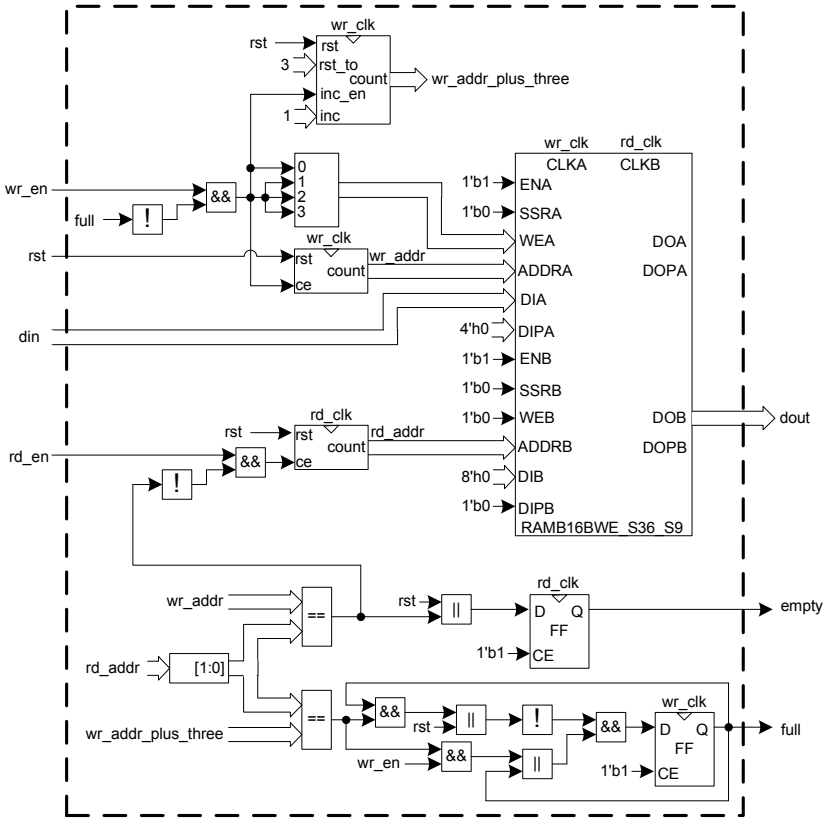


Figura 3-50: Esquema module_5_fifo.

3.4.2.5.8 Módulo module_5_p8

Este módulo es el encargado de determinar a qué campo pertenece el componente de una posición dada. En la Tabla 3-39 se muestran las señales de entrada y salida de este dispositivo.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj
rst	Entrada	-	Señal de reinicio.
h_counter	Entrada	2	Bits menos significativos de posición horizontal de componente.
clr_h	Entrada	-	Señal de reinicio de línea.
v_counter	Entrada	10	Posición vertical de componente.
start_odd_field	Entrada	10	Posición de inicio del campo impar.
start_even_field	Entrada	10	Posición de inicio de campo par.
field	Salida	-	Indica el campo al que pertenece el componente.

Tabla 3-39: Entradas y salidas module_5_p8.

La implementación de este módulo es realizada por un flipflop que cuando el último componente de una línea tiene por posición vertical el inicio del campo par menos uno se activa y cuando el último componente de una línea tiene por posición vertical el inicio del campo impar se desactiva. En la Figura 3-51 se muestra la implementación de este módulo.

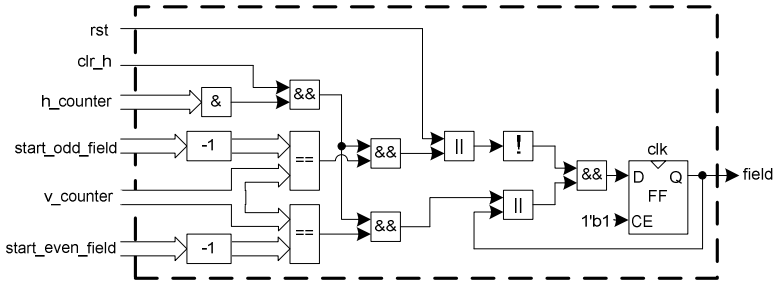


Figura 3-51: Esquema module_5_p8.

3.4.2.5.9 Módulo module_5_p9

Este módulo es el encargado de generar la salida de video. En la Tabla 3-40 se muestran las entradas y salidas de este dispositivo y en la Tabla 3-41 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
h_counter	Entrada	2	Indica si el componente es luma, croma azul o croma rojo.
trs_sav	Entrada	-	Indica si el componente pertenece al código de temporización SAV.
trs_eav	Entrada	-	Indica si el componente pertenece al código de temporización EAV.
field	Entrada	-	Indica el campo al que pertenece el componente.
h_av	Entrada	-	Indica si la posición horizontal corresponde a un componente activo.
v_av	Entrada	-	Indica si el componente pertenece a una línea activa.
is_from_fifo	Entrada	-	Indica si el componente proviene desde la cola de module_5_p7.
fifo_data	Entrada	8	Valor de componente proveniente de cola.
output_video	Salida	8	Salida de video.

Tabla 3-40: Entradas y salidas module_5_p9.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
BLACK_Y	16		Valor de componentes luma en pixeles sin imagen.
BLACK_CB	128		Valor de componentes croma azul en pixeles activos no pertenecientes a imágenes.
BLACK_CR	128		Valor de componentes croma rojo en pixeles activos no pertenecientes a imágenes.

Tabla 3-41: Parámetros module_5_p9.

Este módulo parte generando una señal con solo elementos del color indicado por los parámetros BLACK_Y, BLACK_CB y BLACK_CR, los cuales son multiplexados adecuadamente a través de la señal h_counter. Posteriormente esta señal es usada en conjunto con los datos provenientes de module_5_p7 para crear una nueva señal que contiene los elementos mostrados de imagen0 e imagen1 y en caso de no ser este tipo de elementos corresponden a componentes de color. En paralelo a lo anterior se crea una señal que contiene los elementos de supresión de trama y los códigos de temporización. Finalmente usando la característica del componente de ser activo o inactivo se selecciona entre las dos señales previamente creadas, obteniéndose así la salida de video. En la Figura 3-52 se muestra la implementación de este bloque.

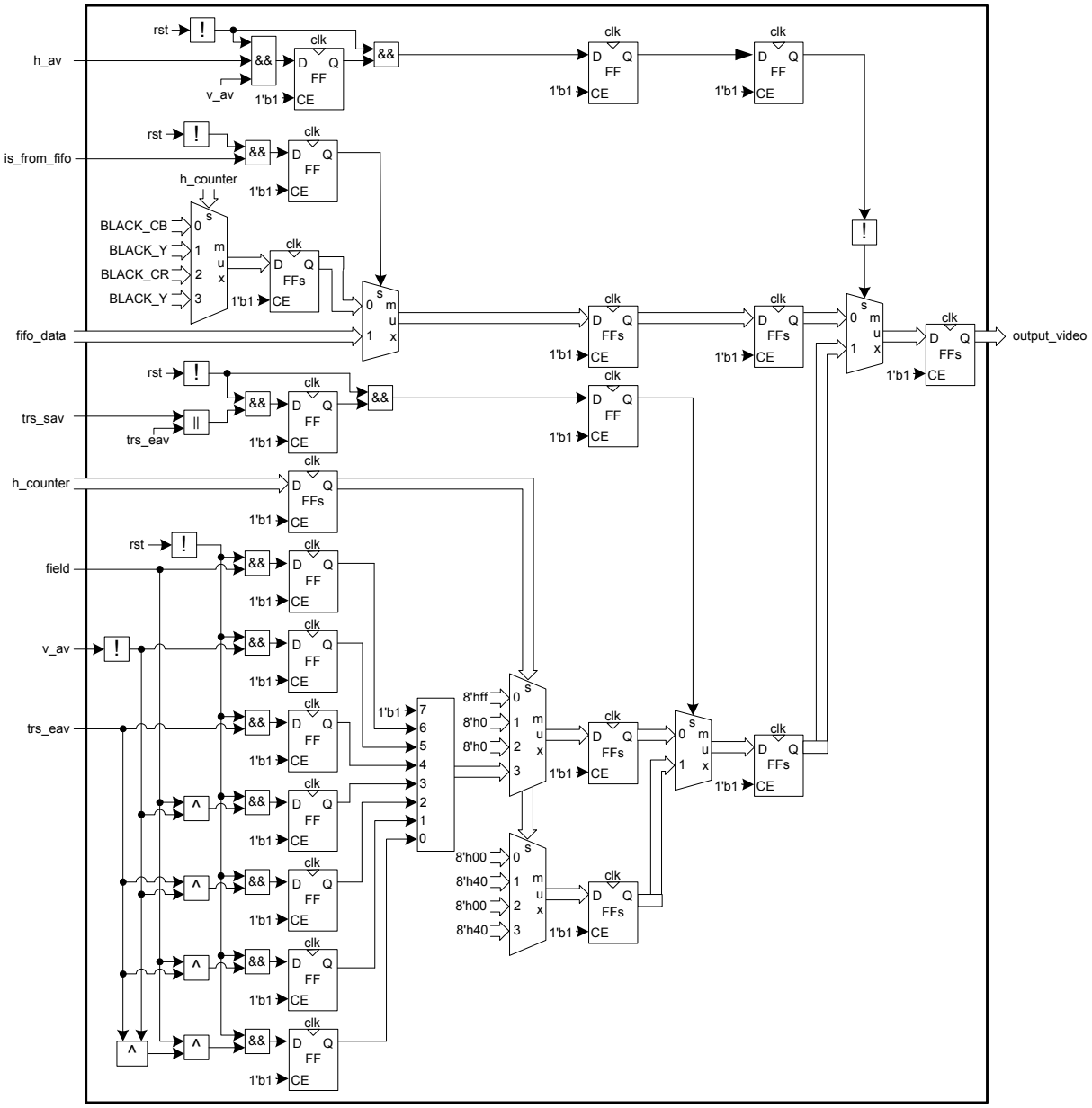


Figura 3-52: Esquema module_5_p9.

3.4.2.6 Módulo module_6

Este bloque es el encargado de realizar la peticiones de transferencia al modulo MPMC, para la obtención de datos por parte de module_5. En la Tabla 3-42 se muestran las entradas y salidas de este dispositivo y en la Tabla 3-43 sus parámetros.

Nombre de Señal	Dirección	Tamaño de Bus (Bits)	Descripción
clk	Entrada	-	Señal de reloj.
rst	Entrada	-	Señal de reinicio.
Interfaz NPI con módulo MPMC			
AddrAck	Entrada	-	Ver Tabla 2-17.
RdFIFO_Empty	Entrada	-	Ver Tabla 2-17.
Addr	Salida	32	Ver Tabla 2-17.
AddrReq	Salida	-	Ver Tabla 2-17.
Size	Salida	4	Ver Tabla 2-17.
RNW	Salida	-	Ver Tabla 2-17.
Interfaz con module_5			
end_f	Entrada	-	Señal que indica termino de campo activo.
send	Entrada	-	Interfaz de comando. Ver Tabla 3-25.
rst_f	Entrada	-	Interfaz de comando. Ver Tabla 3-25.
port	Entrada	-	Interfaz de comando. Ver Tabla 3-25.
rst_l	Entrada	-	Interfaz de comando. Ver Tabla 3-25.
active	Entrada	-	Interfaz de comando. Ver Tabla 3-25.
count	Entrada	10	Interfaz de comando. Ver Tabla 3-25.
Interfaz con module_4			
v0_sa	Entrada	32	Dirección de inicio del buffer de video que contiene la imagen redimensionados de la entrada de video 0.
v0_acpl	Entrada	12	Número de componentes activos de una línea de la imagen redimensionada de la entrada de video 0.
v0_wc	Entrada	-	Indica la existencia de datos validos en el buffer de video que contiene la imagen redimensionados de la entrada de video 0.
v0_mw	Salida	-	Indica si el módulo está leyendo el buffer de video que contiene la imagen redimensionados de la entrada de video 0.
v1_sa	Entrada	32	Dirección de inicio del buffer de video que contiene la imagen redimensionados de la entrada de video 1.
v1_acpl	Entrada	10	Número de componentes activos de una línea de la imagen redimensionada de la entrada de video 1.
v1_wc	Entrada	-	Indica la existencia de datos validos en el buffer de video que contiene la imagen redimensionados de la entrada de video 1.
v1_mw	Salida	-	Indica si el módulo está leyendo el buffer de video que contiene la imagen redimensionados de la entrada de video 1.

Tabla 3-42: Entradas y salidas module_6.

Nombre de Parámetro	Valor por Defecto	Valores permitidos	Descripción
INTERLACED	1		Indica si la adquisición de datos se realizada de forma entrelazada.
P0_VIDEO_INPUT	"in0"	"in0","in1"	Indica que entrada de video es usada por la imagen 0.
P1_VIDEO_INPUT	"in1"	"in0",in1"	Indica que entrada de video es usada por la imagen 1.

Tabla 3-43: Parámetros module_6.

Este módulo parte eligiendo las señales provenientes de module_4 para la caracterización de la imagen0 y la imagen1, utilizando los parámetros P0_VIDEO_INPUT y P1_VIDEO_INPUT, en la Figura 3-53 se muestra la elección de la dirección de inicio del buffer de video usado por la imagen0 y el largo de línea de la imagen a través de multiplexores. Además en esta figura se agrega la definición de la P0_start_elements_left. En el caso de la imagen1 se utiliza la misma configuración pera generar sus señales.

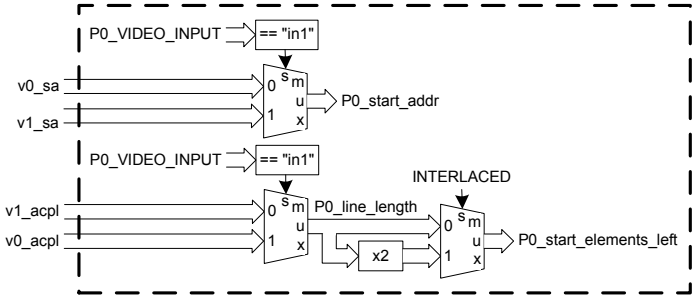


Figura 3-53: Esquema module_6, elección de características de imagen0 e imagen1.

La primera tarea realizada por el bloque es recibir los comandos enviados desde module_5 y guardarlos en una cola, con esto se evita que se pierdan comandos y que estos se realicen en el orden de llegada al modulo. En la Figura 3-54 se puede ver la implementación de esta tarea. La señal send es usada para la escritura en la cola module_6_fifo, y los datos son leídos usando la señal fifo_read que será explicada más adelante.

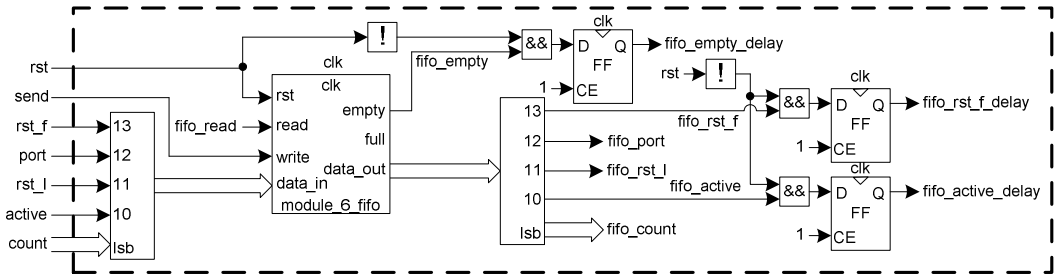


Figura 3-54: Esquema module_6, almacenamiento de comandos.

Una de las señales de control usadas por este dispositivo es pr_cmd, esta indica que se está realizando un comando almacenado en module_6_fifo. El inicio de la ejecución de un comando esta dado por la existencia de elementos en module_6_fifo y el estado inactivo de fifo_read, la duración del proceso de ejecución está determinado por el tipo de comando realizado, siendo solo el comando de petición de transferencia el que único que dura más de un ciclo de reloj, para este comando la condición de término es que realización de todas las peticiones necesarias para la transferencia de todos las palabras requeridas. pr_cmd también es usada para generar fifo_read, esta señal se encarga de cambiar la posición de lectura de la cola, por lo que tiene valor 1 cuando se desactiva la señal pr_cmd. En la Figura 3-55, se muestra la los elementos y señales usados para generación de pr_cmd y fifo_read.

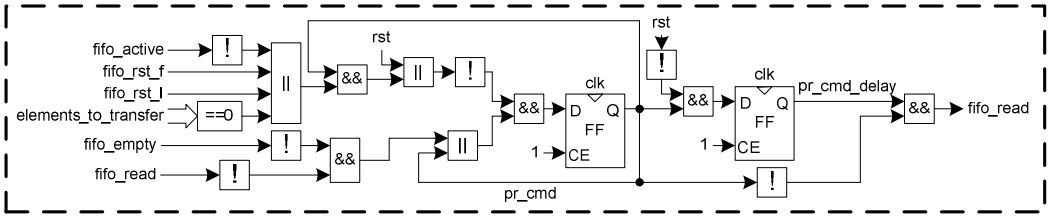


Figura 3-55: Esquema module_6, generación de señales de control pr_cmd y fifo_read.

En la Figura 3-56 se muestra como es generada la señal Addr. Como se ve en el esquema esta señal proviene de la selección, a través de fifo_port, entre las señales P0_addr o P1_addr, las cuales identifican la dirección de lectura de la próxima transferencia a realizarse, para la obtención de datos para la imagen0 o la imagen1 respectivamente. La generación de las direcciones P0_addr y P1_addr es similar en la gran mayoría de sus aspectos, diferenciándose únicamente en la utilización de fifo_port que es negada para la generación de las señales de control usadas para manejar P0_addr. Por esto las siguientes explicaciones se referirán solo a la lógica usada para obtener elementos de la imagen 0 y se considerara que la implementación de la lógica para la imagen 1 es la misma que la de la imagen 0, pero sin el negador posterior a una señal fifo_port.

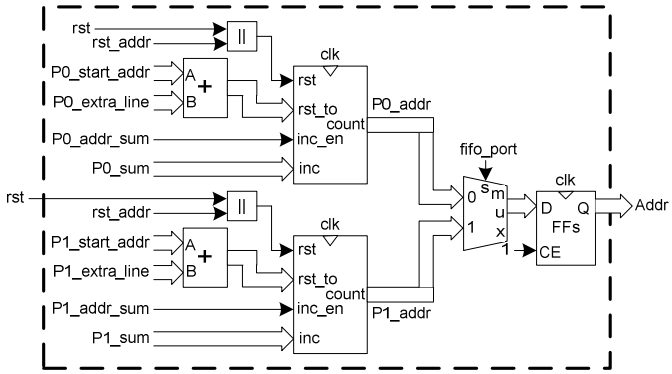


Figura 3-56: Esquema module_6, generación de Addr.

El contador de P0_addr es reiniciado por la señal rst_addr a un valor dado por la suma de P0_start_addr y P0_extra_line. Como se explico en un capitulo anterior la obtención de elementos puede iniciarse en dos puntos distintos de la imagen en caso que la adquisición de datos sea entrelazada, esta situación es manejada por el bus P0_extra_line, el cual si el parámetro INTERLACED es 1 y se está leyendo para el campo par de la imagen tendrá valor 0, en cambio si es la lectura del campo impar este tendrá un valor igual al tamaño de una línea en palabras de 32 bits. Si INTERLACED es 0, siempre tendrá valor 0. En tanto la señal rst_addr se activa cuando el comando leído es un reinicio de cuadro o campo y la pr_cmd esta activa. En la Figura 3-57 se esquematizan la logica usada para la generación de las señales rst_addr y P0_extra_line.

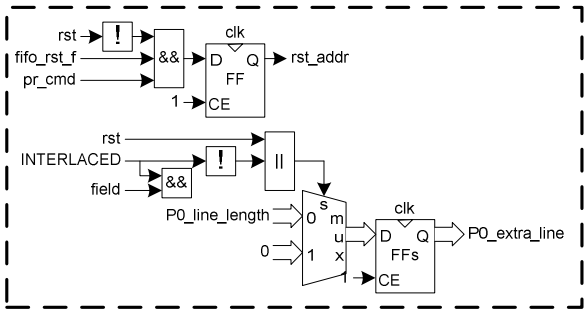


Figura 3-57: Esquema module_6, generación de rst_addr y P0_exta_line.

La condición de incremento del contador P0_addr está dada por la señal P0_addr_sum, y el valor del incremento por P0_sum. Como se explicó en un capítulo anterior el incremento de la dirección puede suceder cuando se realizan peticiones de transferencias, la existencia de elementos de la imagen no mostrados y elementos de la imagen que quedan fuera del intervalo horizontal de video activo. La primera condición ocurre cuando se recibe el AddrAck desde el módulo MPMC, en el caso de las otras dos solo depende que el comando ejecutado corresponda a incremento de la dirección o a un termino de línea. En la Figura 3-58 se muestra la implementación de esta señal y además la de P0_sum.

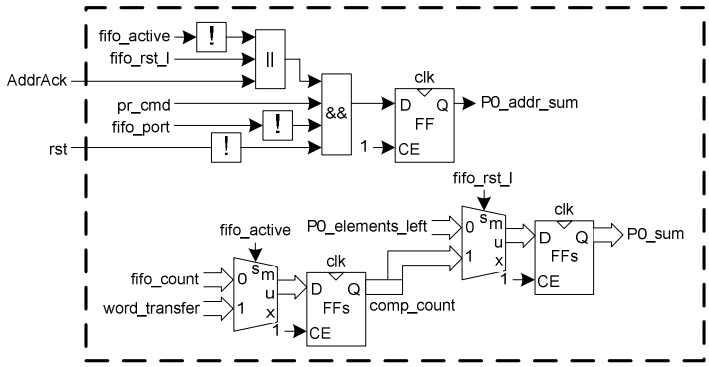


Figura 3-58: Esquema module_6, generación de señales P0_addr_sum y P0_sum.

P0_sum puede obtener su valor de tres formas distintas dependiendo del comando que se esté ejecutando, si es un incremento en la dirección este será obtenido desde fifo_count, en caso de ser una petición de transferencia su valor será igual al tamaño de la transferencia, por lo que será obtenido de word_transfer y si es un término de línea se obtendrá de P0_elements_left. Esta señal corresponde al número de elementos de la o las líneas que no han sido considerados dentro de los comandos de incremento de dirección o petición de transferencia. El número de líneas depende del parámetro INTERLACED, el cual en caso de ser 1 serán dos líneas y en caso de ser 0 será una línea. Esta consideración está dada por la señal P0_start_elements_left, que es generada como se muestra en la Figura 3-53. El valor de P0_elements_left es reiniciado a P0_start_elements_left cuando se recibe el comando de término de línea, como esta señal es utilizada para el incremento de P0_addr, su reinicio debe realizarse después del incremento de la dirección, para realizar esto correctamente se utiliza un retraso de la señal pr_cmd, en vez de la señal en sí. También se añade como condición de reinicio el comando de término de cuadro (o campo). En tanto las condiciones de reducción en el valor de P0_elements_left son el comando de incremento en la dirección o el arbitraje de una petición de transferencia. En la primera condición su valor disminuye en lo señalado por fifo_count y en la segunda por el tamaño de la petición. En la Figura 3-59 se muestra la implementación de P0_elements_left.

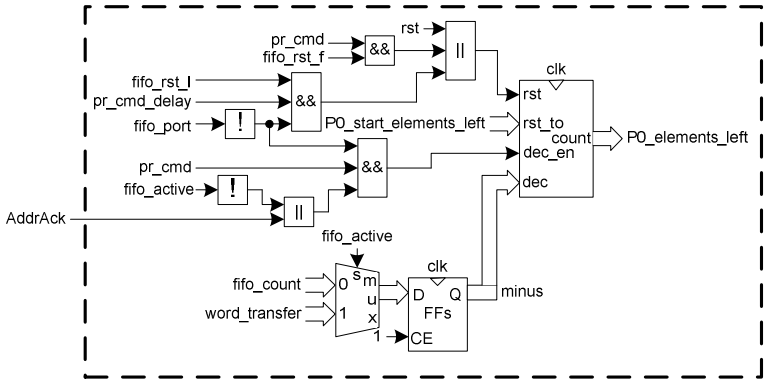


Figura 3-59: Esquema module_6, generación de P0_elements_left.

La generación de la señal Size se realiza de manera similar a los otros módulos con interfaz NPI; un contador lleva el número de elementos que faltan por transferir, con este valor se determina el máximo tamaño que puede tener la transferencia y este valor es codificado para obtener Size. La diferencia con los otros bloques, es que para este se reinicia el valor de elements_to_transfer cada vez que se inicia la ejecución de un comando petición de transferencia, al valor de fifo_count. En la Figura 3-60 se muestra un esquema de la implementación de Size y AddrReq, esta última se puede ver que es controlada por AddrAck, pr_cmd, fifo_active y otra señal que indica que contador lag_counter es distinto de cero. La configuración de los elementos lógicos se implementa para que AddrReq este activa en todo el proceso de ejecución del comando petición de transferencia, menos cuando existe AddrAck es 1 y lag_counter es distinto de cero.

La primera condición de inactividad se debe a que una vez arbitrada la petición AddrReq debe volver a 0, la segunda es para compensar la latencia que tiene el computo de las señales Size y Addr. Para esto lag_counter es reiniciado a un valor de dos cuando se inicia la ejecución del comando o cuando se arbitra una petición y este valor disminuye cada ciclo de reloj hasta llegar a cero.

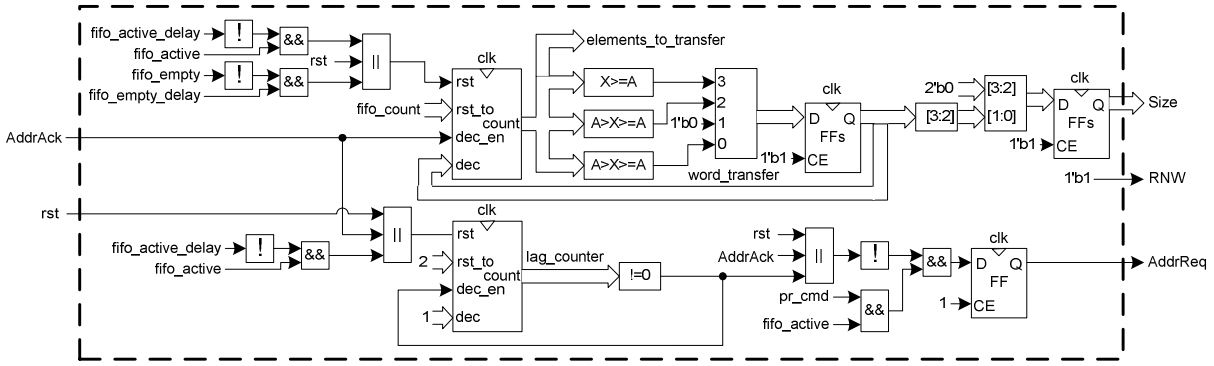


Figura 3-60: Esquema module_6 parte 8.

Las señales v0_mw y v1_mw, toman el mismo valor en este módulo cuando existen elementos en los buffers de video respectivo, esto se debe a que ambas señales son generadas por la señal mem_working. Esta señal tiene como condición de activación la lectura del comando reinicio de dirección, cuando la señal field es 0. Para generar la condición de desactivación de mem_working, se utiliza la señal still_end_f, que permite esperar que sean leídos todos los elementos de RdFIFO antes de desactivar mem_working. En la Figura 3-61 se muestra la implementación de esta señal.

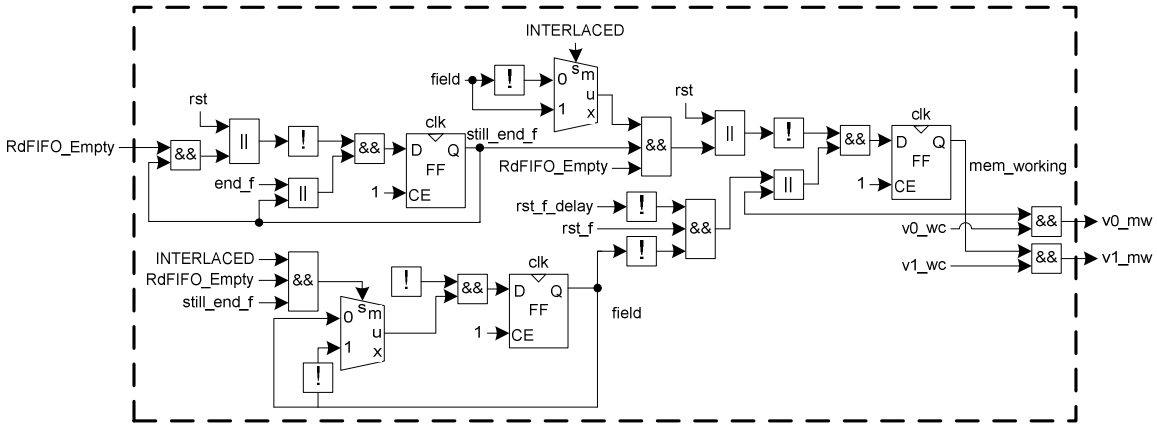


Figura 3-61: Esquema module_6 parte 9.

3.5 Simulación de Comportamiento

Esta simulación tiene la función de verificar el comportamiento teórico de los diseños en HDL, sin la consideración de tiempos de propagación.

Debido a que los tiempos de simulación en video son muy altos, solo se muestran partes relevantes del funcionamiento de cada módulo, las cuales permiten comprobar el correcto funcionamiento de cada módulo.

La simulación consiste en dos entradas de video que tienen las características enunciadas en la siguiente tabla.

Característica	Entrada de video 0	Entrada de video 1
Periodo de muestreo	30ns	30ns
Tipo de video	Entrelazado	Progresivo
Número de componentes totales por línea	104	64
Componentes activos por línea	80	40
Líneas totales por cuadro	35	16
Líneas activas por cuadro	-	8
Líneas activas de campo 0	14	-
Líneas activas de campo 1	15	-

Tabla 3-44: Características de entradas de video en simulación de comportamiento.

Los datos de las entradas de video corresponden a una imagen con muchas columnas de color, en otras palabras todas las líneas de video activo son iguales. Los datos transmitidos en una línea activa corresponden a la secuencia que parte con el valor 0h01 y crece en 2 hasta completar el número de componentes activos por línea como se muestra en la Figura 3-62.

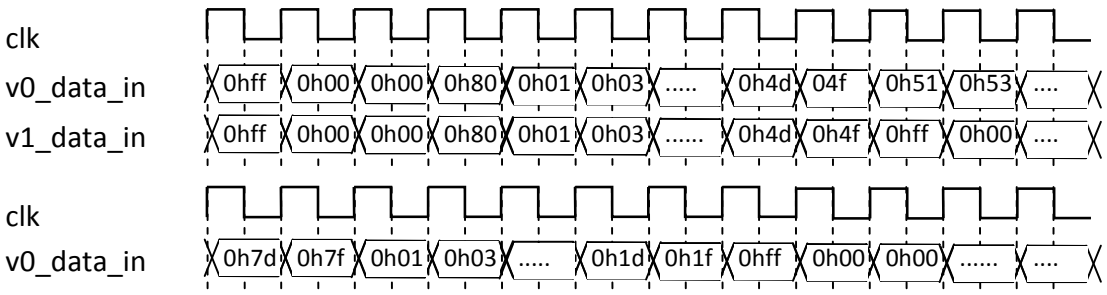


Figura 3-62: Ejemplo entradas de video para línea de video activo.

En la simulación se usa un modelo en Verilog de una memoria Micron, la cual se modificaron sus parámetros para que tenga las mismas características de la memoria usada por la plataforma StarterKit.

Finalmente el sistema se deja configurado con los parámetros de la Tabla 3-45.

Modulo	Parámetro	Valor
module_4	HORIZONTAL_MULTIPLIER_SCALING_VIDEO_0	4'h1
	HORIZONTAL_DIVISOR_SCALING_VIDEO_0	4'h 2
	VERTICAL_MULTIPLIER_SCALING_VIDEO_0	4'h 1
	VERTICAL_DIVISOR_SCALING_VIDEO_0	4'h 1
	HORIZONTAL_MULTIPLIER_SCALING_VIDEO_1	4'h 2
	HORIZONTAL_DIVISOR_SCALING_VIDEO_1	4'h 1
	VERTICAL_MULTIPLIER_SCALING_VIDEO_1	4'h 2
	VERTICAL_DIVISOR_SCALING_VIDEO_1	4'h 1
	START_ADDR_VIDEO_0_SCALING_IN_BUFFER_0	32'h00_00_00_00
	START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	32'h00_80_00_00
	START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_0	32'h01_00_00_00
	START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	32'h01_80_00_00
	START_ADDR_VIDEO_1_SCALING_IN_BUFFER_0	32'h02_00_00_00
	START_ADDR_VIDEO_1_SCALING_IN_BUFFER_1	32'h02_80_00_00
	START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_0	32'h03_00_00_00
	START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_1	32'h03_80_00_00
module_5	SCANNING	"interlaced"
	RASTER	"sim"
	PICT0	"in0"
	PICT0_TL_X	0
	PICT0_TL_Y	0
	PICT0_PRIO	0
	PICT0	"in1"
	PICT0_TL_X	0
	PICT0_TL_Y	10
	PICT0_PRIO	0

Tabla 3-45: Parámetros del sistema probado en simulación.

3.5.1 Módulo module_1

La principal preocupación en las simulaciones de este módulo fue que este detectara correctamente los códigos de temporización y los decodificara correctamente a las señales hs_n, vs_n y field. Lo segundo a verificar es la correcta determinación de las dimensiones de la imagen de video y finalmente comprobar el funcionamiento de las señales locked e is_interlaced.

Para la verificación del primer tema propuesto, se presentan imágenes de las simulaciones del módulo que procesa la entrada de video 0, las que muestran códigos de temporización para distintas partes del video. En algunas de estas imágenes se agrega la generación de algunas dimensiones de la imagen y de la señal locked.

En la siguiente tabla se muestra el valor del cuarto termino del código de temporización al que se le realiza la prueba, el valor esperado de las señales hs_n, vs_n y field, y la correspondiente figura que muestra la correcta decodificación del código.

Valor cuarta palabra de código de temporización (Hexadecimal)	Valor esperado de señal			Figura
	hs_n	vs_n	field	
0hDA	1	0	1	Figura 3-63
0hAB	0	1	0	Figura 3-64
0h9D	1	0	0	Figura 3-65
0h80	0	0	0	Figura 3-66
0hB6	1	1	0	Figura 3-67
0hF1	1	1	1	Figura 3-68
0hEC	0	1	1	Figura 3-69
0hC7	0	0	1	Figura 3-70

Tabla 3-46: Identificación de códigos de temporización en simulaciones.

El cálculo de las dimensiones de la imagen y tipo de video se realiza correctamente para las dos entradas de video, las cuales dan los valores entregados en la Tabla 3-47.

Señal	**Valor por entrada de video**	
	Entrada 0	**Entrada 1**
active_components_per_line	80	40
field_0_active_lines_per_field	14	8
field_1_active_lines_per_field	15	8
is_interlaced	1	0

Tabla 3-47: Dimensiones de entradas de video obtenidas por module_1 en simulación de comportamiento.

Finalmente en la Figura 3-65 se muestra la generación de la señal locked, la cual ocurre tal como se diseño.

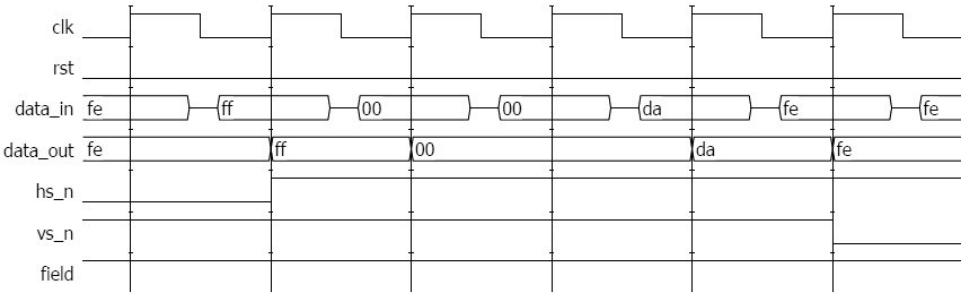


Figura 3-63: Detección de codigo 0hDA.

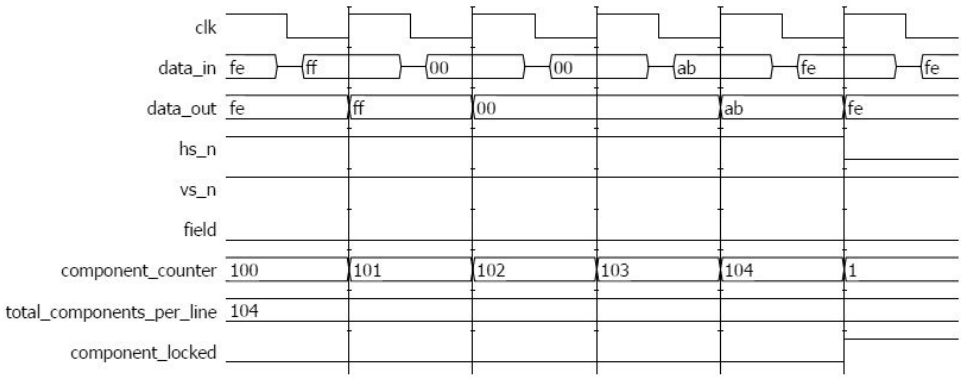


Figura 3-64: Detección de código 0hAB y generación de component_locked.

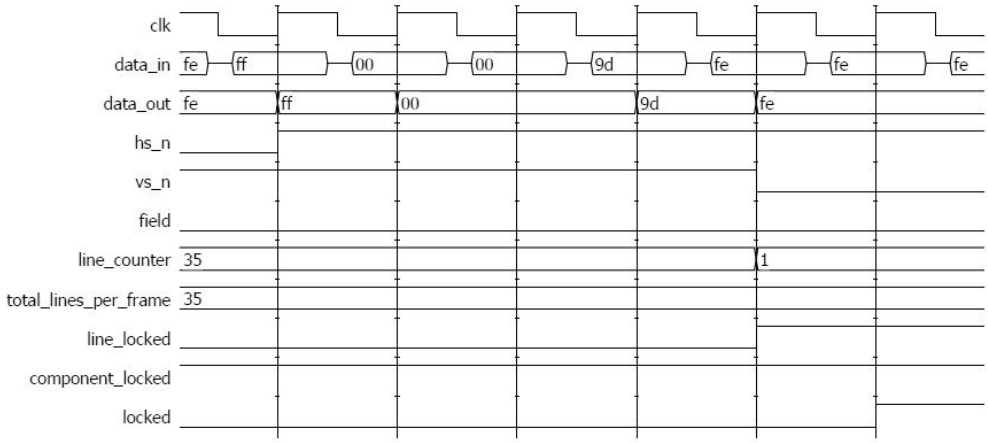


Figura 3-65: Detección de código 0h9D y generación de locked.

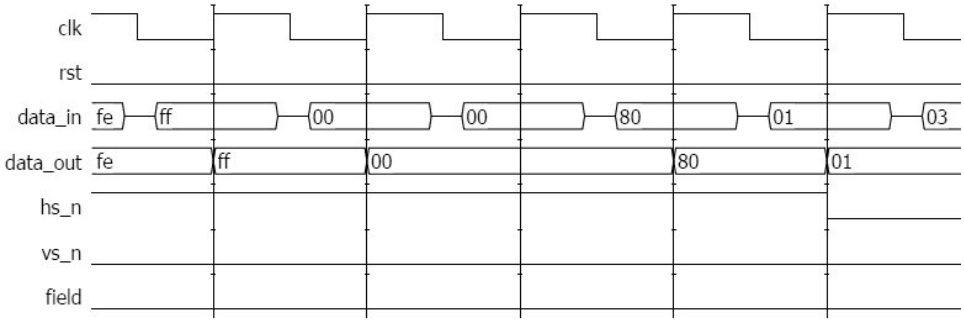


Figura 3-66: Detección de código 0h80.

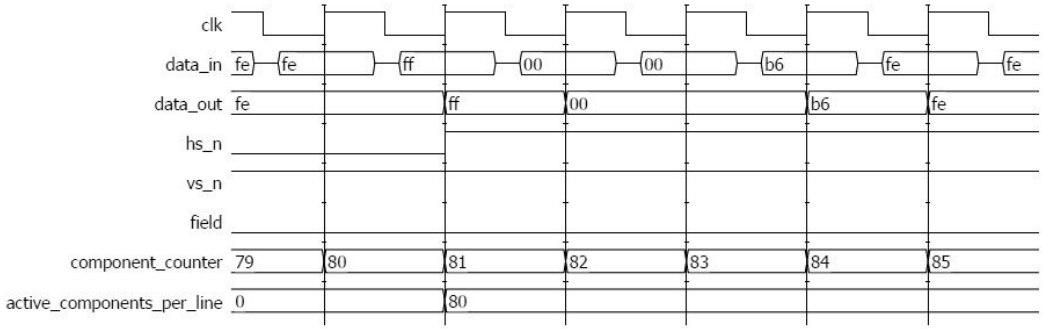


Figura 3-67: Detección de código 0hB6 y generación de señal active_components_per_line.

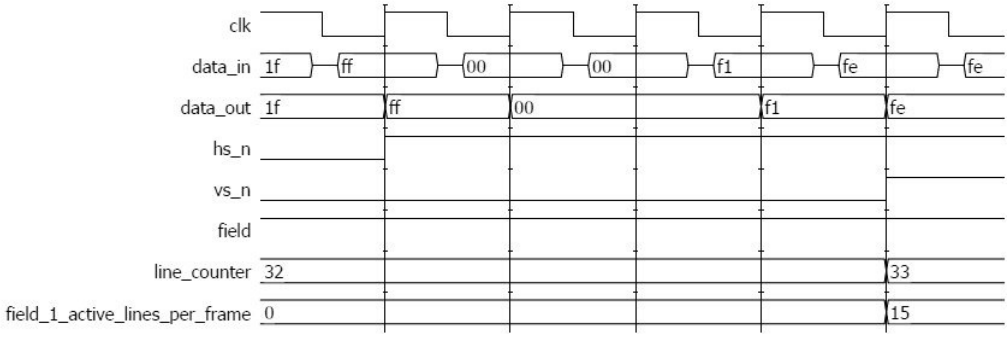


Figura 3-68: Detección de código 0hF1 y generación de señal field_1_actives_lines_per_frame.

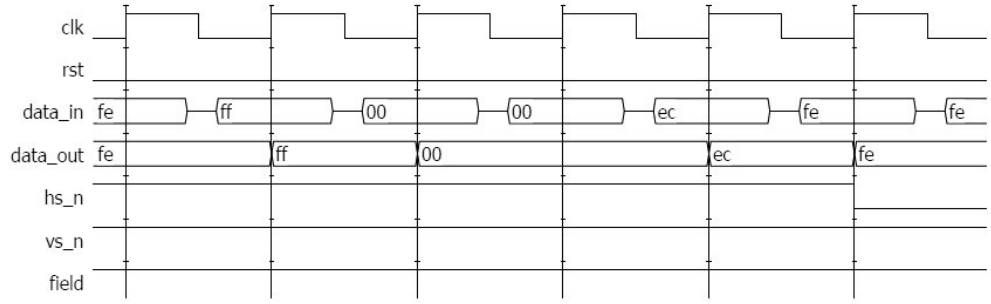


Figura 3-69: Detección de código 0hEC.

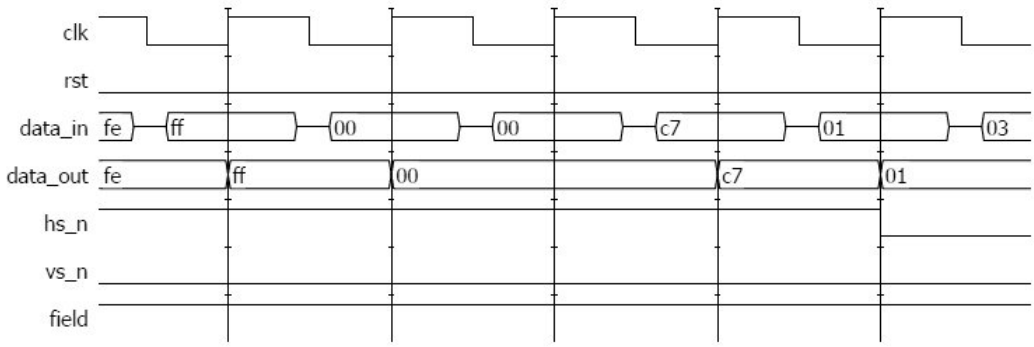


Figura 3-70: Detección de código 0hC7.

3.5.2 Módulo module_2

La principal preocupación de este módulo es almacenar correctamente en memoria los datos de las entradas de video, para verificar el correcto funcionamiento de esta cualidad se usa la capacidad del programa ModelSim para ver los datos de memorias descritas en HDL. Usando esta característica se obtienen la información de los espacios de memoria utilizados en el modelo de memoria DDR2, los cuales son mostrados en los anexos de esta memoria.

Los resultados obtenidos son correctos al comprobarse que las dos entradas generadas son guardadas en memoria de la forma esperada en cada uno de los espacios de memoria asignados para esto. En la Tabla 3-48, se indica para cada espacio de memoria utilizado por este módulo, el parámetro que define el inicio del buffer y la tabla que contiene los datos guardados en esta.

Utilización de espacio de memoria	Definición de dirección de inicio (Parámetro)	Contenidos de buffer (Anexo)
Entrada de Video 0 Buffer 0	START_ADDR_VIDEO_0_SCALING_IN_BUFFER_0	A.1
Entrada de Video 0 Buffer 1	START_ADDR_VIDEO_0_SCALING_IN_BUFFER_1	A.2
Entrada de Video 1 Buffer 0	START_ADDR_VIDEO_1_SCALING_IN_BUFFER_0	A.5
Entrada de Video 1 Buffer 1	START_ADDR_VIDEO_1_SCALING_IN_BUFFER_1	A.6

Tabla 3-48: Parámetros y datos de espacios de memoria utilizados por module_2 en simulación de comportamiento.

3.5.3 Módulo module_3

Para la comprobación del correcto comportamiento de este módulo se revisan los espacios de memoria usados por este módulo, primero revisando que la cantidad de elementos guardados en memoria sea el correcto y luego que los datos correspondan al redimensionamiento deseado.

La entrada de video0 corresponde a una reducción a la mitad de su tamaño horizontal y la mitad de su tamaño vertical. Esta ultima reducción es realizada por la elección del campo 0, por lo que este módulo no realiza redimensionamiento vertical. Así el número de componentes por líneas de la nueva imagen pasa a ser 40 y el de líneas a ser 14, obteniendo una imagen de 560 componentes. En cuanto a los datos almacenados se debe separar los datos por campos para apreciar correctamente cuales datos son eliminados y los que son almacenados en memoria. En la Tabla 3-49 se muestran los primeros 24 componentes de la entrada de video separados por canal y se identifica que componentes son usados en la imagen redimensionada.

Luma	03	07	0b	0f	13	17	1b	1f	23	27	2b	2f
Croma azul	01	09	11	19	21	29	-	-	-	-	-	-
Croma rojo	05	0d	15	1d	25	2d	-	-	-	-	-	-
Usado en imagen de salida	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No

Tabla 3-49: Primeros 24 componentes de entrada de video0 y utilización en imagen redimensionada.

Reordenando los datos para obtener la sucesión Cb, Y, Cr e Y, y extrapolando los datos de la tabla se puede saber que cada línea de la nueva imagen será igual y tendrá los componentes mostrados en la Tabla 3-50.

Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
01	03	05	0b	11	13	15	1b	21	23	25	2b	31	33	35	3b	41	43	45	4b
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
51	53	55	5b	61	63	65	6b	71	73	75	7b	01	03	05	0b	11	13	15	1b

Tabla 3-50: Componentes de cada línea de imagen redimensionada.

La entrada de video1 corresponde a un aumento al doble de su tamaño horizontal y el doble de su dimensión vertical. Así el número de componentes por líneas de la nueva imagen pasa a ser 80 y el de líneas a ser 16, obteniendo una imagen de 1280 componentes. Los datos guardados en memoria deben todos pertenecer a líneas iguales, con excepción de la última línea. Lo primero se debe a que las líneas de entrada son todas iguales lo que da este resultado. La última línea es calculada a partir de una línea de los datos de entrada y una línea de pixeles negros por lo que será distinta al resto de las líneas. Para explicar la composición de las líneas se muestra en la Tabla 3-51 los primeros 16 datos de entrada de una línea y los datos interpolados a partir de ellos para cada uno de los canales de color.

Luma	03	05	07	09	0b	0d	0f	11	03	05	07	09	0b	0d	0f	11
Croma azul	01	05	09	0d	11	15	19	1d	-	-	-	-	-	-	-	-
Croma rojo	05	09	0d	11	15	19	1d	21	-	-	-	-	-	-	-	-
Corresponde a Componente Interpolado	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si

Tabla 3-51: Interpolación de datos para linea de imagen redimensionada de entrada de video 0.

Ordenando y extrapolando lo mostrado en la Tabla 3-51 se puede llegar a la composición de una línea de la imagen redimensionada la que es mostrada en la Tabla 3-52.

Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
01	03	05	05	05	07	09	09	09	0b	0d	0d	0d	0f	11	11	11	13	15	15
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
15	17	19	19	19	1b	1d	1d	1d	1f	21	21	21	23	25	25	25	27	29	29
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
29	2b	2d	2d	2d	2f	31	31	31	33	35	35	35	37	39	39	39	3b	3d	3d
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
3d	3f	41	41	41	43	45	45	45	47	49	49	49	4b	4d	4d	64	4f	66	2f

Tabla 3-52: Composición de línea para imagen redimensionada de entrada de video 1.

La última línea de la imagen redimensionada corresponde a la suma entre la línea mostrada en la Tabla 3-52, con una línea de pixeles negros, cuyo resultado es dividido por dos, obteniendo lo mostrado en la Tabla 3-53.

Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
40	09	42	0a	42	0b	44	0c	44	0d	46	0e	46	0f	48	10	48	11	4a	12
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
4a	13	4c	14	4c	15	4e	16	4e	17	50	18	50	19	52	1a	52	1b	54	1c
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
54	1d	56	1e	56	1f	58	20	58	21	5a	22	5a	23	5c	24	5c	25	5e	26
Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y	Cb	Y	Cr	Y
5e	27	60	28	60	29	62	2a	62	2b	64	2c	64	2d	66	2e	72	2f	73	1f

Tabla 3-53: Composición de última línea en imagen redimensionada de entrada de video 1.

Los resultados obtenidos en la simulación corresponden a lo esperado.En la Tabla 3-54, se indica para cada espacio de memoria utilizado por este módulo, el parámetro que define el inicio del buffer y la tabla que contiene los datos guardados en esta.

Utilización de espacio de memoria	Definición de dirección de inicio (Parámetro)	Contenidos de buffer (Tabla)
Entrada de Video 0 Buffer 0	START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_0	A.3
Entrada de Video 0 Buffer 1	START_ADDR_VIDEO_0_SCALING_OUT_BUFFER_1	A.4
Entrada de Video 1 Buffer 0	START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_0	A.7
Entrada de Video 1 Buffer 1	START_ADDR_VIDEO_1_SCALING_OUT_BUFFER_1	A.8

Tabla 3-54: Parámetros y datos de espacios de memoria utilizados por scaler_out en simulación de comportamiento.

3.5.4 Module_4

Una vez estable las entradas de video, las direcciones de término de los espacios de memoria quedan con los valores enunciados en Tabla 3-55, en donde se agrega la cantidad de componentes almacena el espacio de memoria definido por cada dirección. De los valores obtenidos se comprueba el correcto cómputo de las nuevas dimensiones.

Señal	Valor	Componentes almacenados
ea_v0_si_b0	32'h00_00_04_60	1120
ea_v0_si_b1	32'h00_40_04_60	1120
ea_v0_so_b0	32'h00_80_02_30	560
ea_v0_so_b1	32'h00_c0_02_30	560
ea_v1_si_b0	32'h01_00_01_40	320
ea_v1_si_b1	32'h01_40_01_40	320
ea_v1_so_b0	32'h01_80_05_00	1280
ea_v1_so_b1	32'h01_c0_05_00	1280

Tabla 3-55: Resultados de cálculo de direcciones de término de buffers de video.

En Figura 3-71 se muestra cuando la señal m2_v0_mw se desactiva mientras m3_si_v0_mw está en bajo, teniendo como consecuencia el cambio de las señales mwo y de las direcciones de inicio y término, comprobando el diseño del manejo de las señales mwo y de las direcciones, cuando un módulo de escritura deja de trabajar en su buffer, estando el bloque de lectura inactivo. El caso donde el bloque de lectura está trabajando en memoria es mostrados en la Figura 3-72, comprobando que los espacios de memoria no son cambiados.

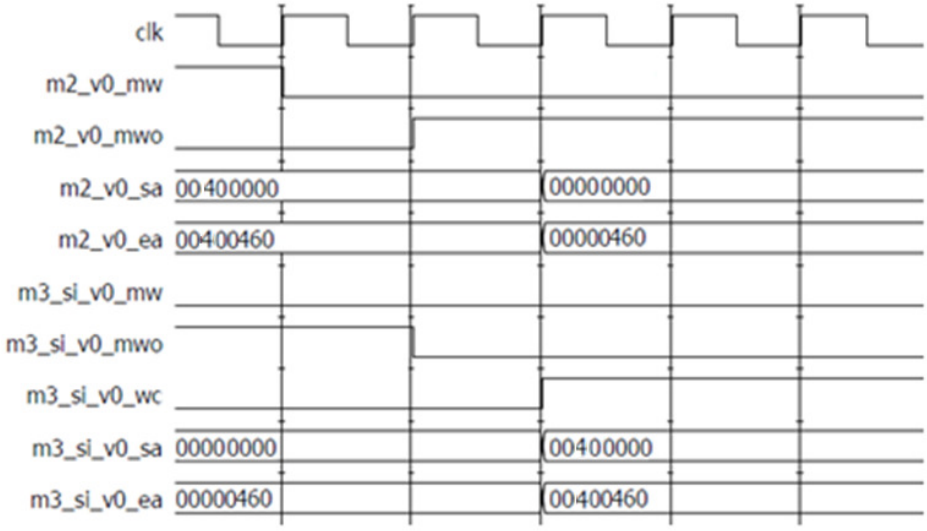


Figura 3-71: Simulación del manejo de buffers de video.

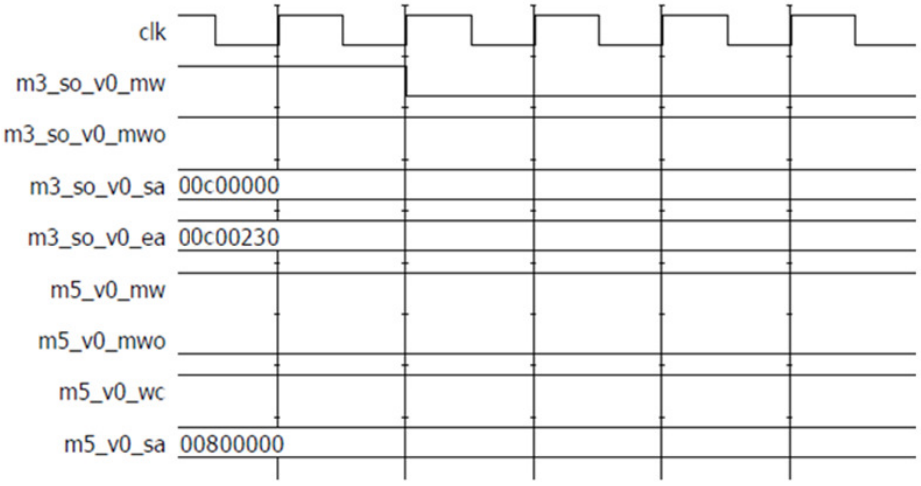


Figura 3-72: Simulación del manejo de buffers de video.

3.5.5 Módulos module_5 y module_6

Estos dispositivos funcionan correctamente, lo que se comprueba al ver los datos de la señal output_video de module_5. Esta señal durante los primeros 221 [us] de la simulación corresponde a cuadros compuestos en su totalidad por pixeles negros. Luego los cuadros cambian para mostrar a la imagen redimensionada de la entrada de video 1 en la parte inferior del cuadro y el resto es rellenado con pixeles negros, esto dura hasta los 439 [us] de la simulación, instante que se empieza a mostrar también la imagen redimensionada de la entrada de video 0 en la esquina superior izquierda. Las últimas dos líneas de la nueva imagen mostrada se sobreponen a la mitad de las dos primeras líneas de la imagen inicial.

3.6 Síntesis

Esta etapa del diseño se completo correctamente. El informe de síntesis entregado por el programa ISE, se encuentra la estimación de la utilización de recursos de la FPGA, cuyo valores son mostrados en la Tabla 3-56.

Recurso	Usado	Disponible	Porcentaje de Utilización
Número de *slices*	3.947	5.888	67%
Número de *slices* Flipflops	4.568	11.776	38%
Número de LUTs de 4 entradas	5.960	11.776	50%
Número de IOBs	89	372	23%
Número de BRAM	14	20	70%
Número de MULT18x18SIOs	16	24	80%

Tabla 3-56: Utilización de recursos para la FPGA post-síntesis.

En el informe además se entregan los recursos de LUTs, Flipflops y bloques de RAM usados por cada módulo, los que se muestran la Tabla 3-57.

Módulo	LUTs	Flipflops	BRAM
module_1	78	105	0
module_2	215	119	0
module_3	1.002	518	2
module_4	151	343	0
module_5	487	314	1
module_6	429	320	0
MPMC	1.393	2.125	9

Tabla 3-57: Utilización de recursos por módulo.

3.7 Asignación de restricciones al diseño

3.7.1 MPMC

Este módulo al ser generado por el programa Xilinx EDK, crea restricción de cronometraje y ubicación para algunas de sus secciones. Estas restricciones deben ser incorporadas en el archivo UCF, para el correcto funcionamiento del módulo.

3.7.2 Cronometraje

Existen tres entradas de reloj al sistemas, a cada una de estas se le impone una restricción de cronometraje de tipo PERIOD. Esta restricción obliga a que todas las rutas entre flipflops sincronizados por el reloj con la restricción tengan un retraso menor a PERIOD. En la Tabla 3-58, se muestra los periodos impuestos a cada entrada de reloj del sistema. La entrada de reloj sys_clk corresponde al reloj que sincroniza al módulo MPMC y todos los que tienen una interfaz NPI. video0_clk y video1_clk son las entradas de reloj que sincronizan a cada entrada de video, en consecuencia es utilizada por los módulos module_1 y module_2. La entrada de reloj video0_clk además es utilizada para generar sincronizar la salida de video, por lo que module_5 es afectado por la restricción sobre video0_clk.

Señal de reloj	Periodo (ps)	Frecuencia (MHz)
sys_clk	7.518	133
video0_clk	30.000	33,3
video1_clk	30.000	33,3

Tabla 3-58: Restricciones de cronometraje impuestas a entradas de reloj.

3.7.3 Ubicación de entradas y salida de video

Se asignan a las señales de entrada y salida de video los pines correspondientes a los pines usados por la plataforma de pruebas, en la Tabla 3-59 se muestra la asignación de pines a las señales de video del sistema.

Señales de entrada de video 0	Pin de la FPGA asignado	Señales de entrada de video 1	Pin de la FPGA asignado	Señales de salida de video	Pin de la FPGA asignado
video_0_data_in<0>	F19	video_1_data_in<0>	A18	video_data_out<0>	H19
video_0_data_in<1>	F18	video_1_data_in<1>	B17	video_data_out<1>	J18
video_0_data_in<2>	E22	video_1_data_in<2>	A17	video_data_out<2>	K18
video_0_data_in<3>	D22	video_1_data_in<3>	A16	video_data_out<3>	K17
video_0_data_in<4>	D21	video_1_data_in<4>	A15	video_data_out<4>	K19
video_0_data_in<5>	D20	video_1_data_in<5>	B15	video_data_out<5>	K20
video_0_data_in<6>	E17	video_1_data_in<6>	A14	video_data_out<6>	L19
video_0_data_in<7>	D18	video_1_data_in<7>	B13	video_data_out<7>	L18

Tabla 3-59: Asignación de pines de FPGA.

3.8 Etapa de mapeo

Esta etapa del diseño se completo correctamente generando la utilización definitiva de recursos de la FPGA, la cual se muestra en la Tabla 3-60. Además se muestra en la Tabla 3-61 la utilización de recursos por parte de cada uno de los módulos.

Recurso	Usado	Disponible	Porcentaje de utilización
Número de *slices de Flipflops* usados para lógica	5.150	11.776	43%
Número de LUTs de 4 entradas usados para lógica	5.736	11.776	48%
Número total de LUTs de 4 entradas Número usado como lógica Número usado como ruta atravesada Número usado como RAM de doble puerto Número usado como registro *Shift*	6.399 4.902 663 704 130	11.776	54%
Número de IOBs (Pines)	76	372	
Número de BUFGMUXs	7	24	29%
Número de DCMs	3	8	37%
Número de MULT18x18SIOs	16	20	80%
Número de RAM16BWEs	14	20	70%

Tabla 3-60: Utilización de recursos de FPGA post-mapeo.

Módulo	LUTs	Flipflops	BRAM
module_1	78	105	0
module_2	346	119	0
module_3	1.320	518	2
module_4	1.404	1052	0
module_5	517	314	1
module_6	516	320	0
MPMC	1.404	2125	9

Tabla 3-61: Utilización de recursos post-mapeo por módulo.

3.9 Etapa de ubicación y ruteo

Esta etapa del diseño fue completado correctamente, obteniendo la distribución definitiva del diseño en la FPGA.

3.10 Cronometraje estático post-ruteo

El diseño cumple con las restricciones de cronometraje impuestas en la etapa de asignación de restricciones. En el informe entregado por esta etapa se entregan las rutas con los mayores retrasos para cada restricción.

En la Tabla 3-62 se muestran los datos de la trayectoria de mayor retraso para la restricción sobre sys_clk. Esta afecta a la trayectoria generada por dos flipflops del módulo MPMC.

Fuente:	MPMC/DDR2_SDRAM/mpmc_core_0/mpmc_ctrl_path_0/arbiter_0/arb_pattern_start_0/ ctrl_refresh_i1(FF)				
Destino:	MPMC/DDR2_SDRAM/mpmc_core_0/mpmc_ctrl_path_0/ctrl_path_0/ctrl_bram_addr_1(FF)				
Requisito	**Retraso de trayectoria**	**Tiempo *de skew***	**Niveles de lógica**	**Porcentaje relacionado a lógica**	**Porcentaje relacionado a ruteo**
7,518 ns	7,176 ns	-0,276 ns	4	51,5%	48,5%

Tabla 3-62: Trayectoria de mayor retraso para restricción de cronometraje sobre señal sys_clk.

En la Tabla 3-63 se muestran los datos de la trayectoria de mayor retraso para la restricción sobre video0_clk. La trayectoria indicada corresponde a la generación de la señal v_region en el bloque module_5.

Fuente:	output_generator/component_counter/h_region_0(FF)				
Destino:	output_generator/line_counter/v_region_0(FF)				
Requisito	**Retraso de trayectoria**	**Tiempo *de skew***	**Niveles de lógica**	**Porcentaje relacionado a lógica**	**Porcentaje relacionado a ruteo**
30ns	9,929 ns	-0,067 ns	3	32,3%	67,7%

Tabla 3-63: Trayectoria de mayor retraso para restricción de cronometraje sobre señal video0_clk.

La Tabla 3-64 muestra los datos de la trayectoria de mayor retraso para la restricción sobre video1_clk. La trayectoria indicada corresponde a la señal is_interlaced generada en module_1, que es usada para construir la señal counter_in de module_2.

Fuente:	video1_in/is_interlaced(FF)				
Destino:	video1_to_mem/counter_in_4(FF)				
Requisito	**Retraso de trayectoria**	**Tiempo *de skew***	**Niveles de lógica**	**Porcentaje relacionado a lógica**	**Porcentaje relacionado a ruteo**
30ns	10,456ns	-0,181ns	5	45,7%	54,3%

Tabla 3-64: Trayectoria de mayor retraso para restricción de cronometraje sobre señal video1_clk.

3.11 Simulación post-ruteo

Esta simulación funciona correctamente. A diferencia de la simulación de comportamiento solo se tiene acceso a las señales del sistema y los elementos almacenados en el modelo de DDR2 SDRAM.

La configuración del sistema y las entradas de video generadas son las mismas usadas para la simulación de comportamiento, esto permite comprobar el correcto funcionamiento revisando los datos almacenados en el modelo de DDR2 y la salida de video. Los resultados obtenidos corresponden a los mismos obtenidos en la simulación de comportamiento.

4 ANALISIS DEL DISEÑO Y FUTURAS MEJORAS

El diseño realizado posee muchas limitaciones dadas por la plataforma de pruebas para la que fue desarrollado, pero además incorpora características que un sistema básico no tendría. Esta razón lleva a sugerir la posibilidad de dos posibles productos comerciales, un sistema básico de bajo costo y un sistema avanzado para trabajo profesional.

4.1 Sistema básico de bajo costo

Este sistema tendrá características fijas tales como el tipo de entradas y salida de video que es manejado, además el número de imágenes mostradas será fijo y del mismo tamaño cada una. Este sistema es ideal para monitoreo de cámaras de seguridad, las que suelen ser del mismo tipo y por consecuencia del mismo tipo de video.

Dadas las características enunciadas este sistema no requiere un redimensionado con aumento de escala, lo que permitirá eliminar una cantidad importante de lógica y hacer más compacto el sistema, permitiendo usar FPGAs de menor valor.

El sistema propuesto se muestra en la Figura 4-1, en esta cada señal de entrada es analizado por module_1 para determinar las señales de sincronización y las características de tamaño que serán usadas para bloquear la entrada de video en caso de no ser el estándar usado por el sistema. Luego los datos de video son entregados a scaler_out, que discrimina que datos son guardados en memoria según las señales de control de scaler_control. Estas señales corresponden a is_valid_data_in, pero modificadas para ser generadas a partir de las señales de sincronización producidas por module_1. Los módulos module_5 y module_6 son usados para generar la salida de video y la administración de los espacios de memoria es realizada por module_4.

Observando la Tabla 3-61, se puede determinar que la utilización de elementos lógicos y flipflops proviene mayoritariamente de los bloques MPMC, module_3 y module_4. El nuevo sistema como se comento elimina los bloques internos de module_3; scaler_in y scaler, y además reduce el tamaño de scaler_control, al eliminar las tareas de sincronizar scaler y administrar los coeficientes del filtro. Las etapas eliminadas de module_3 corresponden al 70% del dispositivo. En el caso de module_4 este deja de administrar los espacios de memoria usados por los bloques module_2, lo que reduce a la mitad su tamaño.

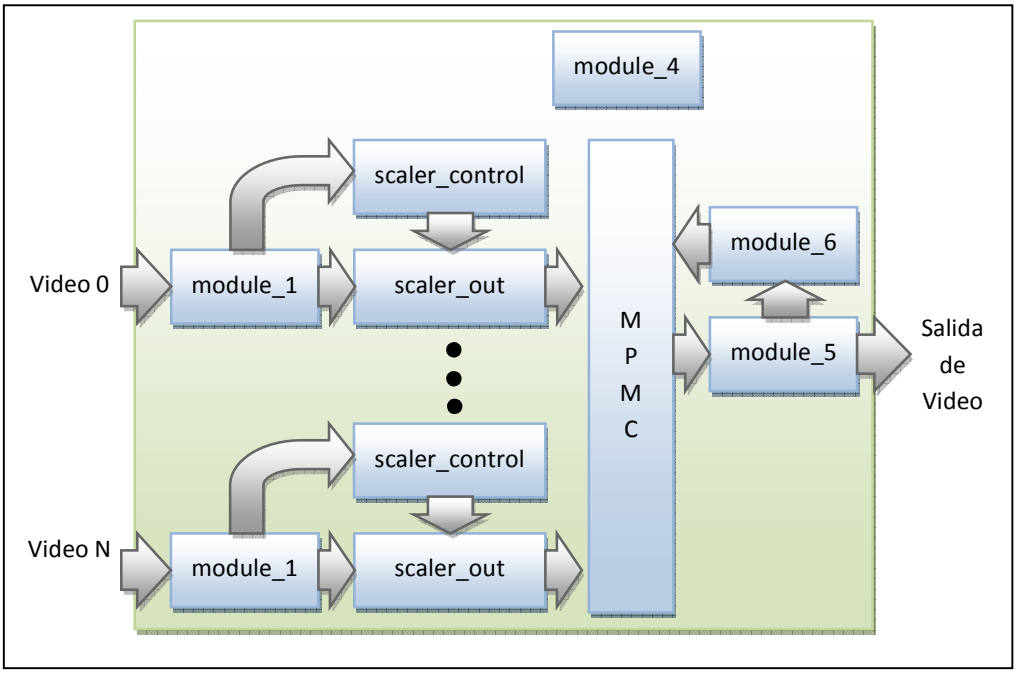


Figura 4-1: Sistema básico de bajo costo propuesto para implementación comercial.

Los bloques module_5 y module_6, también reducirían su tamaño, dado que las imágenes de cada entrada de video tiene una posiciones fija, lo que tendría como consecuencia la eliminación de las superposiciones entre imágenes y un manejo más simple de la salida de video.

Todas las características enunciadas permitirían tener un sistema de un máximo de 6 entradas de video, dado por el número de puertos del bloque MPMC, en una FPGA Spartan-3A de 700k o 1400k. A la fecha de la escritura de esta memoria estos dispositivos pueden tener un valor alrededor de los 40 dólares, lo que permitiría tener un sistema de bajo costo.

4.2 Sistema profesional

La propuesta para un sistema profesional parte con la definición de dos tipos de macro-bloques, uno encargado del procesamiento de las entradas de video y otro encargado del control del sistema y la generación de la salida de video. Esta separación del sistema tiene varias funciones, la primera de ellas es separar las funciones a distintas pastillas FPGA, permitiendo así tener una mayor cantidad de recursos, y en consecuencia da la posibilidad de tener una mayor cantidad de entradas de video en el dispositivo. La separación del sistema permite además una mejor utilización de recursos, al usar dispositivos específicos para cada funcionalidad.

En la Figura 4-2 se muestra un diagrama del sistema propuesto, donde se utilizan cuatro módulos encargados para el procesamiento de video, con los que se puede procesar hasta un máximo de 12 entradas de video.

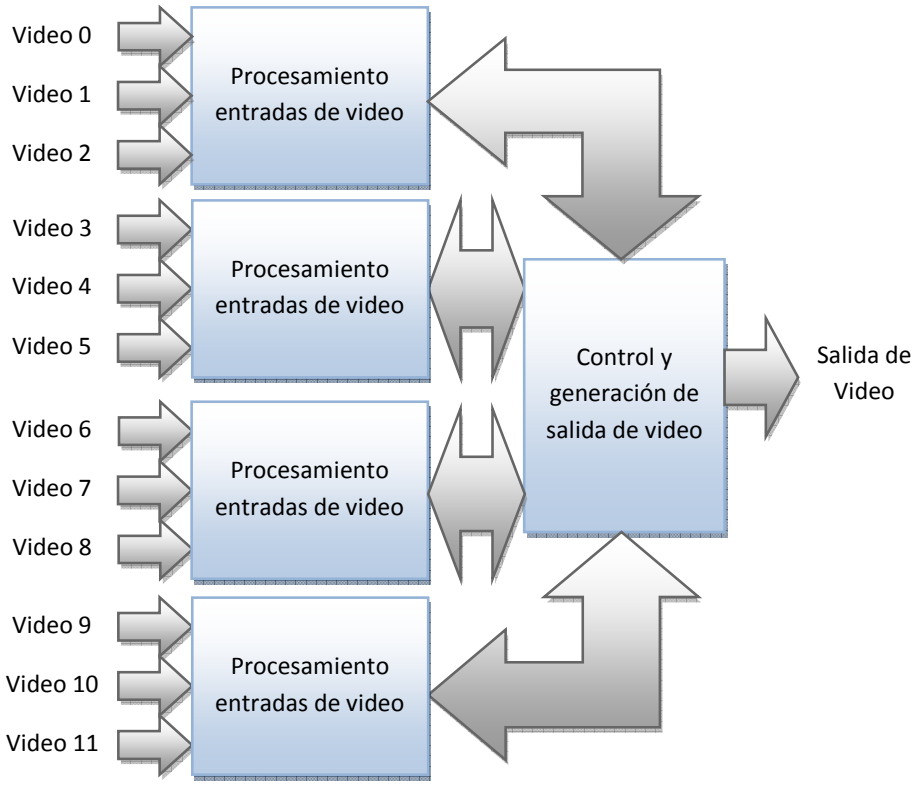


Figura 4-2: Sistema profesional propuesto.

4.2.1 Macro bloque de procesamiento de video

Este bloque usa la mayor parte de los módulos diseñados en esta memoria, con modificaciones y mejoras. En la Figura 4-3 se esquematizan los bloques usados por este macro bloque. Las señales de video parten siendo procesadas por module_1 el cual cumple las funciones de obtener las características de la imagen y obtener las señales de sincronización a partir de los códigos de temporización.

Una vez obtenidas las señales de sincronización estas son usadas por module_2 para almacenar los datos en memoria. Una mejora que se le debe añadir a este módulo es permitir la multiplexación en el tiempo de los datos obtenidos desde los bloques module_1, para utilizar un solo puerto del dispositivo MPMC. Esta nueva función de module_2 permite tener más entradas por FPGA.

Los datos almacenados en memoria son leídos por bloques module_3 que procesan cada uno una entrada de video. Este bloque debe ser modificado para poder trabajar con filtros de más de 4 taps verticales y 4 taps horizontales, para obtener una mayor calidad en el redimensionado. Además deben ser modificados el conjunto de coeficientes usados, debido a que estos corresponden a los de una interpolación lineal y existen conjuntos que permiten un mejor cálculo de los puntos interpolados. Este módulo debe ser considerado al momento de elegir el dispositivo FPGA a usar, debido a que requiere la mayor cantidad de multiplicadores y bloques de RAM del sistema. Además de los recursos usados por este bloque, es importante considerar la velocidad a la que trabaja ya que de este parámetro dependerá el multiplicador máximo que podrá tener el redimensionado para que no existan repeticiones de cuadros en la imagen de salida, por esta razón la frecuencia de trabajo de este bloque siempre debería ser la frecuencia de trabajo del dispositivo MPMC.

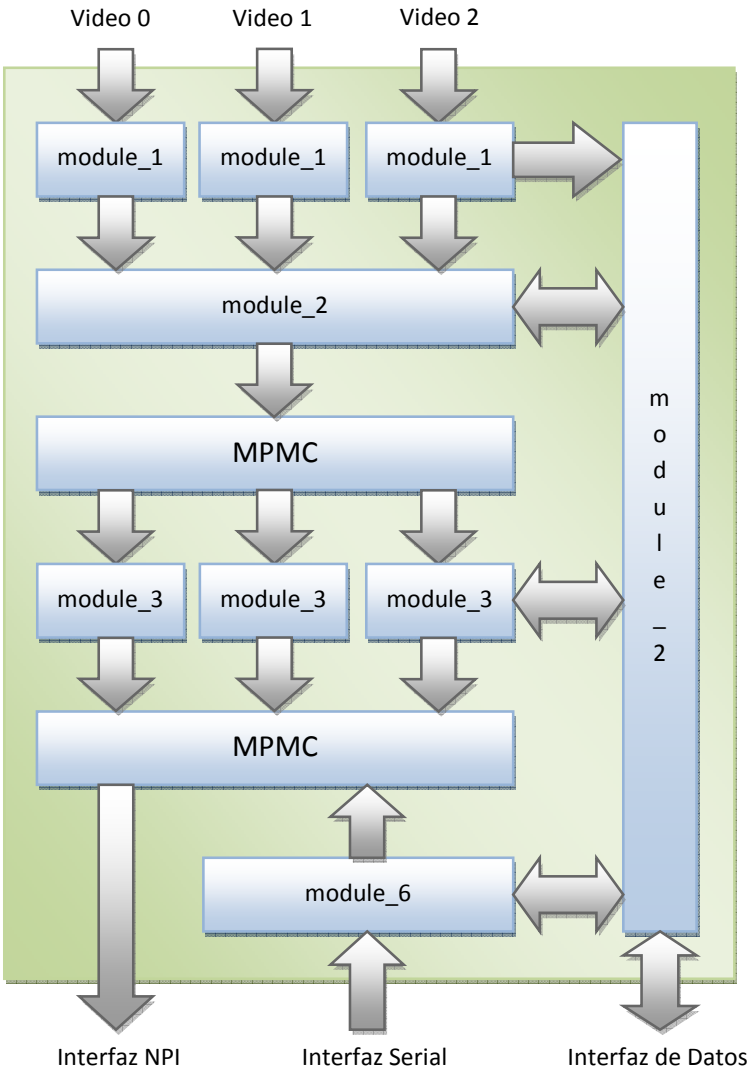


Figura 4-3: Módulo de procesamiento de video.

La petición de los datos de las imágenes redimensionadas es realizado por module_6. En este bloque debe ser modificada la interfaz de comandos paralelos, a una interfaz totalmente serial, que permita una menor utilización de pines del dispositivo FPGA.

El bloque module_4 administra los buffers de video usados en este macro-bloque, pero deja de tener la función de calcular las dimensiones para entregársela al módulo de control, el que atreves de una interfaz de datos modifica los registros de este módulo para seleccionar la configuración del sistema.

4.2.2 Macro bloque de control y generación de salida.

Este bloque cuenta con un procesador que permitirá generar un interfaz de usuario para el control del sistema, asignando los multiplicadores y divisores de dimensión de los distintos module_3, la ubicación de las imágenes en la salida y que entradas de video son mostradas ellas.

Por último module_5 es usado para la adquisición de datos de los dispositivos FPGAs encargados del redimensionado de las entradas de video, a través de las interfaces NPI y seriales, por lo que este bloque debe ser modificado para poder administrar los distintos puertos generados con estas interfaces.

En la Figura 4-4 se muestra el esquema del bloque de control y generación de la salida de video.

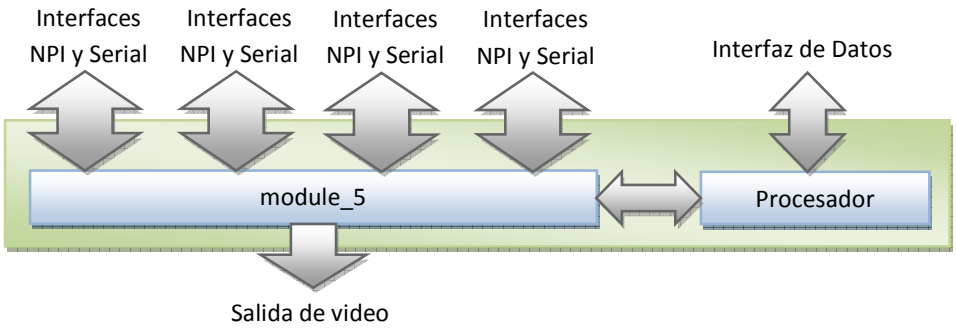


Figura 4-4: Módulo de control y generación de salida de video.

5 CONCLUSIONES

Con la investigación realizada se concluyó que el dispositivo idóneo para la realización de un *multiviewer* corresponde a FPGA debido a su característica de adaptabilidad y paralelización de procesos.

Las etapas de diseño en la FPGA fueron realizadas con éxito obteniendo los resultados deseados de cada una de ellas. El sistema generado permite el redimensionamiento de dos entradas de video, que pueden ser mostradas simultáneamente en una única pantalla. Entre las características configurables de este sistema están la elección de los tamaños de las imágenes de salida y la ubicación de estas dentro de la imagen.

La etapa de diseño de la plataforma de pruebas fue completada satisfactoriamente obteniéndose los archivos necesarios para la construcción de la placa de circuito impreso. La plataforma de pruebas permite la adquisición de video de definición estándar en los formatos PAL, NTSC y SECAM a través del dispositivo ADV7184. Estas señales son procesadas por el dispositivo FPGA de la familia Spartan-3A, para obtener una única salida de video que es codificada a diversos estándares de video en definición estándar y alta definición por el dispositivo ADV7391.

La etapa de construcción de la plataforma no logró ser realizada por problemas de disponibilidad del dispositivo ADV7401, que había sido elegido como primera opción para la decodificación de las entradas de video. Esto generó un atraso en el desarrollo total del proyecto, al tener que rediseñar la placa de circuito impreso para la utilización del dispositivo ADV7184 en vez del ADV7401. A este atraso se suman demoras en la entrega de componentes y la entrega de la placa de circuito impreso, que llevaron a no completar este objetivo.

El objetivo principal de esta memoria no ha sido cumplido en su totalidad pero el avance que se ha realizado es importante ya que se ha logrado dejar un diseño básico de un sistema *multiviewer*, al cual solo resta realizar pruebas con señales reales para comprobar su funcionamiento. Además se ha logrado dejar encaminado los cambios a realizar en el diseño para poder implementarlo como un producto comercial.

El objetivo secundario de esta memoria fue realizado a cabalidad ganando una experiencia invaluable en el área de diseño con FPGAs.

6 BIBLIOGRAFIA

[1] Institute of Electrical and Electronics Engineers. "IEEE Standard for Verilog® Hardware Description Language". IEEE Std 1364™-2005.

[2] Poynton, Charles. Digital Video and HDTV Algorithms and Interfaces. 5ª ed. Estados Unidos, Morgan Kaufmann Publishers, 2007. 694p

[3] Unión Internacional de Telecomunicaciones. "Interfaces para las señales de vídeo con componentes digitales en sistemas de televisión de 525 líneas y 625 líneas que funcionan en el nivel 4:2:2 de la Recomendación UIT-R BT.601". Recomendación UIT-R BT.656-5.

[4] Unión Internacional de Telecomunicaciones."Parámetros de codificación de televisión digital para estudios con formatos de imagen normal 4:3 y de pantalla ancha 16:9". Recomendación UIT-R BT.601-6.

[5] Xilinx " Spartan-3 Generation FPGA User Guide" [en línea]
< http://www.xilinx.com/support/documentation/user_guides/ug331.pdf>
[consulta: Marzo del 2009]

[6] Xilinx "XtremeDSP DSP48A for Spartan-3A DSP FPGAs" [en línea]
<http://www.xilinx.com/support/documentation/user_guides/ug431.pdf>
[consulta: Octubre del 2009]

[7]Xilinx " Spartan-3A/3AN FPGA Starter Kit Board User Guide"[en línea]
< http://www.xilinx.com/support/documentation/boards_and_kits/ug334.pdf >
[consulta: Marzo del 2009]

[8] Xilinx "Multi-Port Memory Controller (MPMC) Product Specification"[en línea]
<http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf>
[consulta: Abril del 2009]

[9] Berkeley Design Technology, Inc "Comparing FPGAs and DSPs for High-Performance DSP Applications" [en línea] presentado en conferencia GSPx, noviembre del 2006
<http://www.bdti.com/articles/20061101_gspx06_fpgas.pdf> [consulta: marzo del 2009]

[10] Xilinx " Divider Generator v3.0 Data Sheet"[en línea]
< http://www.xilinx.com/support/documentation/ip_documentation/div_gen_ds530.pdf>
[consulta: Mayo del 2009]

[11] Xilinx " Multiplier v11.2 Data Sheet"[en línea]
< http://www.xilinx.com/support/documentation/ip_documentation/mult_gen_ds255.pdf>
[consulta: Mayo del 2009]

[12]Xilinx " Video Scaler v2.0 Data Sheet "[en línea]
< http://www.xilinx.com/support/documentation/ip_documentation/v_scaler_ds724.pdf>
[consulta: Marzo del 2009]

[13] Xilinx " Video Scaler v2.0 User Guide"[en línea]
< http://www.xilinx.com/support/documentation/ip_documentation/v_scaler_ug678.pdf>
[consulta: Marzo del 2009]

[14] Xilinx " Xilinx ISE 10.1 Design Suite Software Manuals and Help - PDF Collection"[en línea]
<http://www.xilinx.com/itp/xilinx10/books/manuals.pdf> [consulta: Enero del 2009]

[15] Altera " Video and Image Processing Suite User Guide"[en línea]
< http://www.altera.com/literature/ug/ug_vip.pdf> [consulta: Marzo del 2009]

[16] Analog Devices "ADV7184 Datasheet"[en línea]
< http://www.analog.com/static/imported-files/data_sheets/ADV7184.pdf>
[consulta: Marzo del 2009]

[17]Analog Devices "ADV7401 Datasheet"[en línea]
< http://www.analog.com/static/imported-files/data_sheets/ADV7401.pdf>
[consulta: Marzo del 2009]

[18] Analog Devices " ADV7391 Datasheet"[en línea]
< http://www.analog.com/static/imported-files/data_sheets/ADV7390_7391_7392_7393.pdf>
[consulta: Marzo del 2009]

ANEXO A Datos almacenados en memoria para simulaciones

A.1 Buffer 32'h00_00_00_00

0:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
2:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
4:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
6:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
8:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
a:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
c:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
e:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
10:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
12:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
14:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
16:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
18:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
1a:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
1c:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
1e:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
20:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
22:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
24:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
26:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
28:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
2a:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
2c:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
2e:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
30:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
32:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
34:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
36:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
38:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
3a:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301
3c:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
3e:	5f5d5b59575553514f4d4b4947454341	7f7d7b79777573716f6d6b6967656361
40:	1f1d1b19171513110f0d0b0907050301	1f1d1b19171513110f0d0b0907050301
42:	3f3d3b39373533312f2d2b2927252321	5f5d5b59575553514f4d4b4947454341
44:	7f7d7b79777573716f6d6b6967656361	1f1d1b19171513110f0d0b0907050301

A.2 **Buffer 32'h00_40_00_00**

```
40000: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
40002: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
40004: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
40006: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
40008: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
4000a: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
4000c: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
4000e: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
40010: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
40012: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
40014: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
40016: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
40018: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
4001a: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
4001c: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
4001e: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
40020: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
40022: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
40024: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
40026: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
40028: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
4002a: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
4002c: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
4002e: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
40030: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
40032: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
40034: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
40036: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
40038: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
4003a: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
4003c: 1f1d1b19171513110f0d0b0907050301 3f3d3b39373533312f2d2b2927252321
4003e: 5f5d5b59575553514f4d4b4947454341 7f7d7b79777573716f6d6b6967656361
40040: 1f1d1b19171513110f0d0b0907050301 1f1d1b19171513110f0d0b0907050301
40042: 3f3d3b39373533312f2d2b2927252321 5f5d5b59575553514f4d4b4947454341
40044: 7f7d7b79777573716f6d6b6967656361 1f1d1b19171513110f0d0b0907050301
```

A.3 **Buffer 32'h00_80_00_00**

```
80000:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
80002:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
80004:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
80006:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
80008:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
8000a:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
8000c:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
8000e:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
80010:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
80012:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
80014:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
80016:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
80018:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
8001a:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
8001c:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
8001e:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
80020:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
80022:  1b1513110b0503017b7573716b656361  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

A.4 **Buffer 32'h00_c0_00_00**

```
c0000:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
c0002:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
c0004:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
c0006:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
c0008:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
c000a:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
c000c:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
c000e:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
c0010:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
c0012:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
c0014:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
c0016:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
c0018:  1b1513110b0503017b7573716b656361  3b3533312b2523211b1513110b050301
c001a:  7b7573716b6563615b5553514b454341  1b1513110b0503011b1513110b050301
c001c:  5b5553514b4543413b3533312b252321  1b1513110b0503017b7573716b656361
c001e:  3b3533312b2523211b1513110b050301  7b7573716b6563615b5553514b454341
c0020:  1b1513110b0503011b1513110b050301  5b5553514b4543413b3533312b252321
c0022:  1b1513110b0503017b7573716b656361  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

A.5 **Buffer 32'h01_00_00_00**

100000:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
100002:	0f0d0b09070503014f4d4b4947454341	2f2d2b29272523211f1d1b1917151311
100004:	4f4d4b49474543413f3d3b3937353331	1f1d1b19171513110f0d0b0907050301
100006:	3f3d3b39373533312f2d2b2927252321	0f0d0b09070503014f4d4b4947454341
100008:	2f2d2b29272523211f1d1b1917151311	4f4d4b49474543413f3d3b3937353331
10000a:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
10000c:	0f0d0b09070503014f4d4b4947454341	2f2d2b29272523211f1d1b1917151311
10000e:	4f4d4b49474543413f3d3b3937353331	1f1d1b19171513110f0d0b0907050301
100010:	3f3d3b39373533312f2d2b2927252321	0f0d0b09070503014f4d4b4947454341
100012:	2f2d2b29272523211f1d1b1917151311	4f4d4b49474543413f3d3b3937353331

A.6 **Buffer 32'h01_40_00_00**

140000:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
140002:	0f0d0b09070503014f4d4b4947454341	2f2d2b29272523211f1d1b1917151311
140004:	4f4d4b49474543413f3d3b3937353331	1f1d1b19171513110f0d0b0907050301
140006:	3f3d3b39373533312f2d2b2927252321	0f0d0b09070503014f4d4b4947454341
140008:	2f2d2b29272523211f1d1b1917151311	4f4d4b49474543413f3d3b3937353331
14000a:	1f1d1b19171513110f0d0b0907050301	3f3d3b39373533312f2d2b2927252321
14000c:	0f0d0b09070503014f4d4b4947454341	2f2d2b29272523211f1d1b1917151311
14000e:	4f4d4b49474543413f3d3b3937353331	1f1d1b19171513110f0d0b0907050301
140010:	3f3d3b39373533312f2d2b2927252321	0f0d0b09070503014f4d4b4947454341
140012:	2f2d2b29272523211f1d1b1917151311	4f4d4b49474543413f3d3b3937353331

A.7 **Buffer 32'h01_8_00_00**

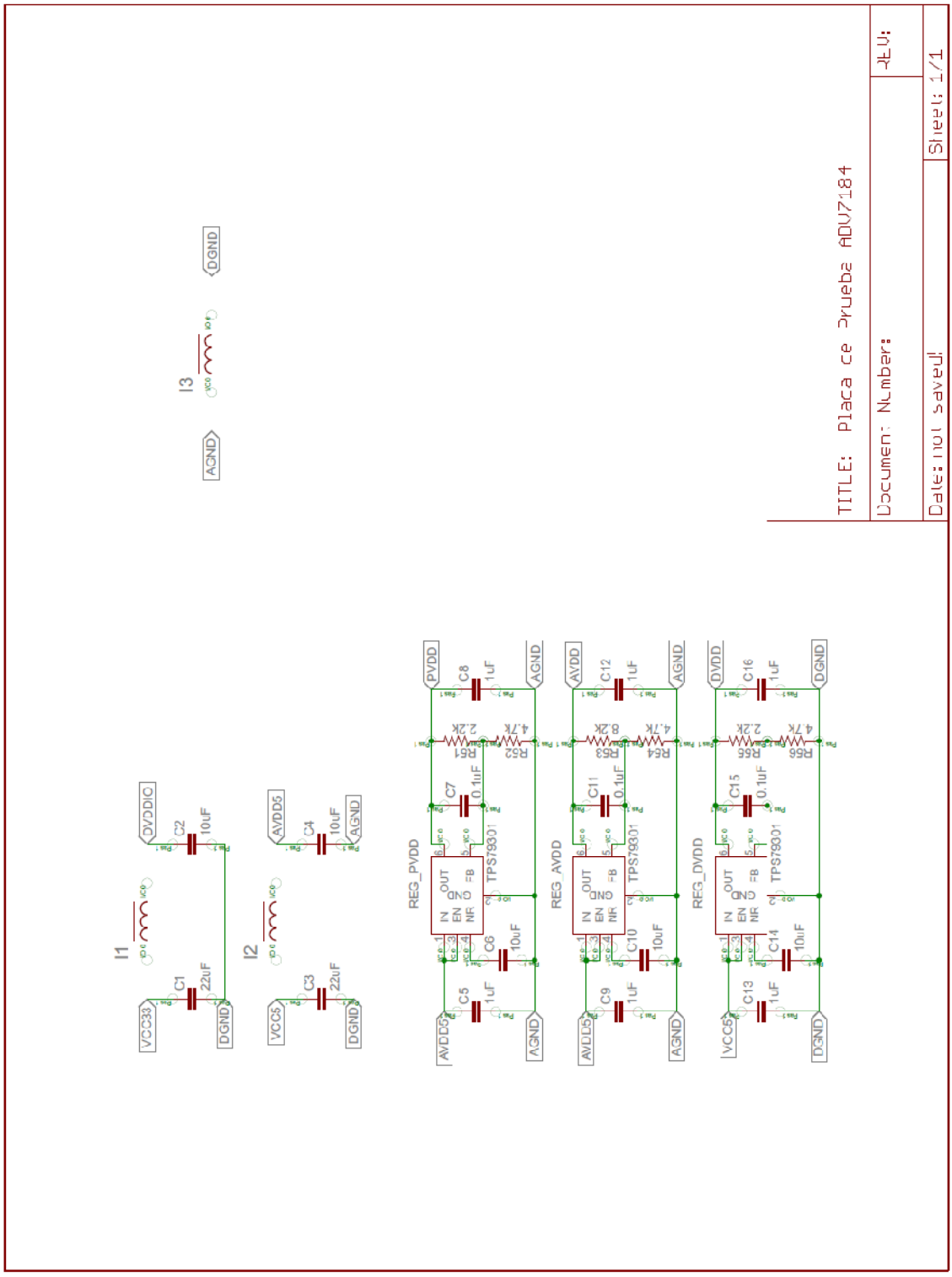
180000: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
180002: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
180004: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
180006: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
180008: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
18000a: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
18000c: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
18000e: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
180010: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
180012: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
180014: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
180016: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
180018: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
18001a: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
18001c: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
18001e: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
180020: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
180022: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
180024: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
180026: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
180028: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
18002a: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
18002c: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
18002e: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
180030: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
180032: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
180034: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
180036: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
180038: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
18003a: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
18003c: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
18003e: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
180040: 2f664f644d4d4b494949474545454341 11110f0d0d0d0b090909070505050301
180042: 21211f1d1d1d1b191919171515151311 31312f2d2d2d2b292929272525252321
180044: 41413f3d3d3d3b393939373535353331 2f664f644d4d4b494949474545454341
180046: 11110f0d0d0d0b090909070505050301 21211f1d1d1d1b191919171515151311
180048: 31312f2d2d2d2b292929272525252321 41413f3d3d3d3b393939373535353331
18004a: 2f664f644d4d4b494949474545454341 10480f460e460d440c440b420a420940
18004c: 1850174e164e154c144c134a124a1148 20581f561e561d541c541b521a521950
18004e: 2860275e265e255c245c235a225a2158 1f732f722e662d642c642b622a622960

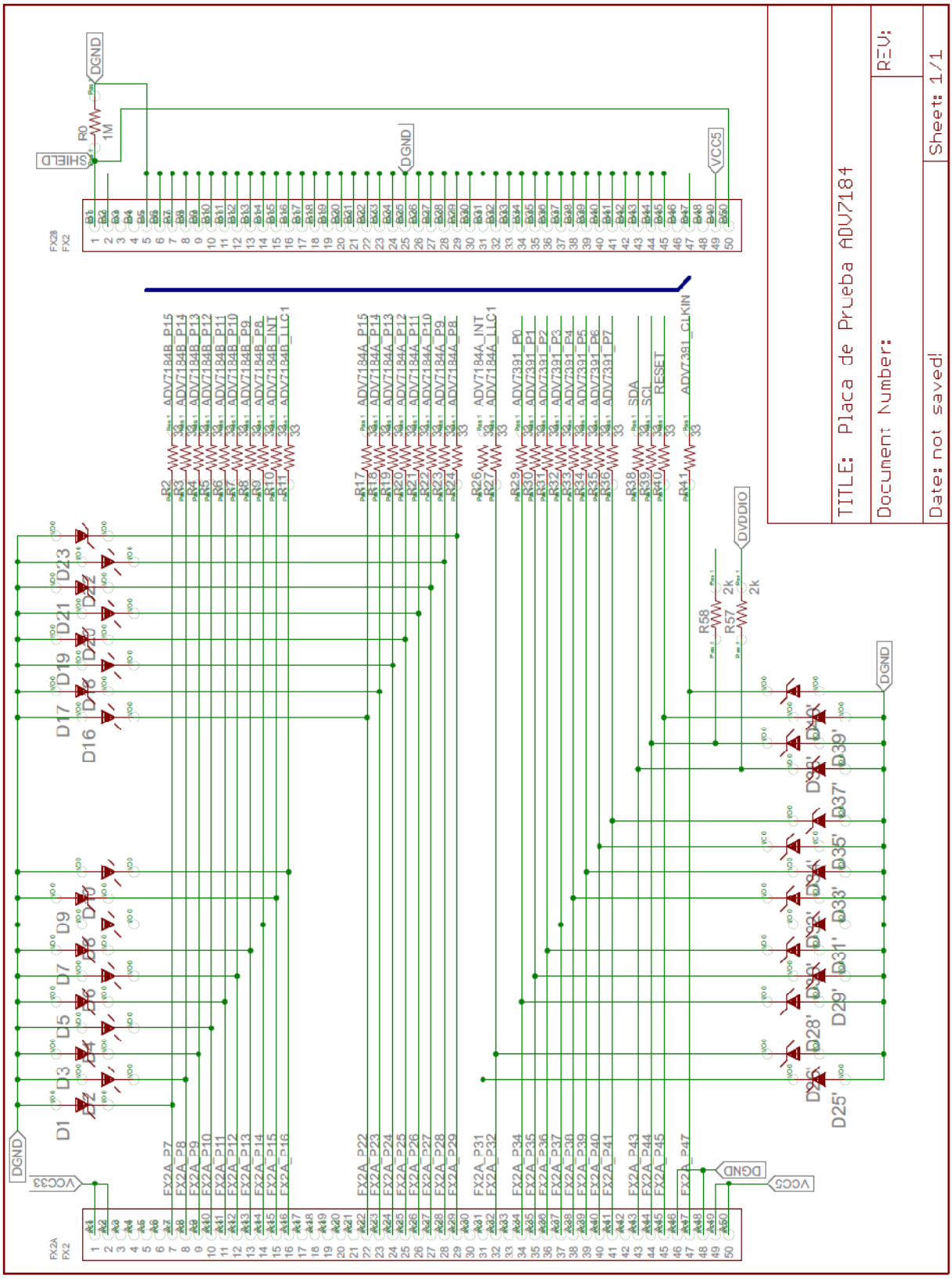
A.8 **Buffer 32'h01_c0_00_00**

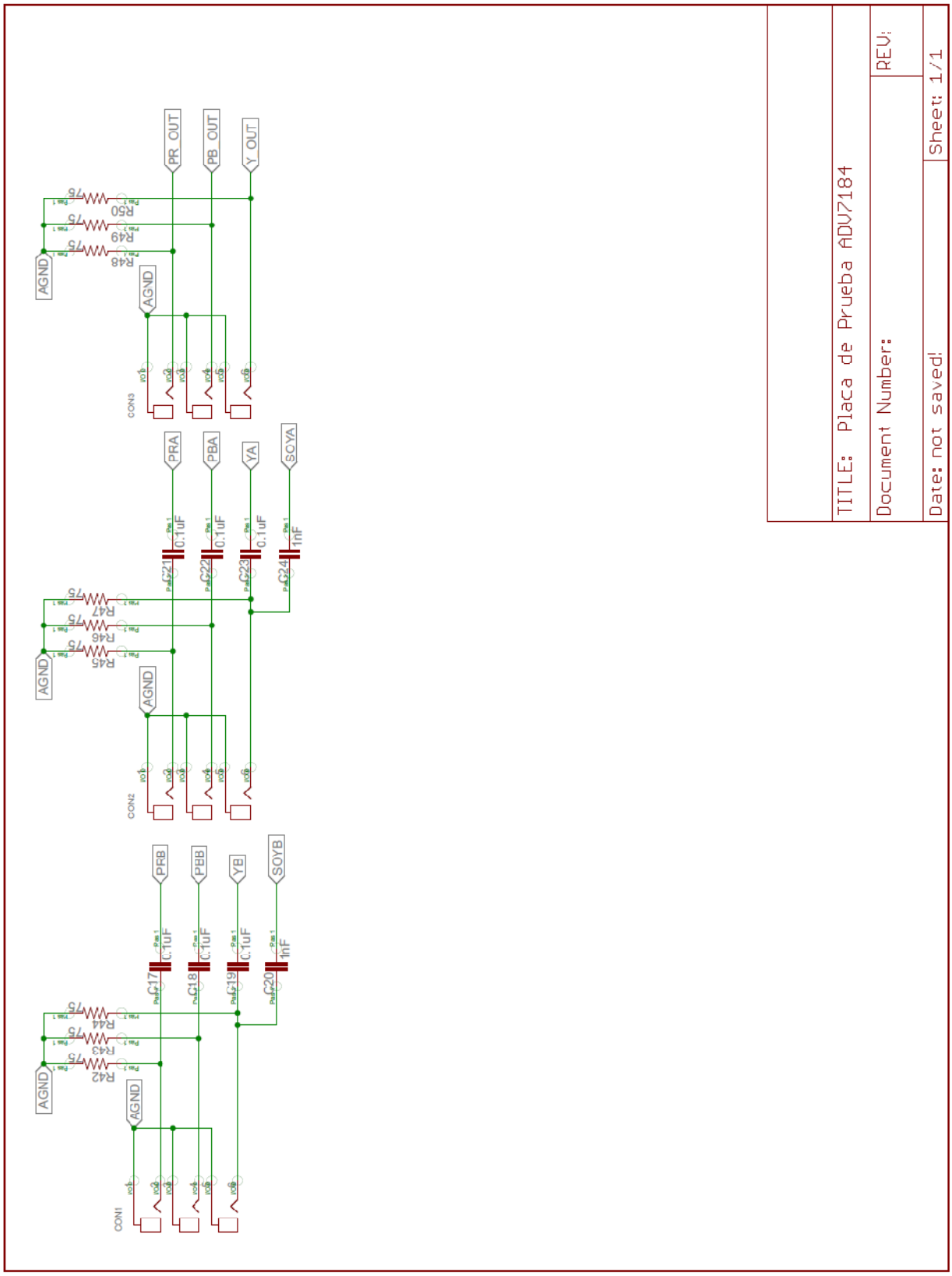
```
1c0000:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c0002:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c0004:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c0006:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c0008:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c000a:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c000c:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c000e:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c0010:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c0012:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c0014:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c0016:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c0018:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c001a:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c001c:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c001e:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c0020:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c0022:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c0024:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c0026:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c0028:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c002a:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c002c:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c002e:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c0030:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c0032:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c0034:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c0036:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c0038:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c003a:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c003c:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c003e:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c0040:  2f664f644d4d4b494949474545454341  11110f0d0d0d0b090909070505050301
1c0042:  21211f1d1d1d1b191919171515151311  31312f2d2d2d2b292929272525252321
1c0044:  41413f3d3d3d3b393939373535353331  2f664f644d4d4b494949474545454341
1c0046:  11110f0d0d0d0b090909070505050301  21211f1d1d1d1b191919171515151311
1c0048:  31312f2d2d2d2b292929272525252321  41413f3d3d3d3b393939373535353331
1c004a:  2f664f644d4d4b494949474545454341  10480f460e460d440c440b420a420940
1c004c:  1850174e164e154c144c134a124a1148  20581f561e561d541c541b521a521950
1c004e:  2860275e265e255c245c235a225a2158  1f732f722e662d642c642b622a622960
```

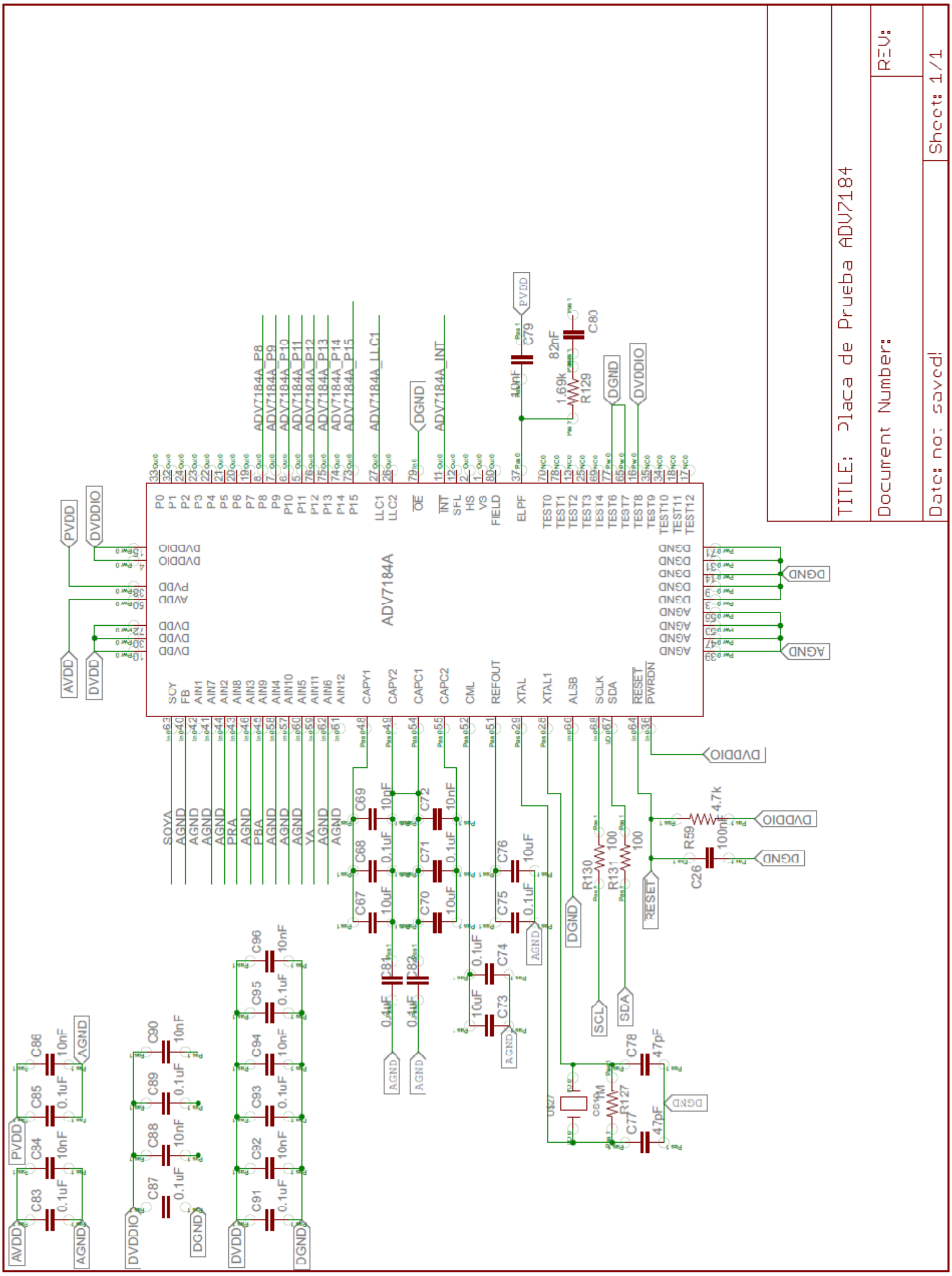
ANEXO B Diseño de plataforma de pruebas con ADV7184

B.1 Esquemáticos

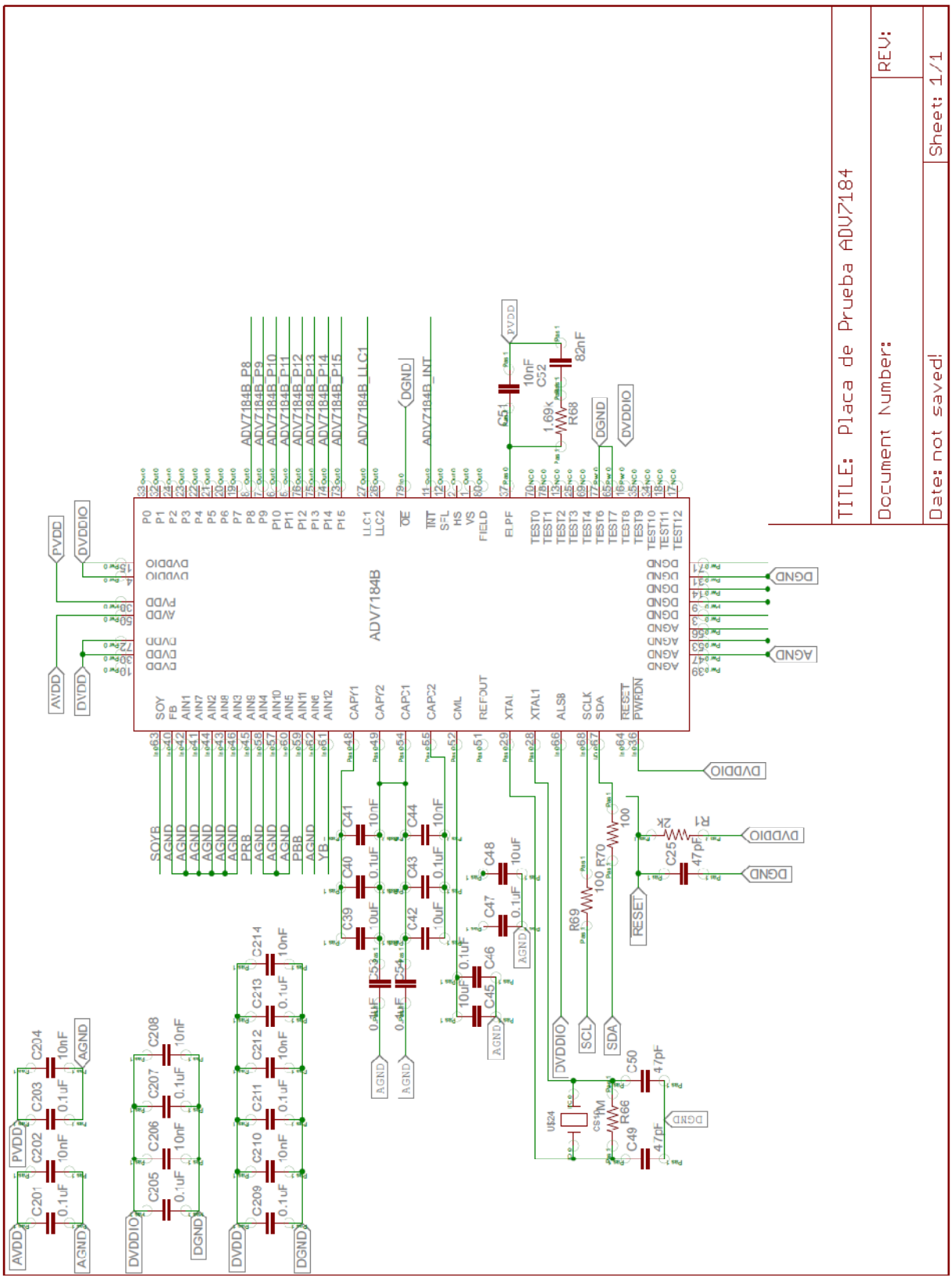


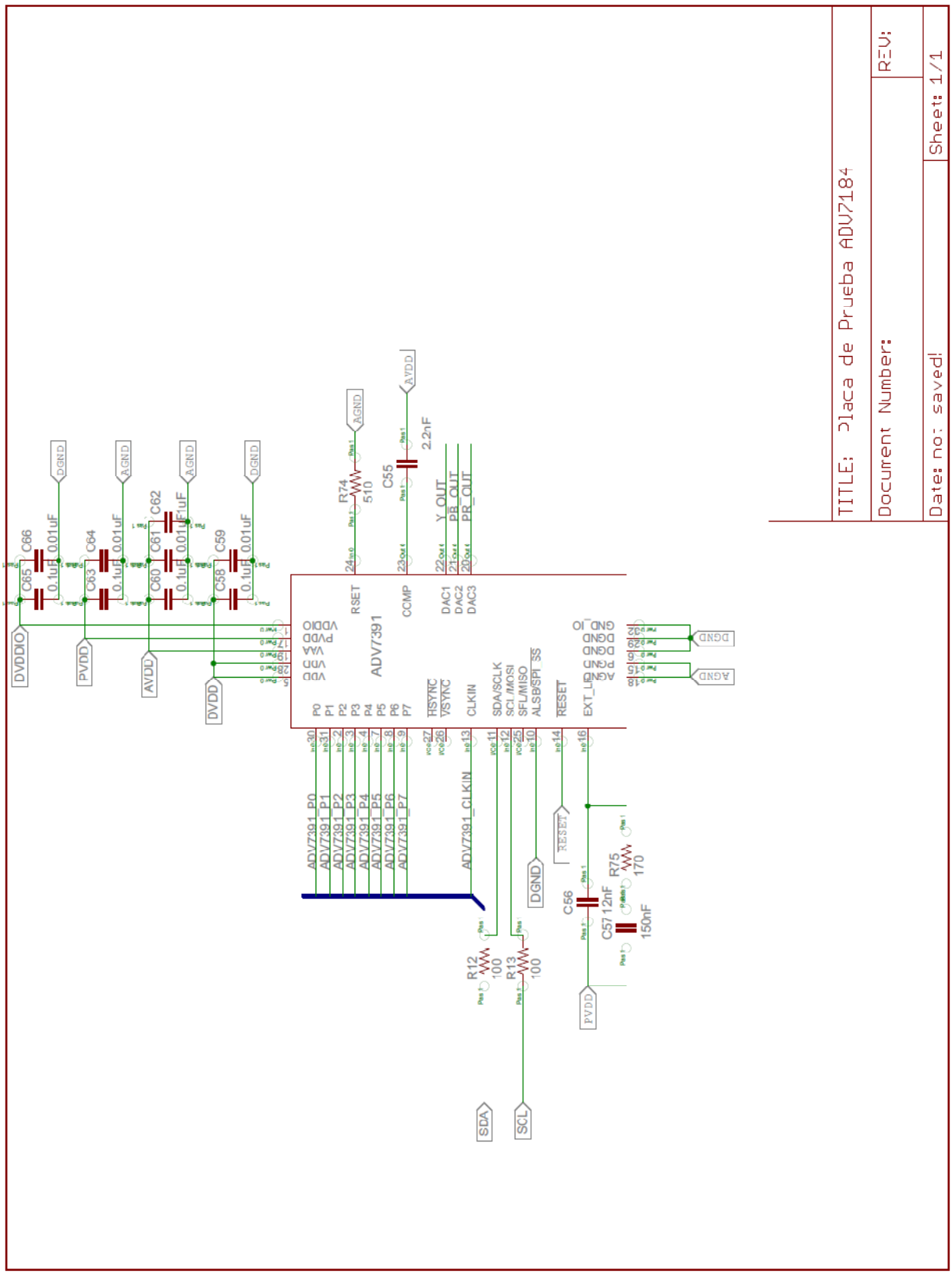

TITLE: Placa de Prueba ADV7184
Document Number:
REV:
Date: not saved!
Sheet: 1/1
DGND
VCC5
VCC33
DVDDIO
SHIELD
R0
1M
2k
SDA
SCL
RESET
ADV7391_CLKIN
FX2A
FX2B
FX2


TITLE: Placa de Prueba ADV7184
Document Number:
REV:
Date: not saved!
Sheet: 1/1
PR_OUT
PB_OUT
Y_OUT
AGND
R48
R49
R50
75
CON3
PRA
PBA
YA
SCYA
C21
C22
C23
C24
0.1uF
1nF
R45
R46
R47
CON2
PRB
PBB
YB
SOYB
C17
C18
C19
C20
R42
R43
R44
CON1



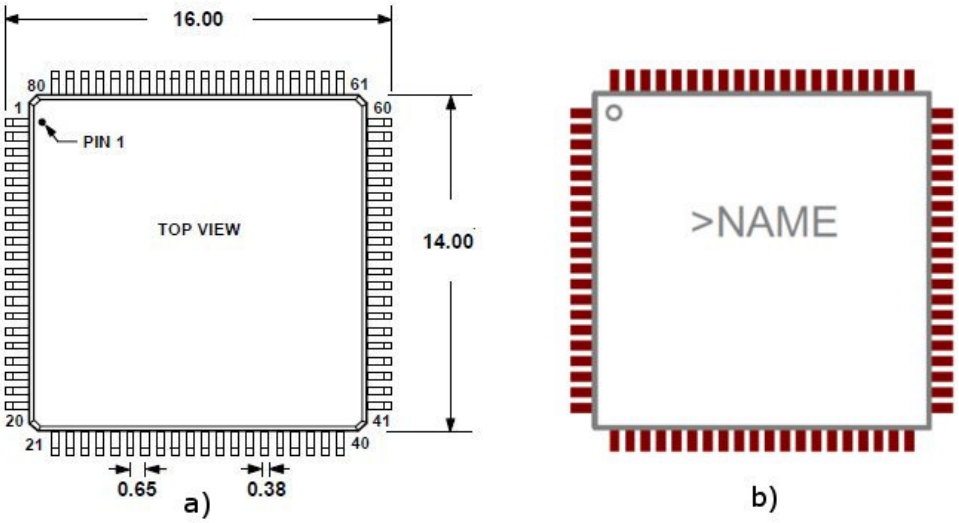
TITLE: Placa de Prueba ADV7184
Document Number:
REV:
Date: not saved!
Sheet: 1/1
ADV7184A


TITLE: Placa de Prueba ADV7184
Document Number:
REV:
Date: not saved!
Sheet: 1/1
ADV7184B
AVDD
DVDD
PVDD
DVDDIO
AGND
DGND
RESET
SCL
SDA
ADV7184B_P8
ADV7184B_P9
ADV7184B_P10
ADV7184B_P11
ADV7184B_P12
ADV7184B_P13
ADV7184B_P14
ADV7184B_P15
ADV7184B_LLC1
ADV7184B_INT

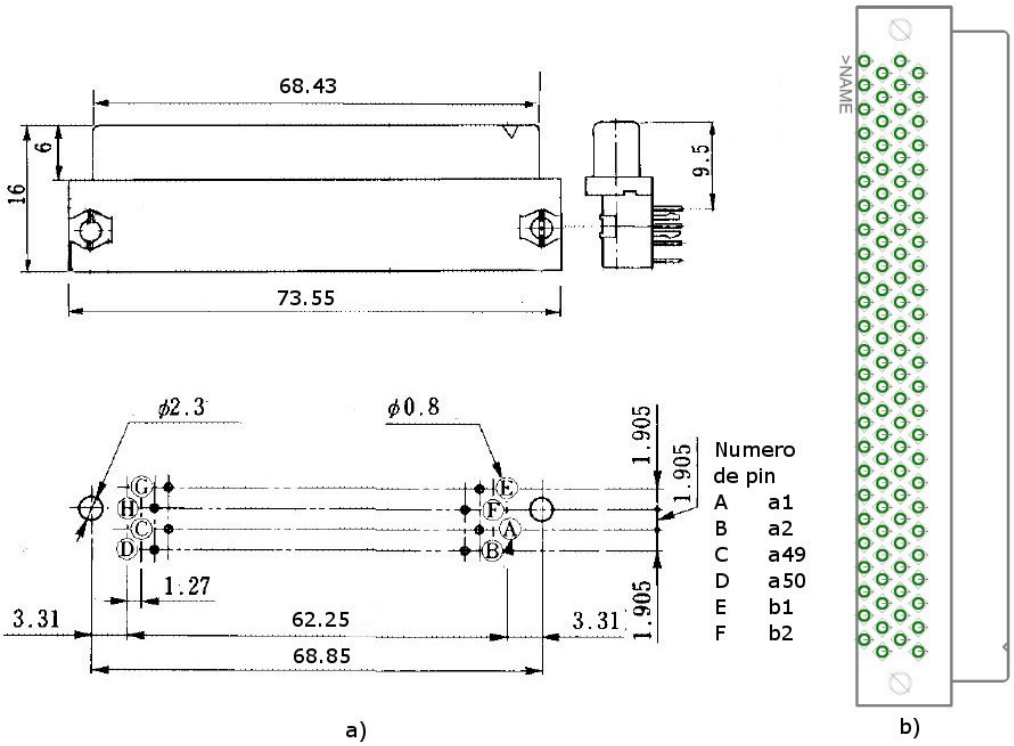

TITLE: Placa de Prueba ADV7184
Document Number:
REV:
Date: no: saved!
Sheet: 1/1
ADV7391
C65 0.1uF
C66 0.01uF
C63 0.1uF
C64 0.01uF
C60 0.1uF
C61 0.01uF
C58 0.1uF
C59 0.01uF
C62 0.1uF
DVDDIO
PVDD
AVDD
DVDD
DGND
AGND
R74 510
C55 2.2nF
Y_OUT
PB_OUT
PR_OUT
RSET
CCMP
DAC1
DAC2
DAC3
HSYNC
VSYNC
CLKIN
SDA/SCLK
SCL/MOSI
SFL/MISO
ALSB/SPI_SS
RESET
EXT_LF
ADV7391_P0
ADV7391_P1
ADV7391_P2
ADV7391_P3
ADV7391_P4
ADV7391_P5
ADV7391_P6
ADV7391_P7
ADV7391_CLKIN
R12 100
R13 100
C56 12nF
C57 150nF
R75 170
SDA
SCL

B.2 Dimensiones de empaquetamientos de componentes y representación en Eagle

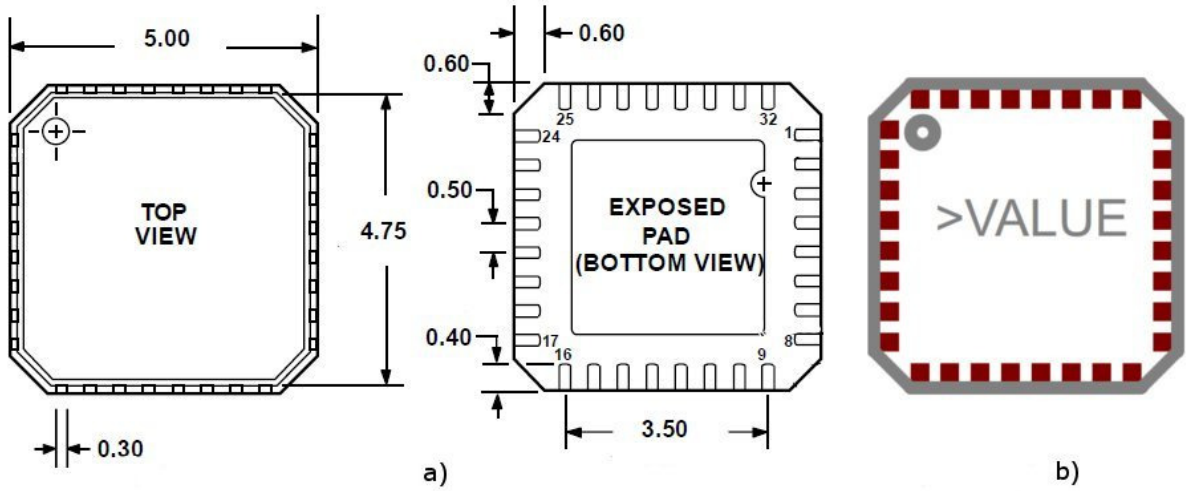
B.2.1 ADV7184



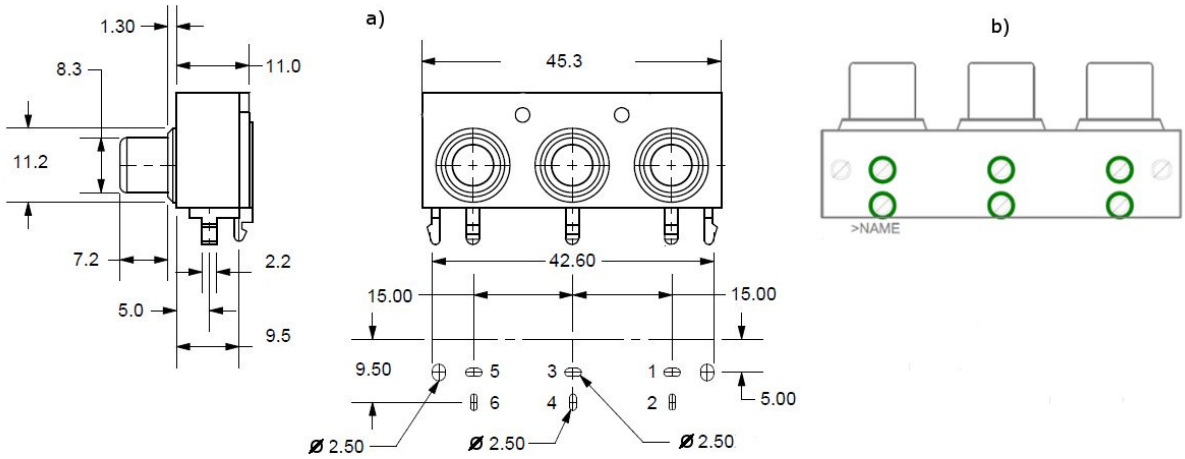
B.2.2 Conector Hirose FX2 de 100 pines



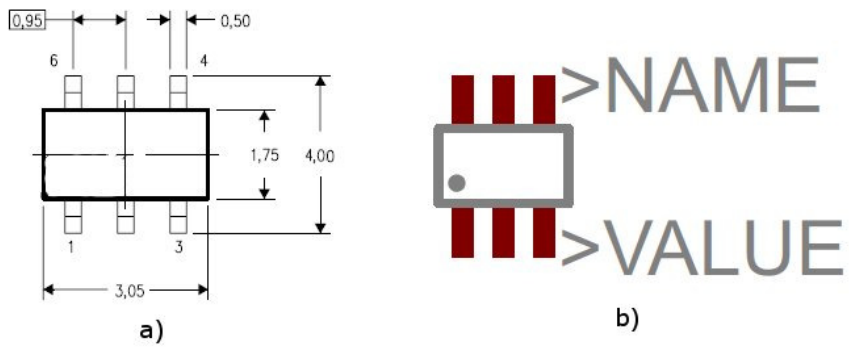
B.2.3 ADV7391



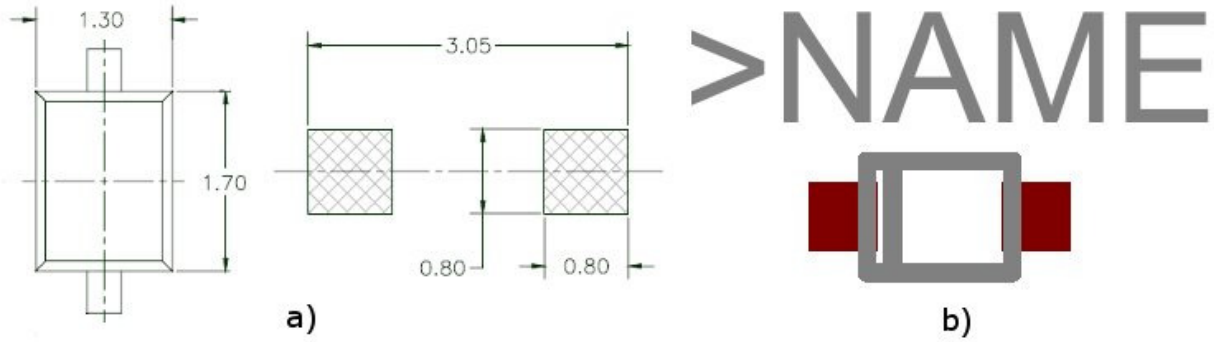
B.2.4 Conector RCA



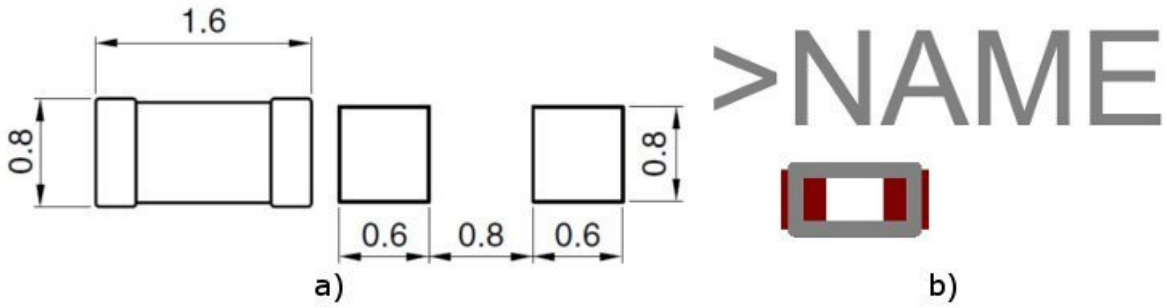
B.2.5 Regulador de voltaje TRS79301



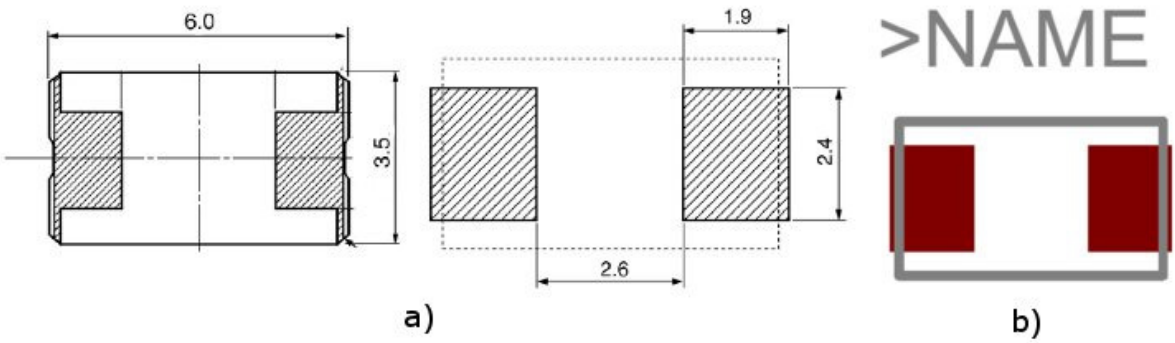
B.2.6 Diodo Zener MM3Z3V3CCT



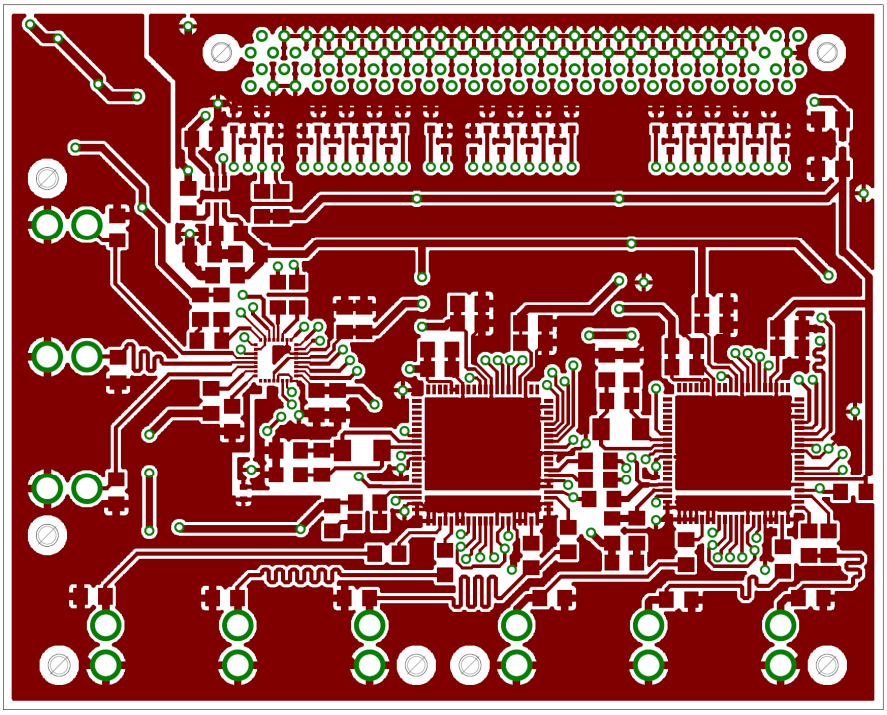
B.2.7 Ferrite Bead MMZ1608

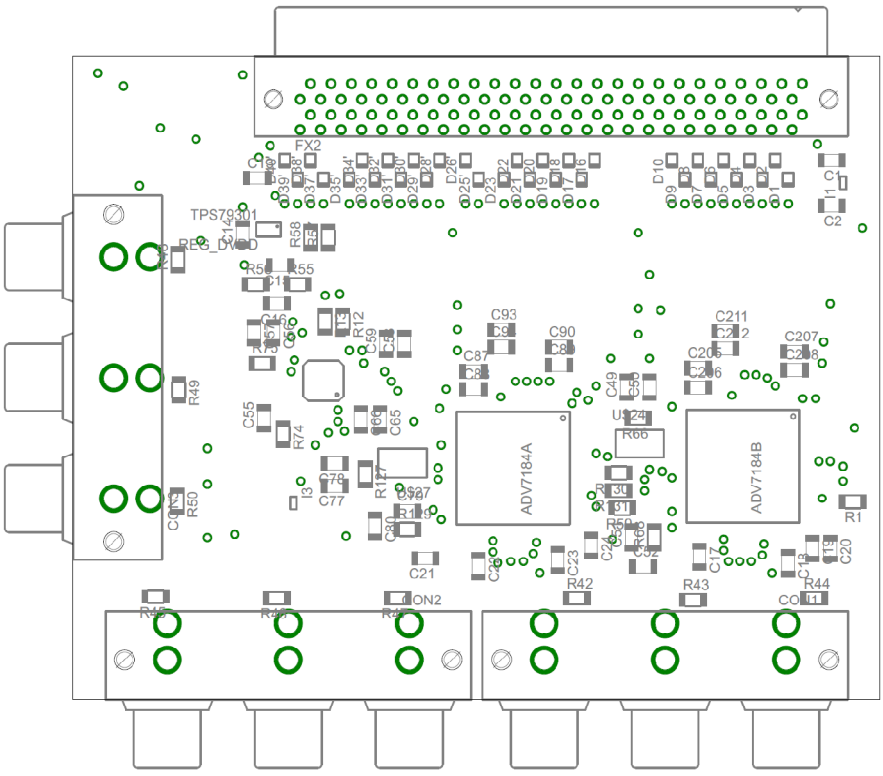


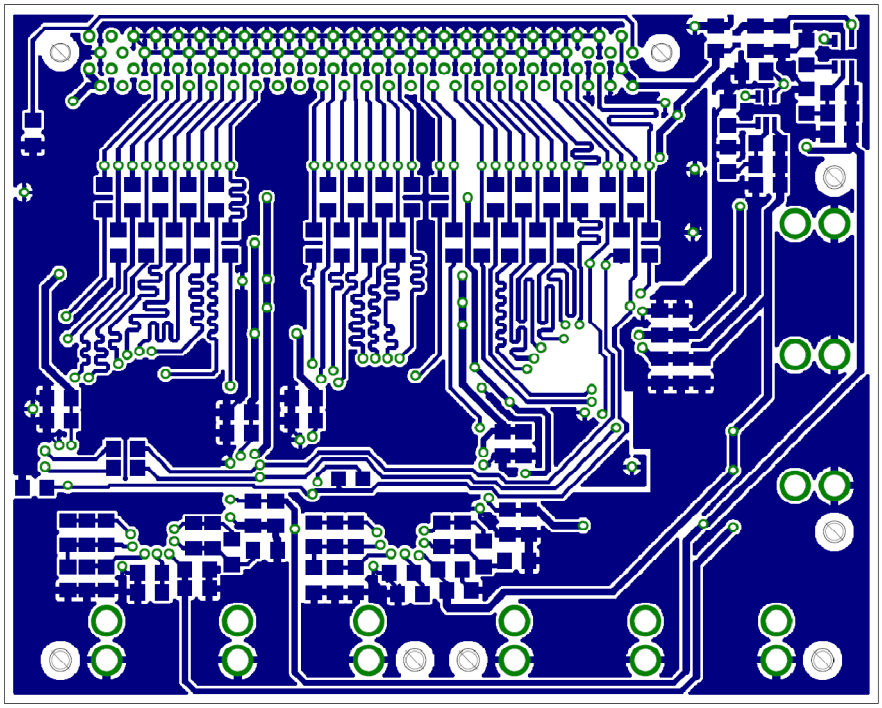
B.2.8 Cristal CS10

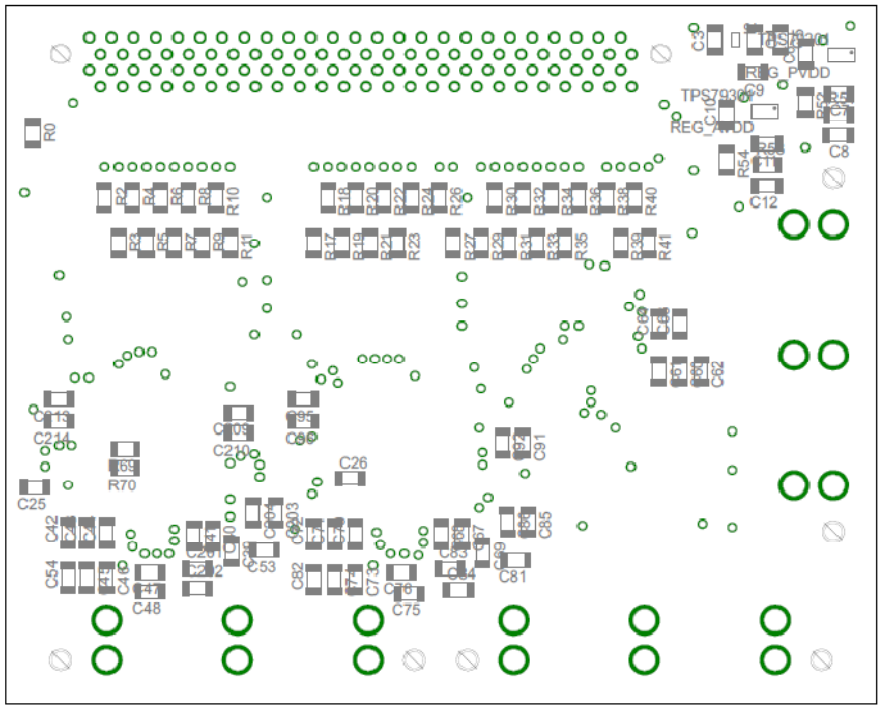


B.3 **Ruteo de placa de prueba**








REG_PVDD
TPS79330
REG_AVDD
C8
C9
C12
R0
R70
C25
C26
C48
C53
C75
C81
C91