UNIVERSIDAD POLITÉCNICA DE CARTAGENA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Ingeniero de Telecomunicación

IMPLEMENTACIÓN EN VHDL DE UN ADAPTADOR DE VÍDEO CÁMARA A MONITOR VGA ESTÁNDAR PARA APLICACIONES DE PROCESAMIENTO DE VÍDEO EN TIEMPO REAL

MIGUEL MORA ANDREU

Cartagena, JUNIO 2009

DIRECTORES: JOSE JAVIER MARTÍNEZ ÁLVAREZ
FRANCISCO JAVIER GRARRIGÓS GUERRERO
Dpto. Electrónica, Tecnología de computadores y Proyectos

Autor	Miguel Mora Andreu
E-mail del Autor	miguelmoraandreu@hotmail.com
Director(es)	Jose Javier Martínez Álvarez Francisco Javier Garrigós Guerrero
E-mail del Director	jjavier.martinez@upct.es Javier.Garrigos@upct.es
Título del PFC **(en inglés)**	Implementación en VHDL de un adaptador de vídeo cámara a monitor VGA estándar para aplicaciones de procesamiento de vídeo en tiempo real. VHDL implementation of a video camera adapter to standard VGA monitor for video processing applications in real time
Resumen	El objetivo de este proyecto es desarrollar una plataforma de prototipado que permita la realización de sistemas de procesamiento de vídeo en tiempo real. Se pretende adaptar la salida de una cámara digital para poder ver las imágenes capturadas en un monitor que funcione con el estándar VGA, estableciendo el soporte para un procesado intermedio de la información. Para ello se programará una arquitectura en una FPGA, que se encargará de sincronizar ambos componentes.
Titulación	Ingeniero de Telecomunicación
Intensificación	
Departamento	Electrónica, Tecnología de Computadores y Proyectos
Fecha de Presentación	Julio de 2009

A mis padres…

Sin vosotros esto no sería posible.

AGRADECIMIENTOS

Son muchas las personas que se merecen un mínimo de reconocimiento por haberme ayudado de una manera u otra a poder terminar mi carrera. En primer lugar, mis más sinceros agradecimientos para mi director principal del proyecto, José Javier, con el que a lo largo de estos dos años he vivido momentos malos, de incertidumbre, frustración y desesperación con el hardware y , sin embargo, han sido muchos más los momentos de alegría, trabajo en equipo, risas, y, por qué no decirlo, "de buen rollo". Pues eso José, empezaste siendo mi profesor y acabaste siendo mi amigo, gracias por todo.

Por otra parte, tengo a todos aquellos que me habéis apoyado durante tantos años. Agradezco muchísimo a mis padres el esfuerzo, apoyo y comprensión que me han dado (sé que ha sido difícil soportarme durante los exámenes). También quiero reconocer el mérito del resto de mi familia: tías y tíos, primos y primas, y abuelas; a todos muchas gracias.

No me olvido de mis amigos. La verdad es que son muchos los que he hecho en la Universidad. Gracias a todos los de mi clase, cada uno me habéis aportado algo nuevo y que me ha enriquecido. Citaros a todos es complicado, pero se va a intentar: gracias a David, Judith, Víctor, Alex (Ortuño), María (delegada forever), Mercedes, Guirlando, Montoro y Aurora, Piñero, Fran, Raquel, Lidia, Antonio, Iván (mecánica), Magalí, Ángel, Andreu, Juanito, Raúl, Cheroki, etc. Gracias también a mis amigos de toda la vida: Perico, Pepe, Felipe, Rafa, Eva, Carmen, Mónica, María, Mila, Miriam, etc.

Por último, una especial mención a las dos personas que más tiempo pasan a mi lado y que más me han apoyado: mi hermano David y mi novia Zaira. Gracias.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. Introducción

1.1 OBJETIVOS Y FUNDAMENTOS DEL PROYECTO

El objetivo de este proyecto es desarrollar una plataforma de prototipado que permita la realización de sistemas de procesamiento de vídeo en tiempo real. Se pretende adaptar la salida de una cámara digital para poder ver las imágenes capturadas en un monitor que funcione con el estándar VGA, estableciendo el soporte para un procesado intermedio de la información. Para ello se programará una arquitectura en una FPGA, que se encargará de sincronizar ambos componentes.

El principal problema que se presenta es la incompatibilidad entre la cámara digital y el monitor VGA, debido a que utilizan una frecuencia y resolución distintas. El monitor VGA trabaja con una frecuencia de 60 Hz y una resolución de 800x525 mientras que la cámara digital funciona a 30 Hz y 858x525 de resolución. Por tanto, surge la necesidad de establecer un circuito de sincronización entre ambos componentes.

Se propone programar en una FPGA una arquitectura que transforme la resolución y frecuencia de la cámara digital, en la resolución y frecuencia propias del estándar VGA. Esta arquitectura debe tener como entradas las señales que proporcione la cámara y como salidas las señales que componen el bus VGA de entrada al monitor. La programación de la FPGA se hará con el lenguaje de descripción hardware VHDL.

En referencia a los componentes utilizados, se debe tener en cuenta que cada cámara digital presenta unas determinadas características en función del sensor de imagen que utilicen. En nuestro caso, el desarrollo del proyecto se va a hacer para un sensor CMOS determinado. En cuanto al monitor será válido cualquiera que sea compatible con el estándar VGA.

Por último, en cuanto a la utilidad del proyecto, el sistema se va a emplear como plataforma de desarrollo y prototipado de aplicaciones de procesamiento de vídeo en tiempo real. Establece un soporte que sincroniza una cámara digital y un monitor VGA, permitiendo incorporar una unidad de procesado de información de vídeo.

1.2 FASES DEL PROYECTO

- En primer lugar fue necesaria una etapa de búsqueda de documentación y de consulta de la bibliografía relacionada.
- Posteriormente, se eligieron la cámara digital, el monitor y el modelo de FPGA que satisficieran las necesidades técnicas del sistema a implementar.
- Después, se realizaron las modificaciones necesarias en la placa de la cámara digital para establecer la configuración deseada.
- A continuación, se programaron en VHDL todos los módulos necesarios para implementar la arquitectura del proyecto.
- Finalmente, se realizó la fase de redacción del proyecto

1.3 HERRAMIENTAS UTILIZADAS

1.3.1 Hardware

- Ordenador de sobremesa. Procesador Intel Pentium 4 a 3 GHz, 1GB de memoria RAM, pantalla plana de 17".
- Placa de evaluación y desarrollo Xilinx Spartan-3 XC3S1500-FG456
- Cámara C3188A con sensor de imagen OmniVision OV7620
- Monitor TFT de 15'' Dell - E151FPp
- Cable de descarga JTAG Parallel IV
- Herramientas de análisis: osciloscopio y generador de funciones.

1.3.2 Software

- ISE 8.2i Service Pack

1.4 CONTENIDO

En el capítulo 2 se describen los componentes utilizados en el sistema. Comenzaremos introduciendo los sensores de imagen y caracterizando la cámara digital utilizada. A continuación, estudiaremos el funcionamiento de un monitor empleado y, por último, explicaremos detalladamente el entorno de trabajo ISE 8.2i y la FPGA en la que se implementa el proyecto.

En el capítulo 3 se realiza un análisis detallado de todos los módulos que componen la arquitectura del proyecto. Comenzaremos con una visión global de la arquitectura para profundizar posteriormente en cada módulo. Al final del capítulo describimos una simulación de la arquitectura con ModelSim y comentaremos las características de velocidad y consumo de área al volcar la arquitectura en la FPGA.

En el capítulo 4 se presentan los resultados del proyecto y el montaje del prototipo.

En el capítulo 5 se recogen las conclusiones y futuras líneas de trabajo derivadas de este proyecto.

Por último, se presentan al final del documento las referencias bibliográficas.

1.5 SISTEMAS DE VÍDEO EN TIEMPO REAL.

La computación en tiempo real es una tecnología disponible para muchas áreas importantes. Una lista de esas áreas incluye: control de procesos, alimentación de plantas nucleares, sistemas para vehículos inteligentes, aviación, telecomunicaciones, multimedia, simulación en tiempo real, realidad virtual, aplicaciones médicas y aplicaciones de defensa. En particular, casi todos los sistemas críticos de seguridad y muchos sistemas embebidos son sistemas de tiempo real. La tecnología en tiempo real está comenzando a crecer de manera importante y perseverante, haciendo que cada vez más y más infraestructuras del mundo dependan de ella.

Las líneas de investigación de la computación en tiempo real implican hacer frete a los nuevos tipos de sistemas de tiempo real incluyendo, sistemas abiertos de tiempo real, sistemas de tiempo real globalmente distribuidos, y sistemas multimedia. Para cada uno de estos, se requiere investigar su evolución, ingeniería de software, rendimiento, fiabilidad, lenguajes de programación, etc. Las soluciones deberán tener en cuenta los aspectos económicos, de seguridad y las propias restricciones temporales.

1.5.1 Ejemplos de sistemas de tiempo real

Un sistema en tiempo real es aquel en el que la exactitud del sistema depende no sólo de los resultados lógicos, sino también del momento en que se producen esos resultados. Muchos de los sistemas de tiempo real son sistemas integrados, es decir, que forman parte de un sistema mayor. Un mal funcionamiento del sistema puede producir un problema crítico para el propio sistema. Como ejemplo, tenemos el sistema Air Trac Control. Este sistema debe gestionar grandes cantidades de datos. A diferencia de algunos grandes sistemas de gestión de datos, tales como las reservas de líneas aéreas, los datos del Air Trac Control están cambiando constantemente, y tienen valor muy alto (en relación con la seguridad) para espacios muy cortos de tiempo (los requisitos para el tiempo de respuesta van desde unos pocos milisegundos para los datos del radar a varios segundos en el control información). En conclusión, el nuevo sistema Air Trac Control de EE.UU. se estima que costará más de mil millones de dólares. Sin embargo, el sistema es tan grande y complejo (el sistema tendrá entre 1 y 2 millones de líneas de código y miles de consolas) que las nuevas investigaciones en tiempo real son necesarias para mejorar aún más la seguridad, reducir el coste del sistema y su mantenimiento, y prepararlo para su continua evolución en tamaño y complejidad.

El tiempo real y la computación embebida también tienen un rol importante en el sector industrial. Consideramos una fábrica de automóviles. Las fábricas de automóviles deben incorporar sistemas de computación de tiempo real en sus coches para ser competitivos. En el futuro, los sistemas distribuidos de tiempo real harán que los coches sean más eficientes y seguros. Antes de incorporar estos sistemas deben abordarse varias líneas de investigación: obtener la respuesta del sistema en un microsegundo, temporización, mantenimiento y coste de su incorporación al mercado. La investigación de la computación en tiempo real está siendo muy efectiva. Por ejemplo, los avances en la ciencia de la temporización de la calidad del servicio (QoS) han permitido crear algoritmos de validación sólidos y robustos para aplicaciones en tiempo real tales como el control digital y la tasa de bit de vídeo y audio. Otro ejemplo de aplicación de estos

sistemas es el software integrado en los satélites del NAVSTAR Sistema de Posicionamiento Global (GPS). Se debe garantizar el cumplimiento de los tiempos de las tareas del software, puesto que muchas aplicaciones terrestres dependen del GPS, y si la información del GPS es errónea, puede tener graves consecuencias para estas aplicaciones.

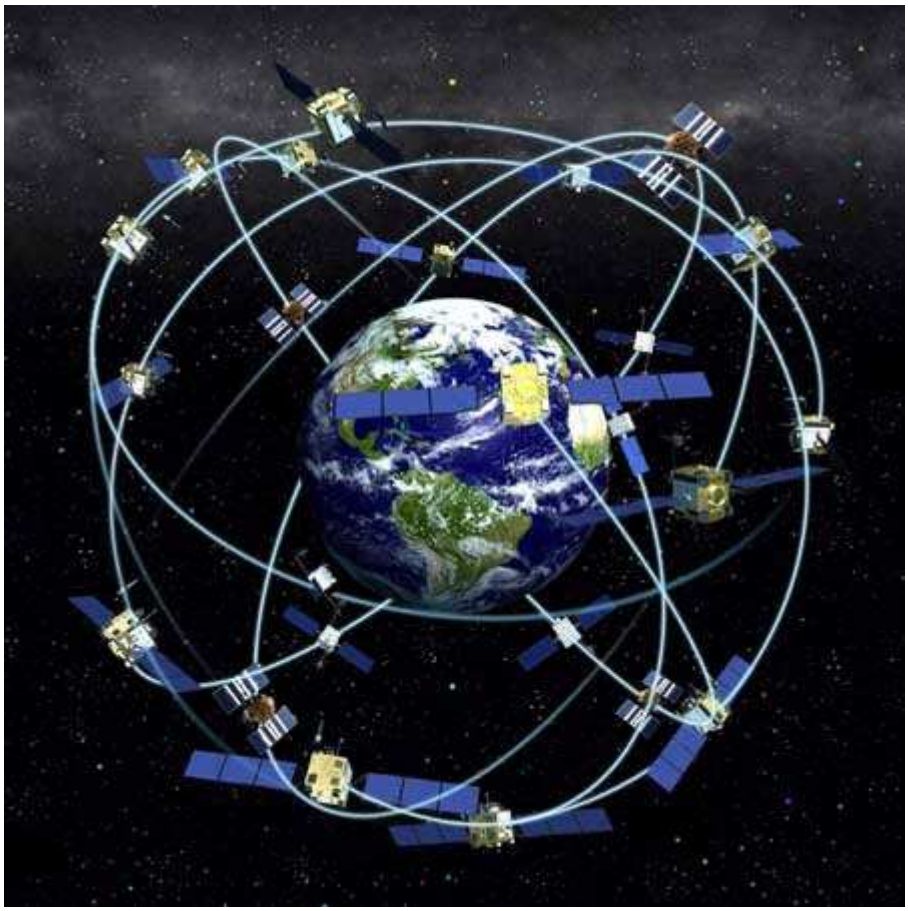


Figura 1. Configuración satélites para GPS

Otro ejemplo del éxito de la tecnología en tiempo real está en la industria del software. Esta industria ha seguido de cerca el progreso de la industria del semiconductor, en general, y de la industria del microprocesador, en particular. Actualmente, se estima que se gastan anualmente más de 2 mil millones de dólares en herramientas, aplicaciones software y sistemas operativos embebidos. Este mercado también está creciendo aproximadamente un 25 % por año. Los sistemas operativos en tiempo real se utilizan en una gran variedad de sistemas embebidos como la aeronáutica, medicina, comunicación, consumo y aplicaciones instrumentales.

También se esperan éxitos en otras áreas. Por ejemplo, el comercio en tiempo real en Internet está transformando el mundo de los negocios. Habrá pequeños procesadores integrados en millones de productos, lo que los hará más inteligentes. Básicamente, el potencial de la de la tecnología en tiempo real es ilimitado.

CAPÍTULO 2. Componentes del sistema

2.1 CÁMARA

2.1.1 Cámara convencional

2.1.1.1 Sensores de imagen

Actualmente, en el mercado fotográfico han aparecido diversas cámaras profesionales que incorporan como sensor de imágenes dispositivos de tecnología CMOS, tecnología que ya podíamos encontrar en algunas cámaras de fotografía digital y webcams, cuya calidad dejaba mucho que desear. No obstante ya existen cámaras que emplean un sensor de imagen CMOS de alto rendimiento, demostrando que los sensores de este tipo pueden producir una calidad de imagen excepcional. Por esto consideramos interesante la explicación un poco más detallada de las diferencias, ventajas e inconvenientes de las dos tecnologías que existen de sensores de imagen: el CCD y el CMOS.

Figura 2. Cámara fotográfica digital

El sensor de imagen es un dispositivo que percibe las variaciones de intensidad de la luz, pero sin distinguir los colores de la imagen; y para que el sensor pueda captarlos, se deben emplear filtros que dividan los colores de la escena en Rojo, Verde y Azul. Esto deriva en una limitación física a la resolución porque con cada celda de la matriz sólo podremos capturar la luz de un solo color y, en consecuencia, en un archivo resultante muy pequeño, generándose además falta de información de color en algunos puntos. Un problema que se soluciona calculando estos puntos mediante técnicas

matemáticas de interpolación, en las que el software de la cámara determina el color posible de una celda sobre la base de los colores de las celdas adyacentes.

Figura 3. Sistema de captación de colores de los sensores de imagen.

Los sensores de un solo disparo pueden estar basados en dos tipos de tecnologías, CCD (Charged Couple Device) o CMOS (Complementary Metal Oxide Semiconductor). Tradicionalmente se utilizaban los CCD para las cámaras profesionales y semi-profesionales y los CMOS para las cámaras de aficionados y las webcam. Pero esta distribución ha cambiado y los fabricantes ya incorporan en sus modelos la tecnología CMOS.

2.1.1.2 Sensor CCD (Charge Coupled Device)

Este tipo de sensor debe su nombre a una tecnología que en sus principios se empleó en la fabricación de memorias de acceso secuencial. Los dispositivos de carga acoplada (CCD) no son más que dispositivos electrónicos de silicio que en cada uno de sus puntos fotosensibles incorpora un fotodiodo, cuya propiedad es generar electrones dependiendo de la cantidad de luz que incide sobre él. Al captar la imagen estos diferentes fotodiodos producen una carga electrónica proporcional a la luz incidente y es, en la lectura de estas cargas electrónicas, donde encontramos la mayor diferencia entre las dos tecnologías aquí analizadas.

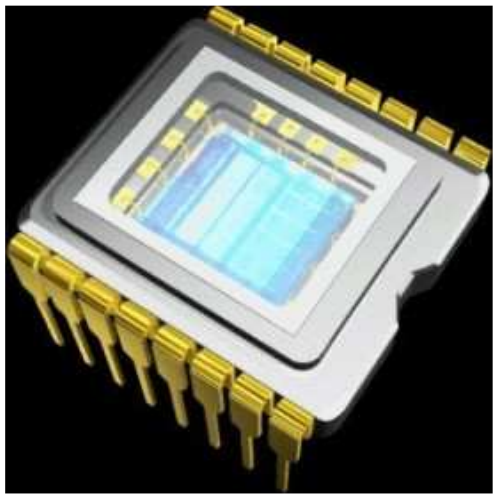


Figura 4. Sensor CCD

En el CCD, mediante una señal de reloj procedente del circuito integrado de la cámara, la carga que posee uno de estos fotodiodos va pasando de éste al adyacente y así sucesivamente hasta llegar a un registro (también formado por dispositivos de carga acoplada) que es el encargado de ir suministrando, por orden secuencial, las diferentes cargas que poseen los distintos fotodiodos que forman el sensor. Estas cargas electrónicas se convierten en potencial eléctrico (voltaje), que se amplifica y se recoge en el circuito integrado de la cámara, encargado de procesar estos datos y proporcionar una señal digital que se graba en memoria.

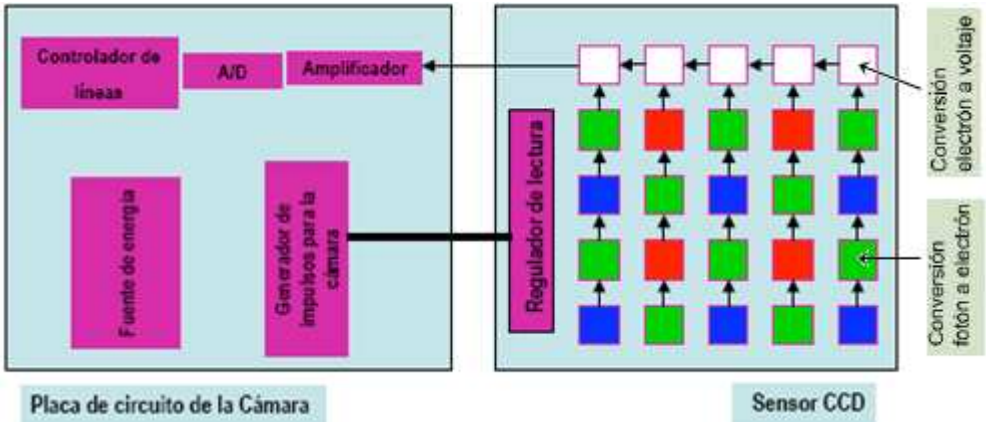


Figura 5. Estructura de una cámara CCD

2.1.1.3 Sensor CMOS (Complementary Metal Oxide Semiconductor)

Este dispositivo recibe su nombre del método de fabricación utilizado en su elaboración. CMOS (Complementary Metal Oxide Semiconductor) no es más que una forma de fabricación de circuitos integrados que se ha convertido en la más generalizada en la fabricación de microchips, relegando la tecnología TTL con la que se fabricaban los primeros chips. Esta es su gran ventaja, ya que su fabricación es posible en cualquier planta de fabricación de memorias, microprocesadores y demás controladores sin apenas realizar cambios en la cadena de montaje, lo que repercute en un menor coste.

Figura 6. Sensor CMOS

Estos dispositivos se caracterizan ante todo porque cada fotodiodo integrado en el sensor lleva consigo la electrónica necesaria para convertir la carga de electrones generada en voltaje, así como un registro individual de este voltaje. Esto supone que la superficie necesaria para captar la luz, a un mismo tamaño de celda, es menor que en un CCD, pero tiene la gran ventaja de poder acceder a la información captada no solo en la totalidad del dispositivo sino también a una zona particular de éste. El chip sensor CMOS no sólo integra los fotodiodos sino que también integra toda la electrónica necesaria para el control y lectura de estos, así como el conversor analógico-digital, lo que se traduce en un menor tamaño de los circuitos necesarios para la captura de imágenes.

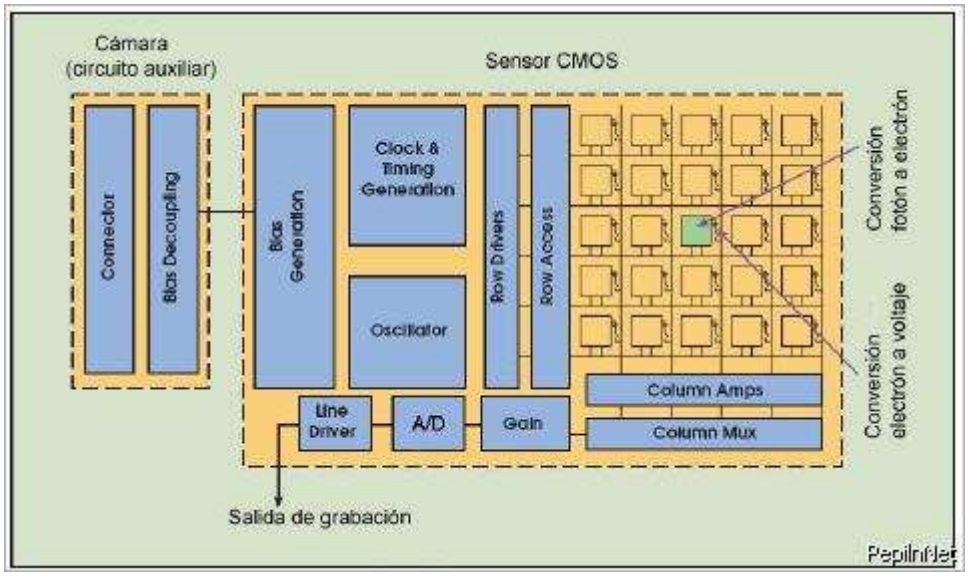


Figura 7.Estructura de una cámara CMOS

2.1.1.4 Comparativa: CCD vs CMOS

Como hemos visto, la gran diferencia entre ambos dispositivos es su construcción. Ambos se basan en el silicio para ello, pero mientras en el CCD la carga electrónica va pasando de forma secuencial hasta el dispositivo que la convierte en voltaje, en el CMOS esta conversión se realiza en el mismo fotodiodo. No obstante, esta forma de actuar es la que marca las diferencias entre uno y otro.

Electrónica de control

La primera diferencia es que, mientras en el CMOS la electrónica de control se encuentra integrada en el dispositivo de captura, en el CCD está fuera, lo que hace que sea más fácil la actualización de las cámaras basadas en este dispositivo, ya que si se demanda alguna mejora sin la necesidad de utilizar otro sensor, como sería el caso del CMOS, variando la electrónica de control podemos cambiar el resultado de la imagen sin tener que cambiar de sensor.

Sensibilidad

Nos referimos a la capacidad que tiene el sensor de generar carga eléctrica por unidad de luz que incide sobre él. En el CMOS, al amplificarse directamente la señal que incide en el fotodiodo, esta respuesta es mejor, aunque actualmente algunos fabricantes de CCD están cambiando este concepto mediante la aplicación de nuevas técnicas de amplificación de la señal.

Responsividad

Se define con este término al nivel de señal que es capaz de ofrecer el sensor por cada unidad de energía óptica incidente. Dado que nos interesa que el sensor tenga una responsividad elevada, se necesita que con poca cantidad de luz el sensor nos proporcione una señal aceptable. Con esto en mente, podemos ver que los sensores CMOS son superiores a los de tecnología CCD, debido a que integra elementos amplificadores en cada celda. Además el sistema de construcción CMOS permite una alta amplificación con un bajo consumo, mientras que en CCD la amplificación al ser externa al sensor supone un consumo más elevado.

Rango Dinámico

Nos indica el nivel de señal que es posible medir entre el umbral del fotodiodo y su saturación, lo que va a influir en la gama de luminosidad que podamos obtener del sensor. En este aspecto, el CCD supera en casi el doble al CMOS, ya que, como hemos visto en su construcción, al mismo tamaño de sensor, la superficie responsable de captar la luz es mayor.

Velocidad

Si nos concentramos en la velocidad con la que se captura la imagen, veremos que los CMOS son bastante superiores a los CCD, debido a que muchas funciones, como la propia conversión analógico-digital son realizadas en el propio sensor. Si bien por el momento esta ventaja es ligera, se espera que aumente con el tiempo.

Consumo eléctrico

En referencia al consumo, el consumo energético necesario es mucho menor en un CMOS que en un CCD. Los CMOS están altamente optimizados, con un consumo entre 30 y 50 mW, en tanto que un CCD consume entre 2 y 5 Wattios.

Ruido

En este punto también sale favorecido el CCD ya que, al integrar menos electrónica en el sensor, el ruido electrónico también es menor.

Figura 8. Ruido fotónico

Blooming

Este efecto, por el cual si un fotodiodo de un CCD se satura demasiado puede afectar a otros fotodiodos próximos a él, no se produce en el sensor CMOS ya que en él no hay transferencia de carga entre los diferentes fotodiodos.

Figura 9. Efecto "blooming"

En conclusión, los sensores CCD ofrecen mejor calidad de imagen y mayor flexibilidad que los sensores CMOS, a cambio estos consumen mucha menos energía y permiten un tamaño de integración más pequeño. De todas formas, ambas tecnologías están evolucionando hacia mayores niveles de calidad, por lo que en un futuro próximo veremos cámaras que incorporan CMOS con un nivel de calidad equiparable a los mejores CCD y, a estos niveles de calidad, sin apenas diferencia en coste entre un dispositivo y otro.

2.1.2 Cámara digital C3188A

La cámara digital OmniVision se emplea en el proyecto como el componente que captura y proporciona la información. Después esta información enviada a una FPGA, donde se sincronizará para poder ser representada en un monitor VGA.

En concreto, se trata del módulo de cámara a color C3188A de 1/3''. Incorpora un sensor de imagen CMOS OV7620 de la marca OmniVision. La combinación de la tecnología CMOS con una interfaz digital sencilla hace al módulo C3188A una solución de bajo coste para aplicaciones de vídeo de alta calidad.

Figura 10. Cámara C3188A sin óptica (izquierda) y con óptica (derecha)

Entre sus características destacamos que tiene un tamaño pequeño (40 x 28 mm), bajo consumo (menos de 120 mW), se alimenta con 5V DC, realiza una mejora en la imagen actuando sobre el brillo, contraste, saturación, nitidez, etc. Además, incorpora un sistema anti blooming y permite seleccionar el modo de escaneo entre progresivo o entrelazado.

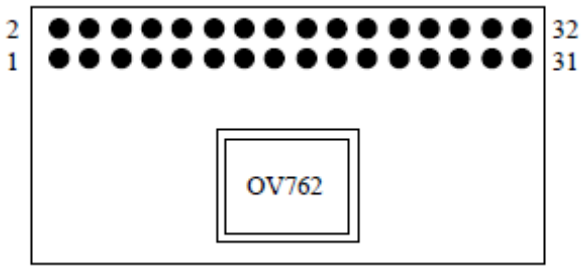


Figura 11. Pines cámara C3188A

La Tabla 1 muestra la definición de sus pines:

Pines		
Número	**Nombre**	**Descripción**
Y0 -Y7	*1 – 8*	Bus de salida digital Y
PWDN	*9*	Modo **de bajo consumo**
RST	*10*	Reset de la cámara
SDA	*11*	Datos de I^2C
FODD	*12*	**Indicador de campos impares.**
SCL	*13*	Reloj de entrada de I^2C
HREF	*14*	Sincronismo horizontal de salida
AGND	*15*	Masa analógica
VSYN	*16*	Sincronismo vertical de salida
AGND	*17*	Masa analógica
PCLK	*18*	Salida Pixel clock (reloj de salida)

EXCLK	*19*	Reloj externo de entrada
VCC	*20*	Alimentación a 5V DC
AGND	*21*	Masa analógica
VCC	*22*	Alimentación a 5V DC
UV0 – UV7	*23-30*	Bus de salida digital UV
GND	*31*	Masa común
VTO	*32*	Salida analógica de vídeo

Tabla 1. Descripción de los pines de la cámara C3188A

2.1.2.1 Características sensor

Como ya se ha comentado en el apartado anterior el sensor que utiliza la cámara es el modelo OV7620 de la marca OmniVision. Se trata de un chip de alta densidad de integración para una cámara de vídeo de alta resolución (640x480). Soporta distintos modos de escaneo (progresivo y entrelazado) y diferentes formatos de vídeo (YUV y RGB).

La siguiente tabla presenta sus especificaciones:

Especificaciones	
Imagen	Sensor de imagen CMOS OV7620
Tamaño array	664 x 492 píxeles
Tamaño píxel	7.6 x 7.6 µm
Escaneo	Progresivo/Entrelazado
Área efectiva de la imagen	4.86 mm x 3.64 mm
Exposición electrónica	500:1
Corrección Gamma	Colección de curvas 128
Relación S/N	>48 dB
Mínima Iluminación	2.5 lux @F1.4
Voltage de operación	5 V DC
Corriente de operación	129 mW Activa 10µW Standby
Lente	f6 mm, F1.6

Tabla 2. Especificaciones del sensor OV7620

A continuación, exponemos el diagrama de pines del chip y la descripción de los más significativos para nuestro proyecto.

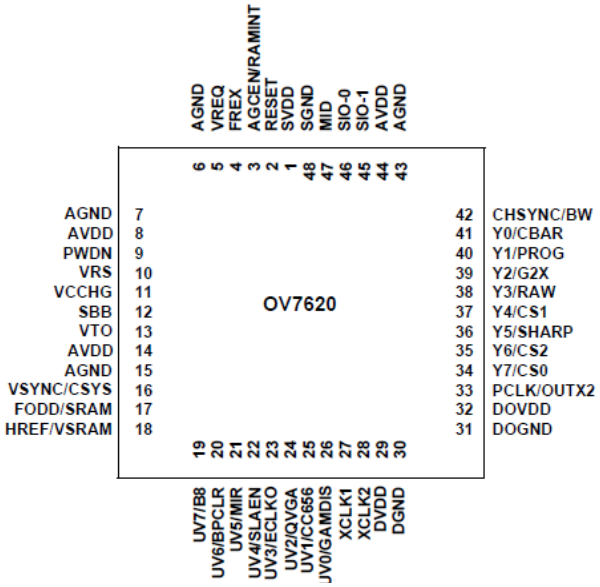


Figura 12. Diagrama de pines del chip OV7620

Pines			
Número	**Nombre**	**Clase**	**Descripción**
8,14,44	AVDD	Alimentación	Alimentación pines analógicos (+5V)
29	DVDD	Alimentación	Alimentación pines digitales (+5V)
32	DOVDD	Alimentación	Alimentación entrada/salida pines digitales (+5V)
6, 7, 15, 43	AGND	Masa	Masa conexiones analógicas
30	DGND	Masa	Masa conexiones digitales
31	DOGND	Masa	Masa salidas digitales
2	RESET	Entrada	Reset del chip, activo en “alta”
12	SSB	Entrada	SSB = 1 selecciona el método de programación de alimentación de funciones internas. SBB = 0 selecciona el método de programación SCCB.
16	VSYNC	Salida	VSYNC: señal de sincronismo vertical. Se pone en alta durante varias líneas.
17	FODD	Salida	FODD: Indicador de campo impar. En modo entrelazado se pone en nivel alto durante los campos impares y nivel bajo en los impares. En modo progresivo siempre en nivel bajo
18	HREF	Salida	HREF: señal de sincronismo horizontal. Se pone en nivel alto en los píxeles visibles de la ventana y en nivel bajo en los no visibles.
19 - 26	UV7-UV0	Salida	Bus de salida digital UV. Utilizado en las operaciones de 16 bits como salida de datos de crominancia.
33	PCLK	Salida	Señal de reloj a 13.5 Mhz.
34 - 39	Y7 – Y2	Salida	6 bits más significativos del bus de salida digital Y. Utilizado en las operaciones de 16 bits como salida de datos de luminancia a la velocidad de 1 byte por pixel.
40	Y1/PROG	Salida/Entrada	Y1: 2º bit menos ignificativos del bus de salida digital Y. Utilizado en las operaciones de 16 bits como salida de datos de luminancia a la velocidad de 1 byte por pixel. PROG: Modo de escaneo Progresivo.
41	Y0	Salida	Y0: bit menos significativos del bus de salida digital Y. Utilizado en las operaciones de 16 bits como salida de datos de luminancia a la velocidad de 1 byte por pixel.

Tabla 3. Descripción de los pines del chip OV7620 utilizados en nuestro proyecto

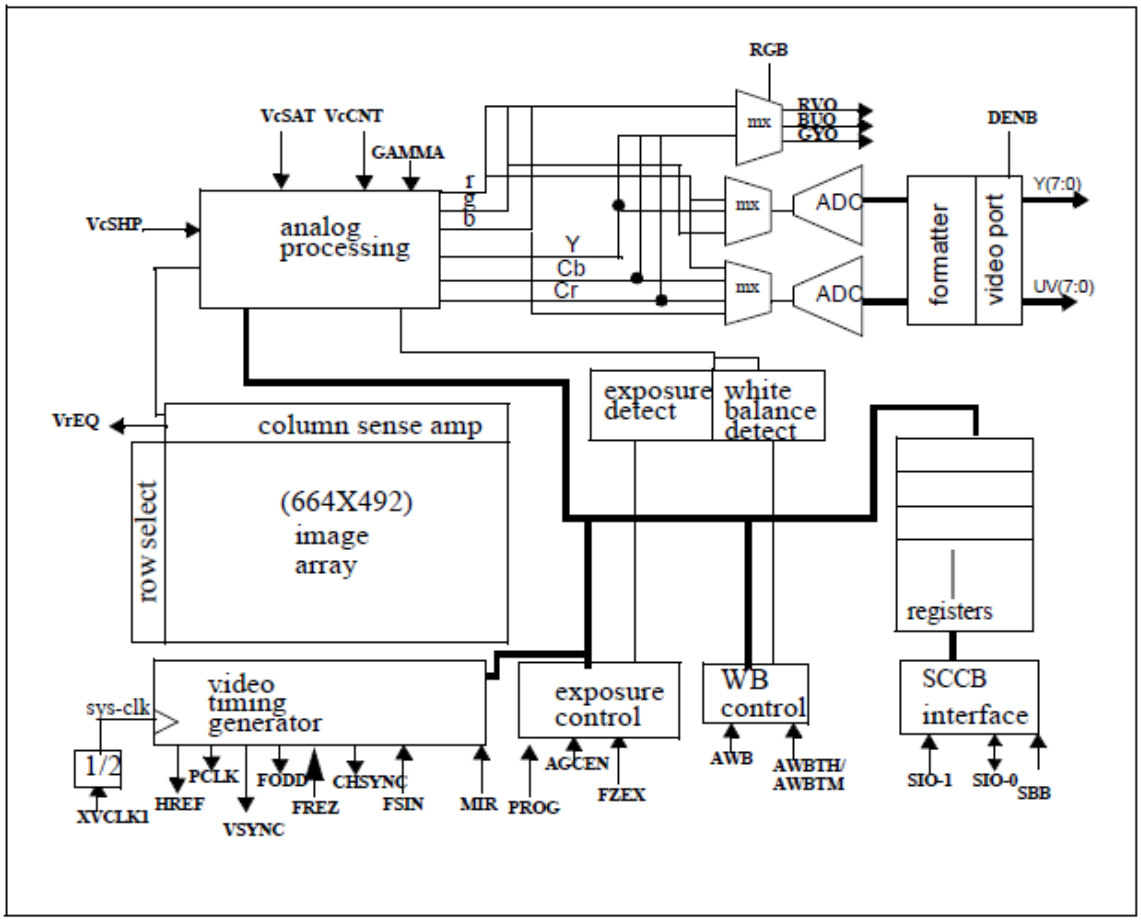


Figura 13. Diagrama de bloques del chip OV7620

Estándar YUV

El chip será configurado para trabajar con el estándar YUV (también llamado CCIR 601), conocido anteriormente como YCrCb (Y Cr Cb), que es un modelo de representación del color dedicado al vídeo análogo.

Se basa en un modo de transmisión de vídeo con componentes separados que utiliza tres cables diferentes para llevar información con respecto a los componentes de luminancia (luminosidad) y los dos componentes de crominancia (color). Es el formato utilizado en las normas PAL (Phase Alternation Line) y SECAM (Séquentiel Couleur À Mémoire).

El parámetro Y representa la luminancia (es decir, información en blanco y negro), mientras que U y V representan la crominancia (es decir, información con respecto al color). Este modelo se desarrolló para permitir la transmisión de información a color en televisores a color y a la vez garantizar que los televisores blanco y negro existentes continuaran mostrando una imagen en tonos de gris.

A continuación se muestran las relaciones entre Y y R, entre G y B, entre U, R y luminancia, y finalmente entre V, B y luminancia:

$$Y = 0.299R + 0.587\,G + 0.114\,B$$
$$U = -0.147R - 0.289\,G + 0.436B = 0.492(B - Y)$$
$$V = 0.615R - 0.515G - 0.100B = 0.877(R-Y)$$

Por lo tanto, U a veces se escribe como Cr y V a veces se escribe como Cb, de ahí la notación YCrCb.

Formato de vídeo

El formato de vídeo empleado es el 4:4:2 de 16 bits. Esto significa que cada pixel quedará codificado con 16 bits de información (8 de crominancia UV y 8 de luminancia Y). A continuación se definen su secuencia de salida de datos y su temporización:

Bus de datos	**Secuencia de salida**					
Y7	Y7	Y7	Y7	Y7	Y7	Y7
Y6	Y6	Y6	Y6	Y6	Y6	Y6
Y5	Y5	Y5	Y5	Y5	Y5	Y5
Y4	Y4	Y4	Y4	Y4	Y4	Y4
Y3	Y3	Y3	Y3	Y3	Y3	Y3
Y2	Y2	Y2	Y2	Y2	Y2	Y2
Y1	Y1	Y1	Y1	Y1	Y1	Y1
Y0	Y0	Y0	Y0	Y0	Y0	Y0
UV7	U7	V7	U7	V7	U7	V7
UV6	U6	V6	U6	V6	U6	V6
UV5	U5	V5	U5	V5	U5	V5
UV4	U4	V4	U4	V4	U4	V4
UV3	U3	V3	U3	V3	U3	V3
UV2	U2	V2	U2	V2	U2	V2
UV1	U1	V1	U1	V1	U1	V1
UV0	U0	V0	U0	V0	U0	V0
YFRAME	0	1	2	3	4	5
UV FRAME	0		2		4	

Tabla 4. Formato 4:2:2 de 16 bits

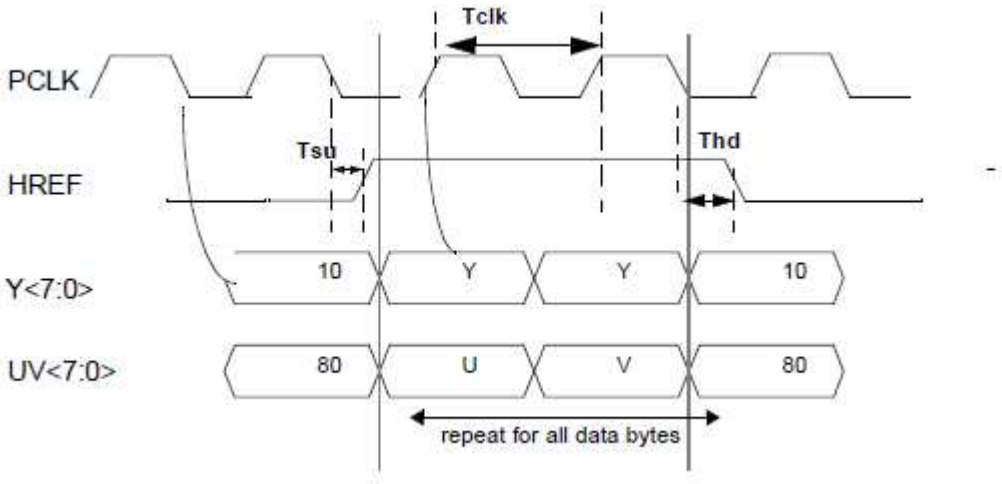


Figura 14. Salida YUV de la cámara

Notas: TCLK = 74 ns (señal de reloj a 13.5 Mhz). Tsu = tiempo de subida de Href (máximo 15 ns). Thd = tiempo de bajada de Href (máximo 15 ns).

Modos de escaneo

Esta cámara permite seleccionar como modos de escaneo entre los modos entrelazado y progresivo.

El formato entrelazado presenta la imagen de vídeo como dos campos separados que barren las líneas de la pantalla de forma alternativa, uno las líneas pares y el otro las líneas impares, para mostrar la imagen completa. Es la forma en que funcionan las televisiones convencionales.

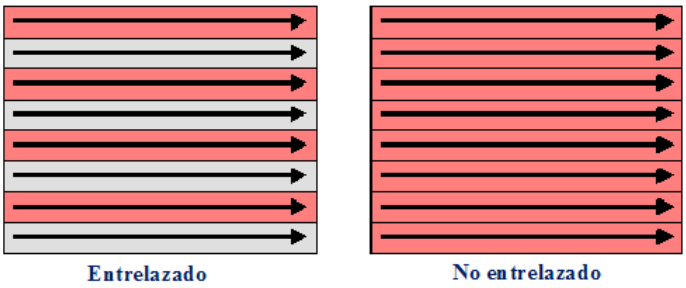


Figura 15. Modos de escaneo

El formato progresivo muestra cada imagen de vídeo en un solo fotograma. Es decir, todas las líneas de la imagen se muestran a la vez, lo que coincide con el formato cinematográfico.

En el chip la configuración por defecto trabaja con el modo entrelazado, por lo que necesitaremos introducir cambios para hacerlo trabajar en progresivo.

Salida de información

La cámara será configurada (ver apartado 2.1.2.2) para trabajar en modo progresivo, con formato de vídeo 4:2:2 de 16 bits y con una frecuencia de frames de 30 Hz. Por tanto, cada 33.33 ms (1÷30) la cámara capturará una imagen (frame), cuya información en formato YUV saldrá siguiendo unos determinados sincronismos.

En primer lugar, es necesario saber que la ventana de una imagen capturada por la cámara tiene 525 líneas, de las cuales 480 líneas serán visibles y 45 no visibles. Cada línea contiene 858 píxeles, siendo visibles sólo 640 de ellos y a cada píxel se corresponde a 1 periodo de la señal de reloj de 13.5 MHz. La información visible es proporcionada en los buses de salida (YUV) del modo que muestra la figura 16, de izquierda a derecha y de arriba abajo.

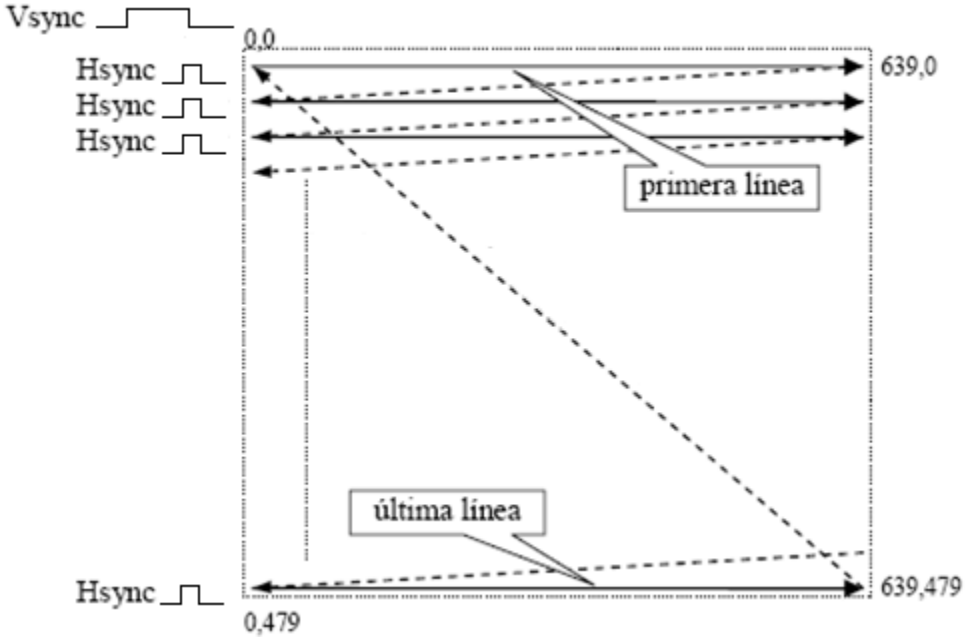


Figura 16. Lectura de la ventana de la imagen capturada por la cámara

La señal de sincronismo vertical *VSYNC* indica con su pulso positivo el comienzo de un nuevo frame (imagen capturada), mientras que la señal de sincronismo horizontal *HREF* indica cuando está en alta que los buses de salida están proporcionando información válida (640 datos válidos por línea). La siguiente figura muestra la salida de información controlada por las señales de sincronismo, así como su temporización:

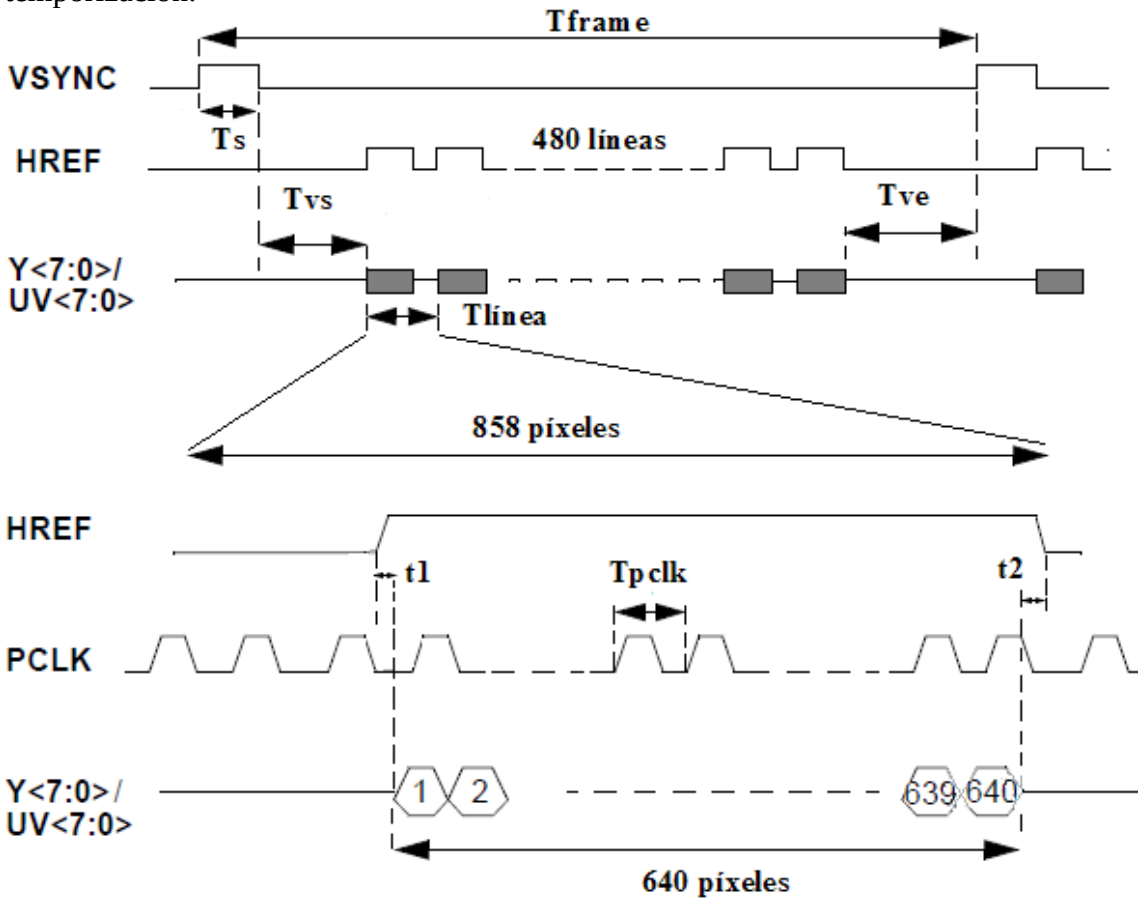


Figura 17. Salida de datos de la cámara

Símbolo	Parámetro	Duración	
		Tiempo	*Ciclos de reloj*
Tframe	Duración de un frame (un sincronismo vertical)	33.6 [ms]	525 x 858
Ts	Pulso sincronismo vertical	190.7 [µs]	3 x 858
Tvs	Portal trasero	2.36 [ms]	37 x 858
Tve	Portal frontal	319.5 [µs]	5 x 858
Tlínea	Duración de una línea	63.55 [µs]	858
Tpclk	Periodo de la señal de reloj (13.5 Mhz)	74.074 [ns]	1
t1	Tiempo entre subida Href e información en buses válida	10 [ns]	
t2	Tiempo información en buses válida del último píxel y bajada Href	20 [ns]	

Tabla 5. Tiempos de sincronismo para la salida de datos de la cámara 3188A funcionando a una frecuencia de refresco de 30 Hz

2.1.2.2 Configuración para el proyecto

Para poder trabajar con la cámara C3188A es necesario hacer algunas modificaciones respecto a la configuración por defecto. En concreto, se necesitan hacer cambios en el modo de escaneo y en la salida digital.

1) Modo de escaneo

Por defecto, el chip selecciona automáticamente como modo de escaneo de imágenes el modo entrelazado. Hacerla funcionar en modo progresivo requiere los siguientes cambios:

- Soldar una resistencia de 10 KΩ (R2 en esquema) en el pin 12 del sensor (SBB) para ponerlo en alta impedancia.
- Soldar una resistencia de 10 KΩ en el pin 40 (PROG). Así cuando la cámara se enciende detecta este nivel de tensión y pasa a funcionar en modo progresivo.

2) Salida digital

La configuración por defecto de la cámara hace que proporcione los datos de salida con un nivel de tensión de 5V. Como los buses de datos de salida actuarán como entradas de la FPGA, es necesario que los niveles de tensión no superen los 3.3 V (tecnología 3.3 V). Para ello es necesario poner en el pin 32 del chip (entrada DOVDD) un nivel de tensión de 3.3 V. Lo haremos de la siguiente manera:

- Desconectamos el pin DOVDD de la alimentación
- Soldamos un espadín en la placa que contacte con este pin.
- Conectamos el espadín a un nivel de tensión de 3.3 V

Además, la arquitectura de nuestro proyecto está diseñada para trabajar en blanco y negro, es decir, que los datos de entrada a la FPGA contengan la información en escala de grises. Para conseguir esto no se necesita hacer ninguna modificación, ya que el formato de los datos es el YUV y nos basta con trabajar únicamente con el bus de luminancia Y<7:0>.

Nota: para comprender con claridad los cambios realizados en la placa se sugiere ver el esquema de conexiones dispuesto en el “Anexo A” de este mismo apartado

2.2 MONITOR VGA

El objetivo de este apartado es explicar las señales VGA que controlan la información que se presenta en la pantalla de nuestro monitor. El dispositivo empleado es un TFT de 15'' de la marca Dell, en concreto el modelo E151FPp que muestra la Figura 18.

Figura 18. Monitor TFT 15'' Dell E151FPp

2.2.1 Funcionamiento

En un monitor TFT los píxeles son generados por transistores, de manera que cada píxel es representado como si de una bombilla minúscula se tratase, la cual emite una luz que pasa por diferentes filtros, que determinan que color va a tomar el píxel en cada momento (Figura 19). Cambiando la orientación de los cristales podemos dejar pasar más o menos luz. Los filtros proporcionan los tres colores básicos, pudiendo así formar cualquier imagen.

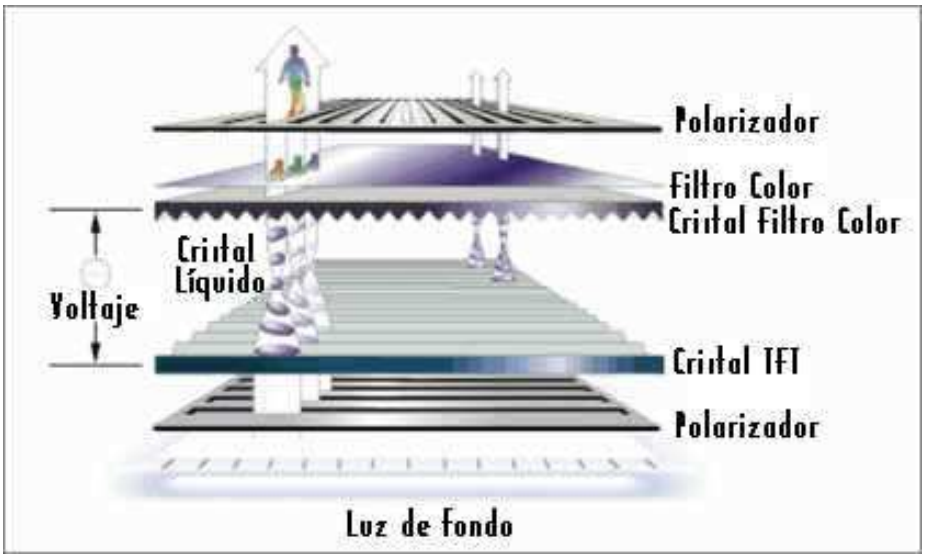


Figura 19. Funcionamiento pantallas TFT

El cuadro de vídeo VGA tiene 525 líneas, de las cuales 480 líneas serán visibles y 45 no visibles. Cada línea contiene 800 píxeles, siendo visibles sólo 640 de ellos y a cada píxel se corresponde a 1 periodo de la señal de reloj de 25.175 MHz.

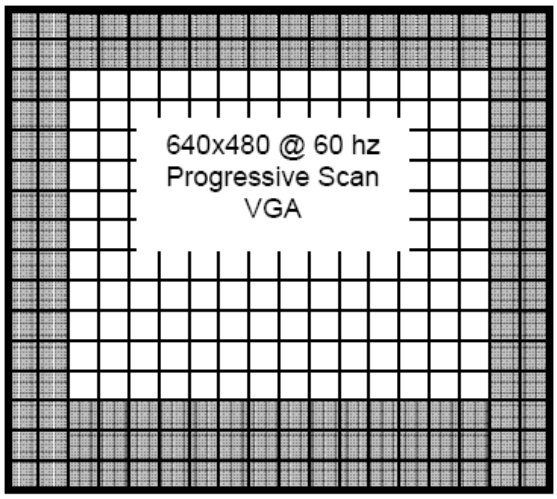


Figura 20. Cuadro VGA 800x525 píxeles. En color blanco Información visible (640x480 píxeles).

Con el fin de colocar una imagen en la pantalla, se requieren dos señales de sincronización, una para cada eje, de manera que se presente la información en la zona visible de la pantalla del monitor. Con estas señales se controla la iluminación de los píxeles visibles de izquierda a derecha y de arriba debajo de la pantalla. La figura 21 muestra un ejemplo de visualización temporal de una imagen VGA. Las líneas continuas representa la información desplegada en pantalla de acuerdo al contenido de la señal RGB; cada línea está precedida por un pulso de sincronismo horizontal, y cada actualización de la imagen completa (**frame**) está precedida por un pulso de sincronismo vertical.

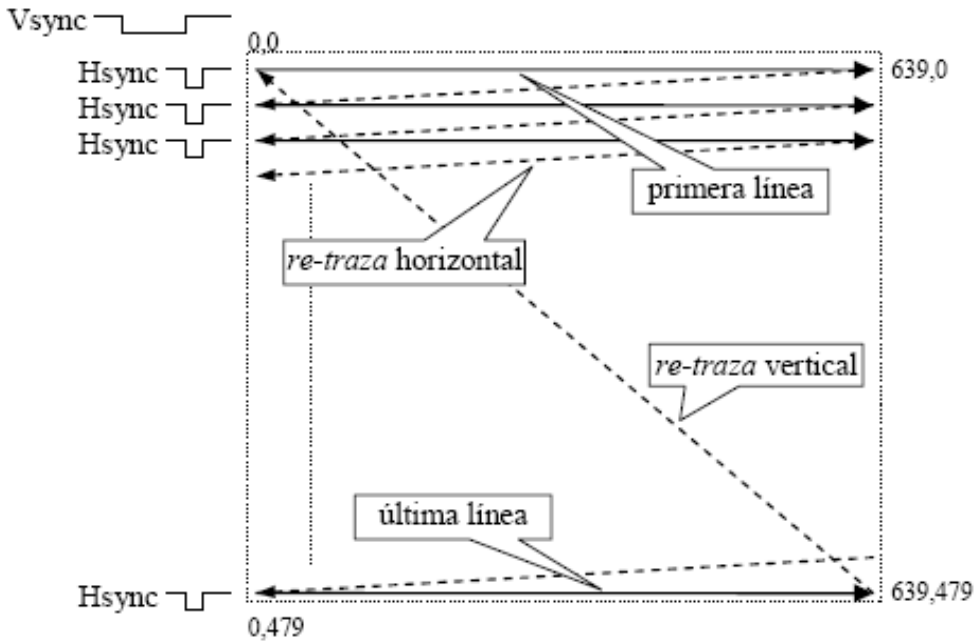


Figura 21. Visualización de imagen VGA de 640x480 píxeles

Los pulsos negativos en la señal de sincronismo horizontal marcan el comienzo y el final de una línea y aseguran que el monitor despliegue los pixeles entre los bordes, izquierdo y derecho, del área visible de la pantalla.

La figura 21 muestra las señales de sincronismo y de color para un monitor VGA. Tanto la señal de sincronismo horizontal como vertical tienen la misma forma, y sólo cambia la duración de los distintos segmentos. Además, la señal de sincronismo horizontal sólo se generará en las líneas visibles, tomando un valor alto (sin pulsos negativos) en las líneas no visibles. En cuanto a la señal RGB, lleva la información de colores para cada píxel. En cada línea horizontal la señal RGB tiene una zona denominada *blanking*, correspondiente al tiempo de retrazado horizontal, donde dicha señal debe ser cero. En la Tabla 6 se muestran los tiempos de las señales de sincronismo para un número de píxeles totales de 800x525 en una pantalla de 640x480 y a una frecuencia de refresco de 60 Hz (sincronismo vertical).

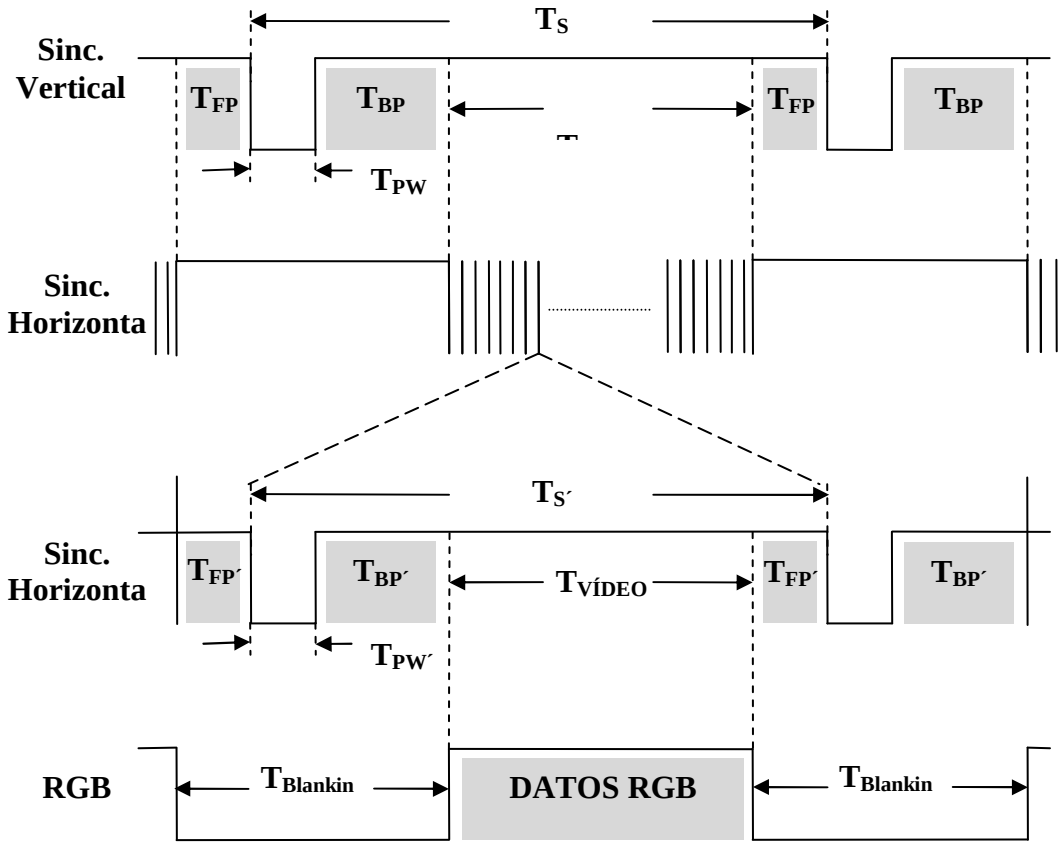


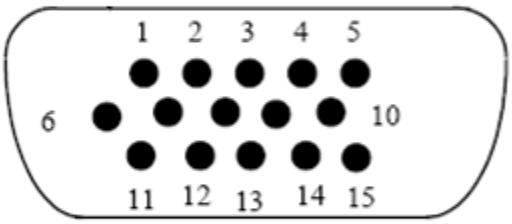
Figura 22. Señales de sincronismo

Tabla 6. Tiempos de sincronismo VGA 640x480 con frecuencia de refresco de 60 Hz

Símbolo	Parámetro	Sincronismo Vertical (T_x)			Sincronismo Horizontal ($T_{x'}$)	
		Duración	*Líneas*	*Ciclos*	*Duración*	*Ciclos*
T_S	Señal de sincronismo	16.68 [ms]	525	420000	31.77 [µs]	800
$T_{VÍDEO}$	Tiempo de	15.25 [ms]	480	384000	25.42 [µs]	640

	despliegue					
T_{BP}	Portal trasero	1.05 [ms]	33	26400	1.91 [µs]	48
T_{FP}	Portal frontal	317.78 [µs]	10	8000	0.64 [µs]	16
T_{PW}	Pulso de sincronismo	63.55[µs]	2	1600	3.81 [µs]	96
$T_{Blanking}$	Bordes señal RGB				6.35 [µs]	160

Todas estas señales que componen la señal VGA son llevadas al monitor a través de su conector VGA. Su distribución se muestra en la siguiente figura.



Pin	Señal
1	Vídeo Rojo
2	Vídeo Verde
3	Vídeo Azul
4	Masa
5	Masa
6	Masa Rojo
7	Masa Verde
8	Masa Azul
9	N/C
10	Masa
11	Masa
12	Datos serie
13	Sinc. Horizontal
14	Sinc. Vertical
15	Reloj (clk)

Figura 23. Conector VGA

2.3 FPGA

2.3.1 Entorno de trabajo ISE

2.3.1.1 Introducción

Actualmente cualquier proceso de ingeniería dispone de un soporte software que asiste al ingeniero de aplicaciones o sistemas en el desarrollo de sistemas complejos. Los sistemas electrónicos reconfigurables del tipo FPGA son un buen ejemplo de la complejidad que se puede alcanzar, esta complejidad no sería abarcable sin la ayuda de un entorno con herramientas que asistan en el proceso de diseño, simulación, síntesis del resultado y configuración del hardware. Un ejemplo de un entorno de este tipo es el software de la empresa Xilinx denominado ISE (*Integrated Software Environment*). Este software constituye un verdadero entorno EDA (*Electronic Desing Automation*). La figura 24 representa el esquema de los componentes más importantes del ISE y la secuencia en que se utilizan.

La interfaz gráfica de usuario (GUI: *Graphic User Interface*) se denomina *Project Navigator* y facilita el acceso a todos los componentes del proyecto. Los diseños de usuario se pueden introducir mediante diferentes formatos. Los más utilizados son: los esquemáticos, los grafos de estados y las descripciones hardware en VHDL. Una vez compilados los diseños se puede simular su comportamiento a nivel funcional o a nivel temporal. A nivel funcional no tiene en cuenta los retardos provocados por el hardware y a nivel temporal simula el diseño teniendo en cuenta cómo se va a configurar el hardware. El último paso será programar la FPGA.

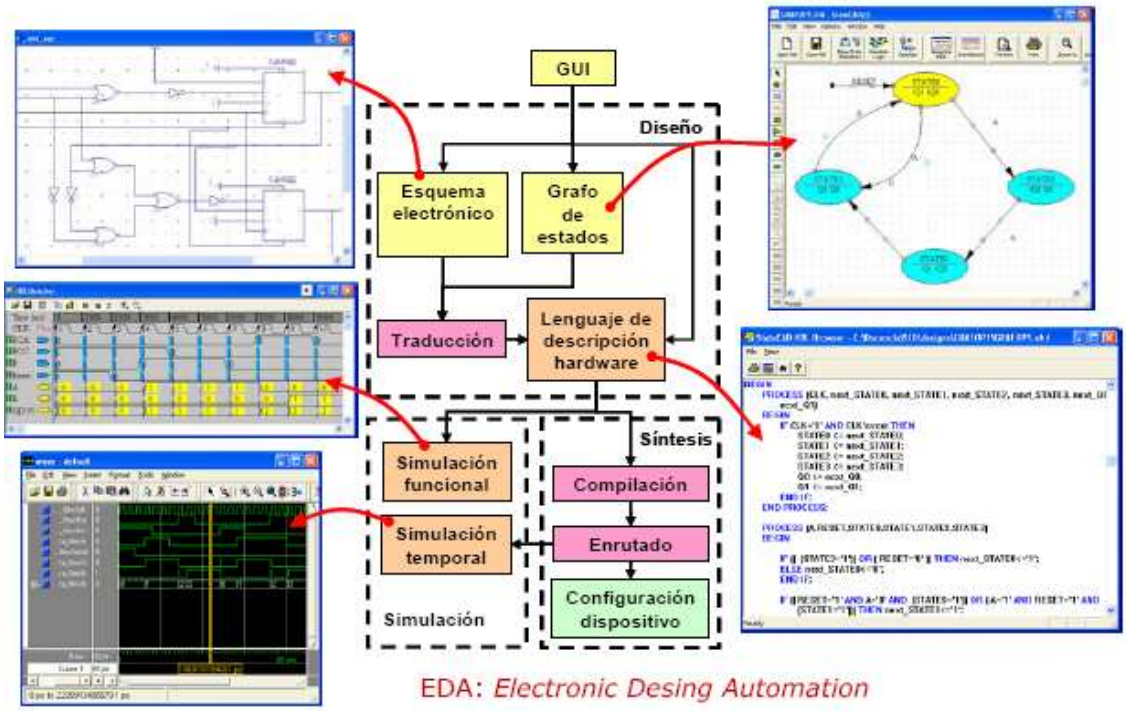


Figura 24. Opciones y flujo de diseño con ISE

2.3.1.2 Flujo de diseño con ISE 8.2i

En este apartado describiremos el flujo de diseño que hemos seguido para crear nuestro proyecto. En concreto, se ha utilizado como software de diseño la versión ISE 8.2i con service pack 3.

Figura 25. Icono Project Navigator del ISE 8.2i de Xilinx

1) Crear el proyecto
   - Seleccionar **File** ⇨ **New proyect**

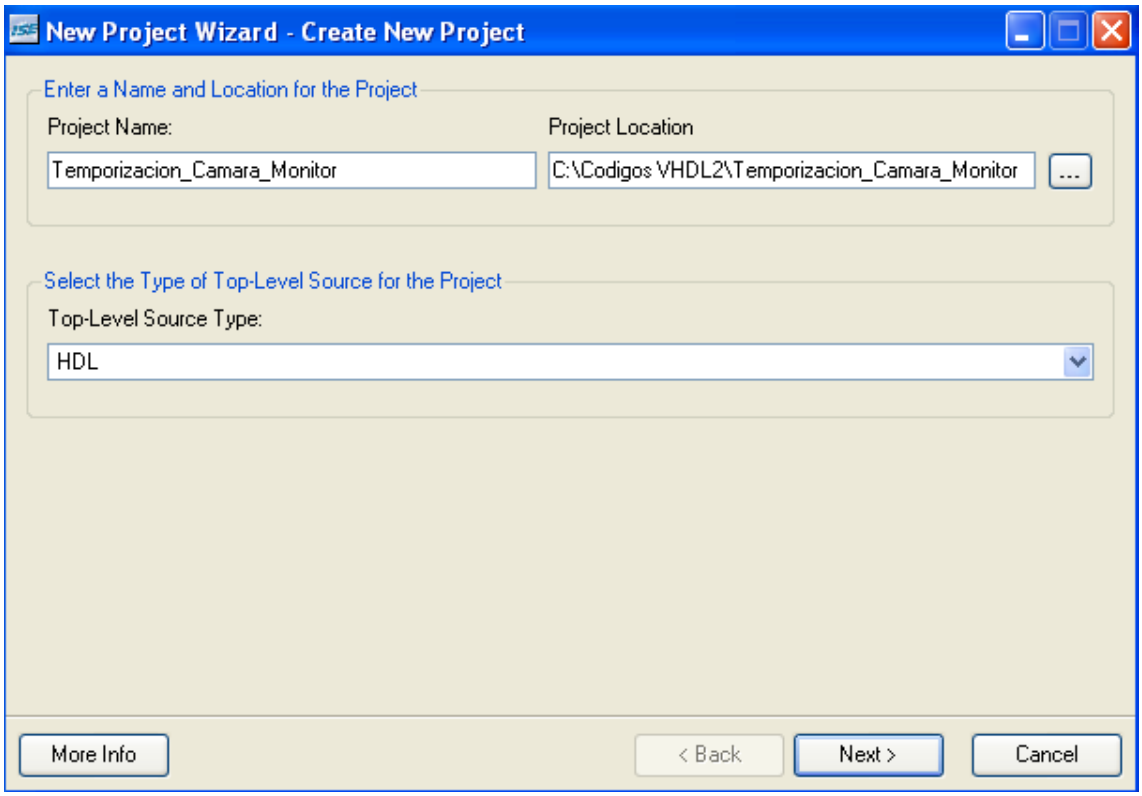


Figura 26. Crear proyecto nuevo

   - Seleccionar **Next** y definir las propiedades del dispositivo:

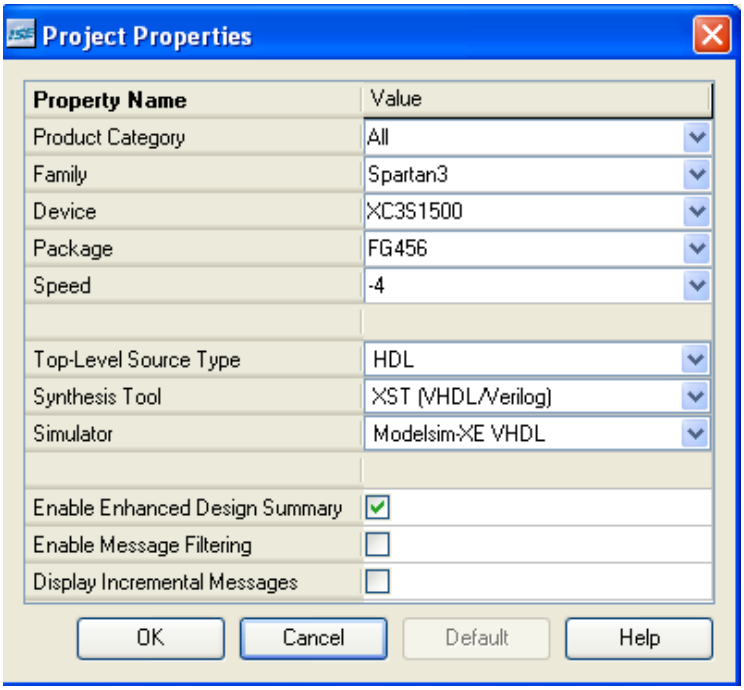


Figura 27. Propiedades del dispositivo

2) Descripción de la arquitectura en VHDL

El siguiente paso es describir toda nuestra arquitectura en lenguaje VHDL. Para ello hemos creado e introducido en el proyecto los siguientes archivos ".vhd" que nos permiten definirla completamente:

TopLevel, GeneradorSincronismos, Modulo1_GeneradorSincronismos, Modulo2_GeneradorSincronismos, Modulo3_GeneradorSincronismos, fifoctlr_ic_v2 y AdaptadorVsyn.

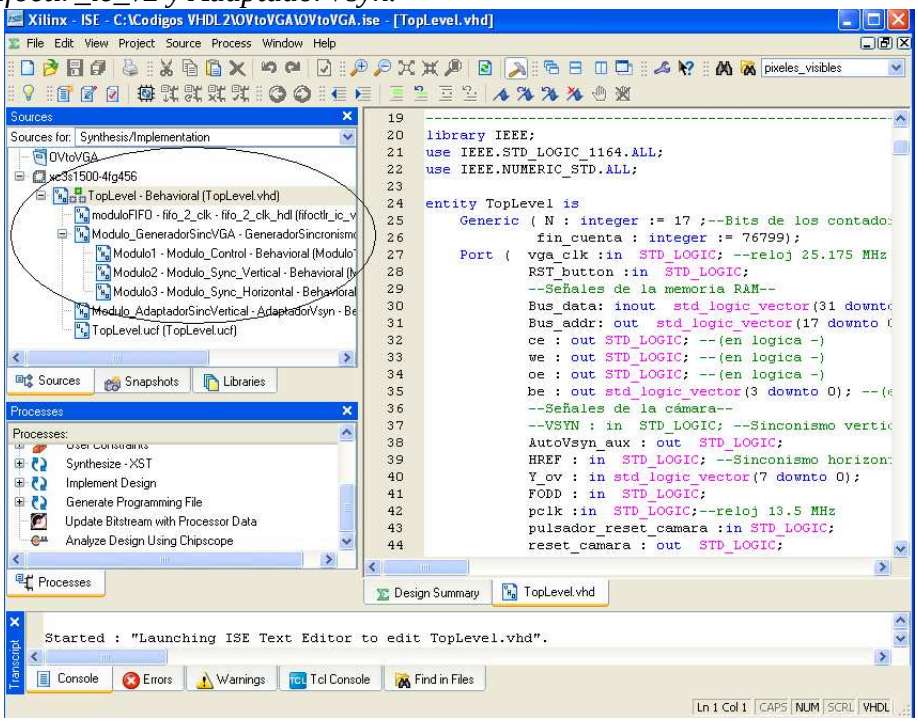


Figura 28. Descripción de la arquitectura en VHDL

3) Síntesis

Este proceso se define como la tarea de traducir un modelo a nivel RTL de un sistema hardware expresado en un lenguaje de descripción hardware, en una

implementación a nivel de puertas del circuito, optimizada para una tecnología electrónica específica. Consta de tres fases de optimización: a nivel RTL, a nivel lógico y a nivel de puerta.

- Hacer doble clic sobre **Synthetize** en la ventana de procesos.

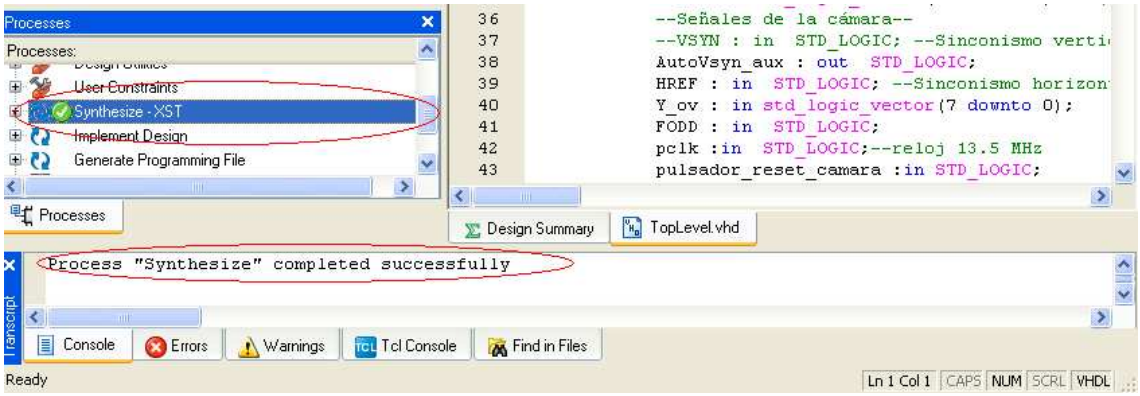


Figura 29. Proceso de síntesis

4) Diseño de un banco de pruebas: testbench

El lenguaje VHDL nos permite definir test para un circuito, además de describir el circuito en sí. Para ello creamos un archivo de testbench tomando como fuente el TopLevel.vhd (archivo principal de la arquitectura) y definimos la forma y duración de las señales de entrada a nuestra arquitectura.

- Seleccionar **Project ⇨ New source ⇨ VHDL Test Bench**

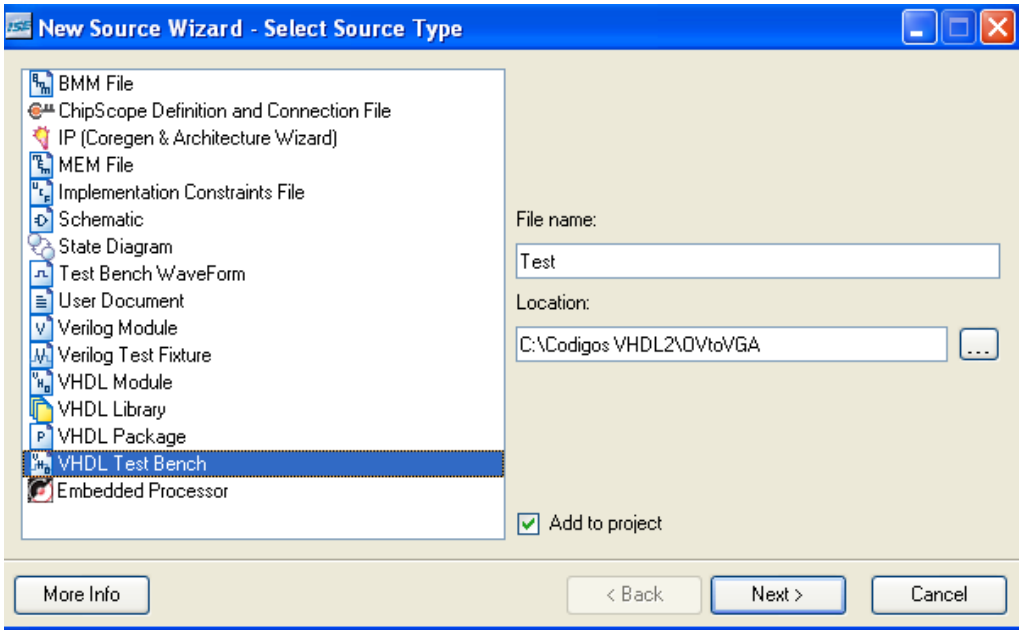


Figura 30. Nuevo archivo -> testbench

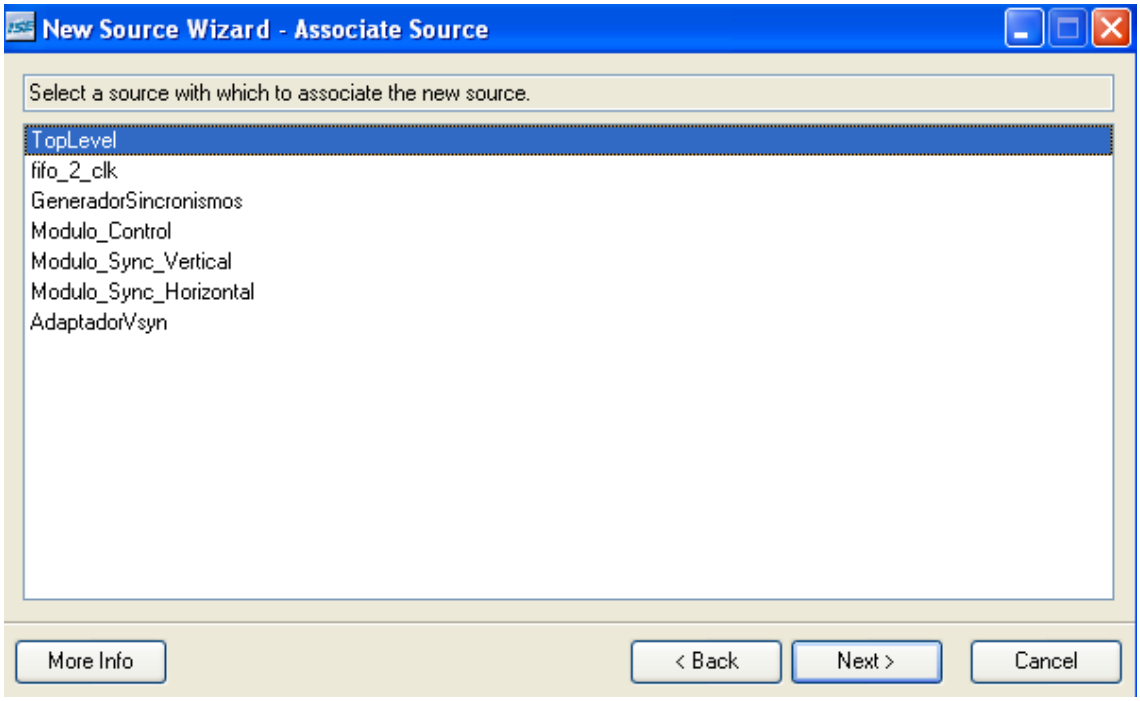


Figura 31. Selección del archivo fuente del testbench

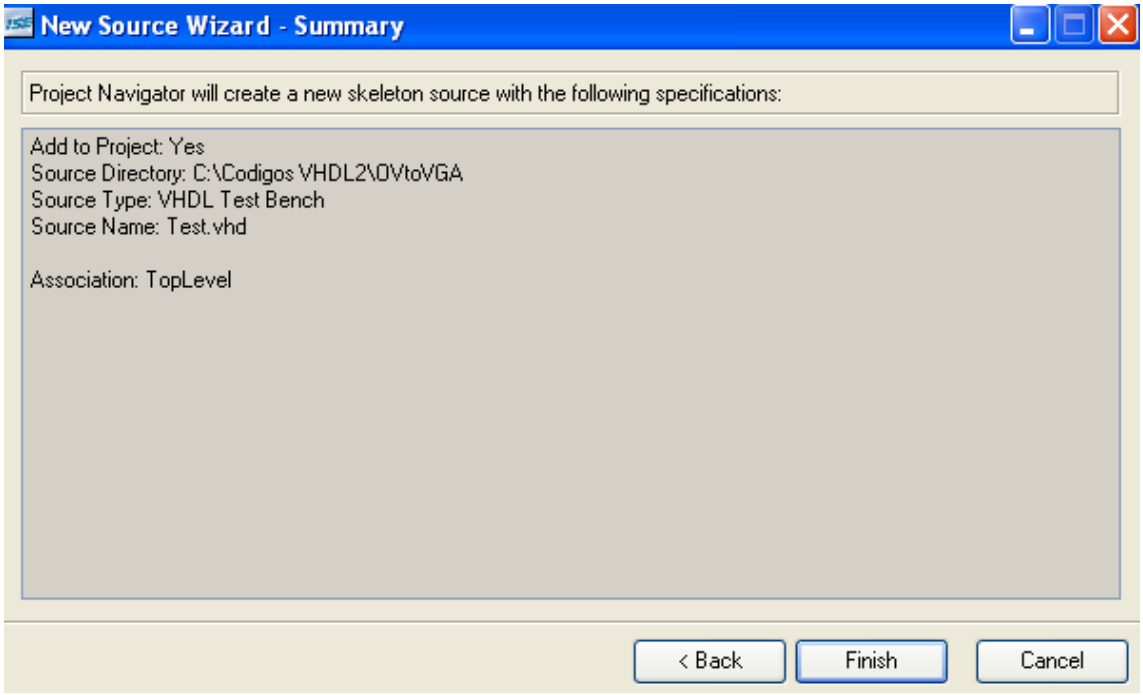


Figura 32. Resumen del testbench creado

5) Simulación Funcional

El objetivo de realizar esta simulación es verificar el comportamiento del circuito diseñado. Para ello se empleará la herramienta de simulación *ModelSim SE 6.0d* asociada a *Project Navigator*.

Se trata de un modo de simulación con retardos de propagación nulos. Esto significa que un cambio en la entrada produce un cambio en la salida instantáneamente.

- Seleccionar en la ventana **Sources** la opción **Behavioral Simulation**

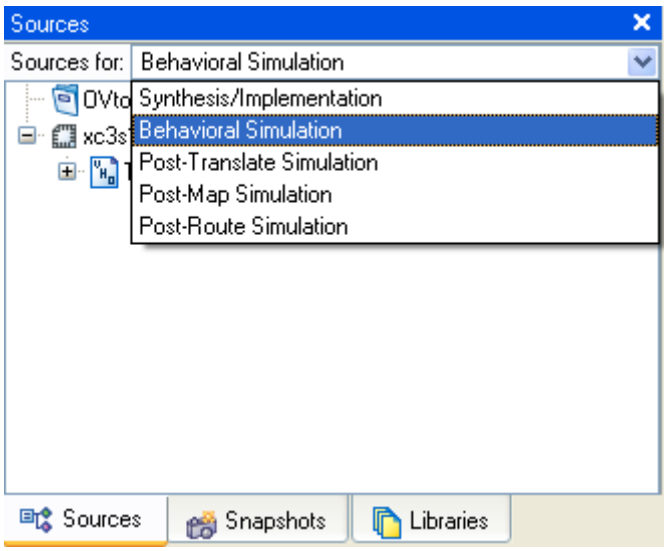


Figura 33. Selección Simulación Funcional

- Expandir en la ventana de procesos **ModelSim Simulator** y hacer doble clic sobre la opción **Behavioral Simulation Model**

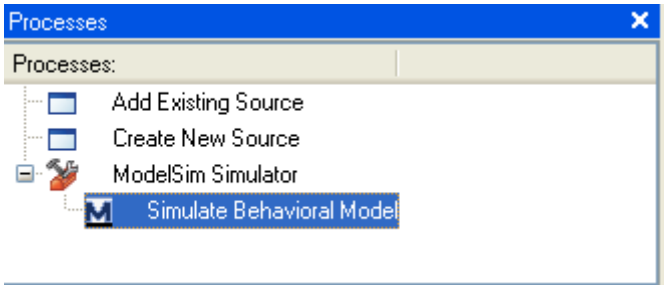


Figura 34. Acceso directo a Simulación funcional en ModelSim 6.0d

- Inmediatamente se abrirá el programa *ModelSim* mostrando los resultados de la simulación funcional definida por el testbench:

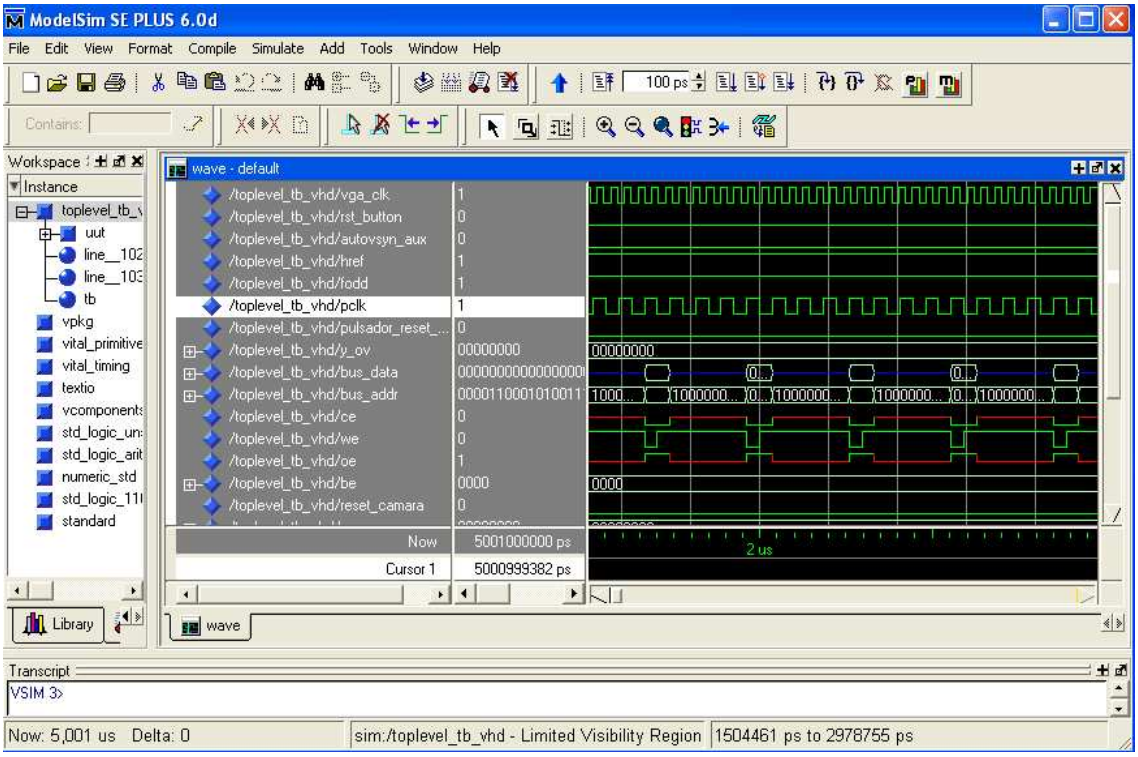


Figura 35. Simulación funcional en ModelSim 6.0d

6) Crear archivo *.ucf*

Una vez comprobado el comportamiento del circuito con la simulación funcional, el siguiente paso es definir los pines de entrada y salida en un archivo *.ucf*.

- Seleccionar **Project ⇨ New source ⇨ Imlementation Constraints File**

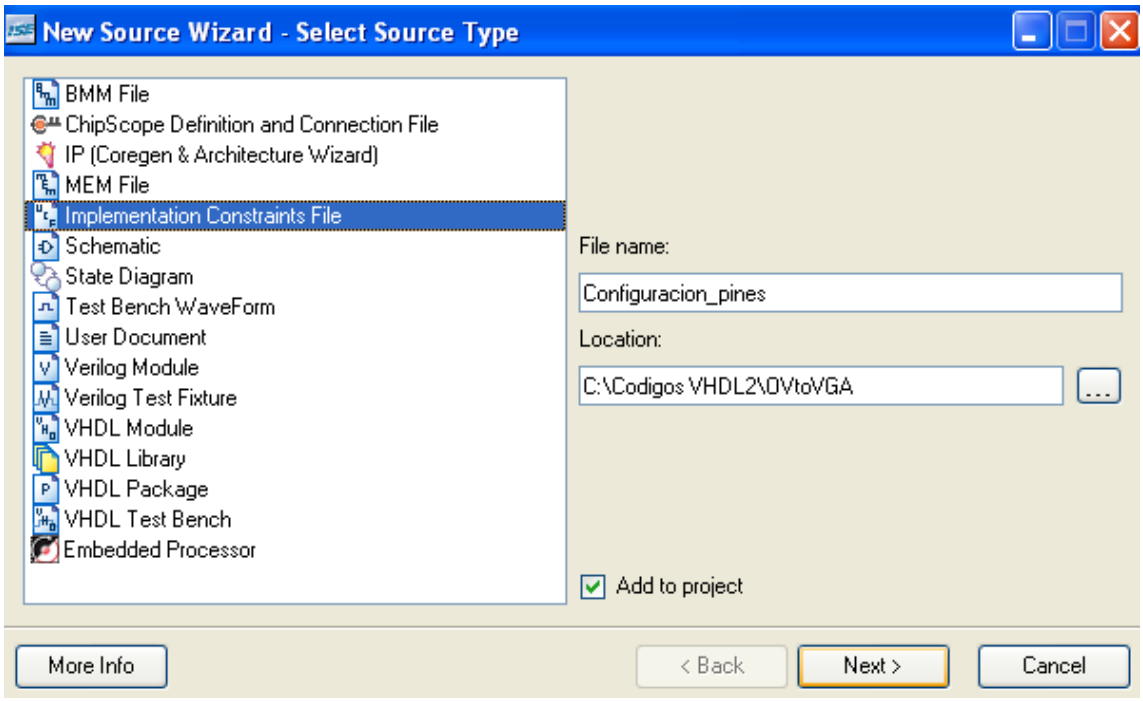


Figura 36. Nuevo archivo -> Implementation Constraints File

- Seleccionar el archivo *TopLevel.vhd* como archivo fuente.

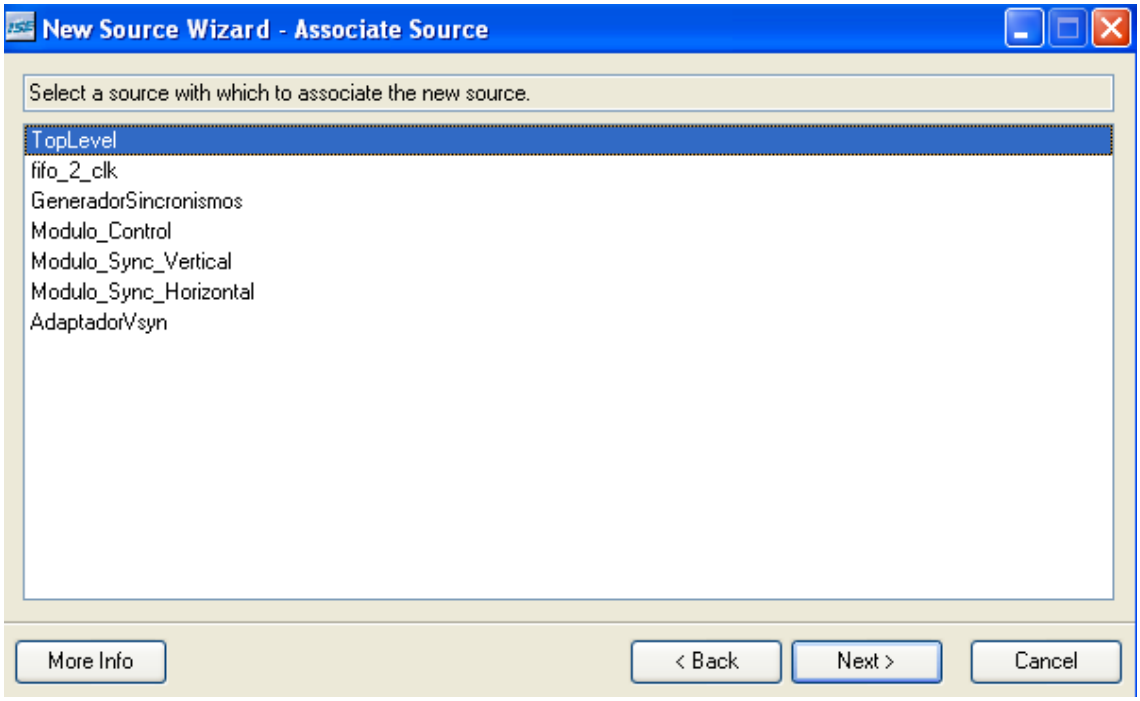


Figura 37. Selección del archivo fuente para el *.ucf*

Finalmente obtenemos un archivo que define a que pines de la placa están conectadas las entradas y salidas de nuestra arquitectura:

```
1  ####Cámara####
2  NET "Y_ov<0>" LOC = "Y6" ; # Pin 6 (JP7)
3  NET "Y_ov<1>" LOC = "AB4"; # Pin 8 (JP7)
4  NET "Y_ov<2>" LOC = "AA4"; # Pin 10 (JP7)
5  NET "Y_ov<3>" LOC = "W8"; # Pin 12 (JP7)
6  NET "Y_ov<4>" LOC = "Y5"; # Pin 14 (JP7)
7  NET "Y_ov<5>" LOC = "W5"; # Pin 16 (JP7)
8  NET "Y_ov<6>" LOC = "AA9"; # Pin 18 (JP7)
9  NET "Y_ov<7>" LOC = "W6"; # Pin 20 (JP7)
10 NET "HREF"  LOC = "W11"  | PULLUP ; # Pin 30 (JP7)
11 ##NET "VSYN"  LOC = "Y11" | PULLDOWN ; # Pin 28 (JP7) (Prueba Adapte
12 NET "pclk" LOC = "U7"; # Pin 9 (JP7) reloj externo (13.5MHz)
13 NET "pulsador_reset_camara" LOC = "W2"; # (sw4)
14 NET "reset_camara" LOC = "Y4"; # Pin 2 (JP7)
15 NET "FODD"  LOC = "AA6" | PULLDOWN ; # Pin 7 (JP7)
16
17 ###Adaptador Sincronismo vertical###
18 NET "AutoVsyn_aux" LOC = "V6"; # Pin 22 (JP7)
19
20 ####Memoria RAM Asíncrona####
21 NET "Bus_addr<0>" LOC = "E22";
22 NET "Bus_addr<10>" LOC = "F18";
23 NET "Bus_addr<11>" LOC = "G18";
24 NET "Bus_addr<12>" LOC = "G19";
25 NET "Bus_addr<13>" LOC = "E18";
26 NET "Bus_addr<14>" LOC = "F19";
```

Design Summary | TopLevel.ucf

Figura 38. Código VHDL

7) Implementación

Este proceso elabora el modelo físico proyectado sobre un chip concreto. Consta de varias tareas entre las que destacan la selección de la ubicación y el enrutado. La primera define la posición de cada una de las celdas que componen nuestro circuito sobre la superficie disponible del chip. La segunda define la ruta de

las conexiones existentes entre estas celdas y las celdas que se conectan a las patillas del circuito integrado. A estas dos tareas se les conoce conjuntamente como Place&Route.

- Hacer doble clic sobre **Implement Design** en la ventana de procesos.

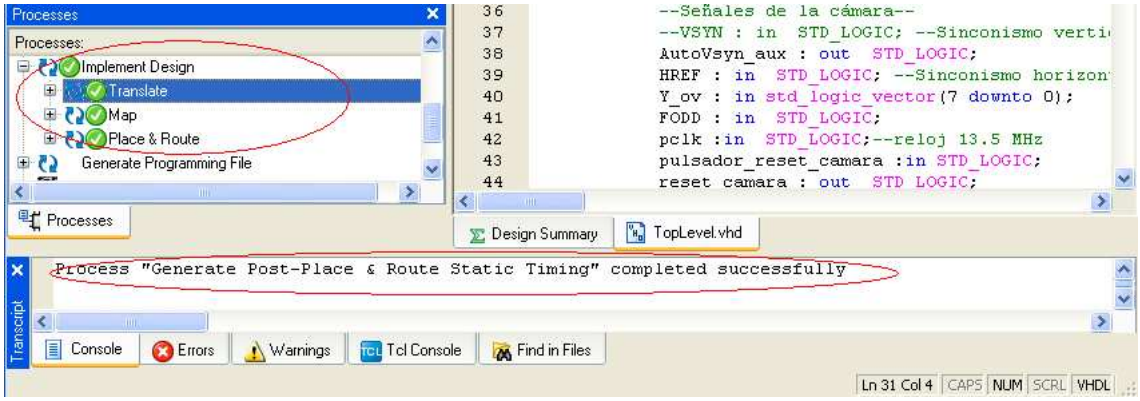


Figura 39. Proceso de implementación

8) Simulación temporal

Se trata del modelo de simulación más completo que podemos realizar. En él son considerados los retardos de propagación de todos los modelos de los dispositivos primitivos. Estos modelos permiten al simulador ModelSim SE 6.0d proporcionar información sobre violaciones temporales, tiempos de establecimiento insuficientes, etc.

- Seleccionar en la ventana **Sources** la opción **Behavioral Simulation**

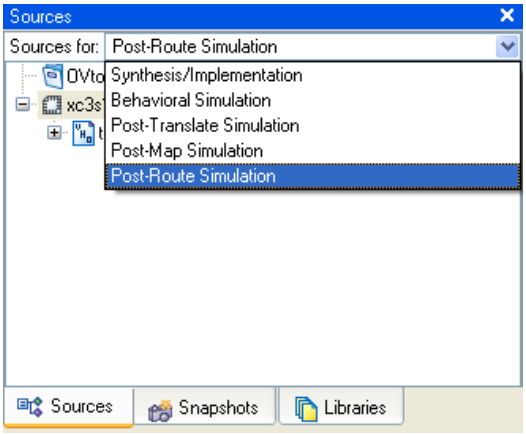


Figura 40. Selección Simulación Temporal

- Expandir en la ventana de procesos **ModelSim Simulator** y hacer doble clic sobre la opción **Simulate Post-Place & Route Model**

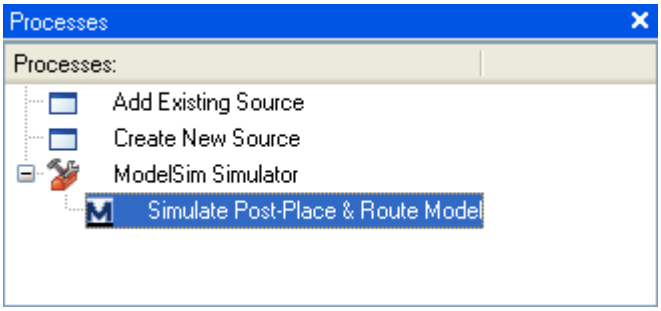


Figura 41. Acceso directo a Simulación temporal en ModelSim 6.0d

Inmediatamente se abrirá el programa *ModelSim* mostrando los resultados de la simulación temporal definida por el testbench:

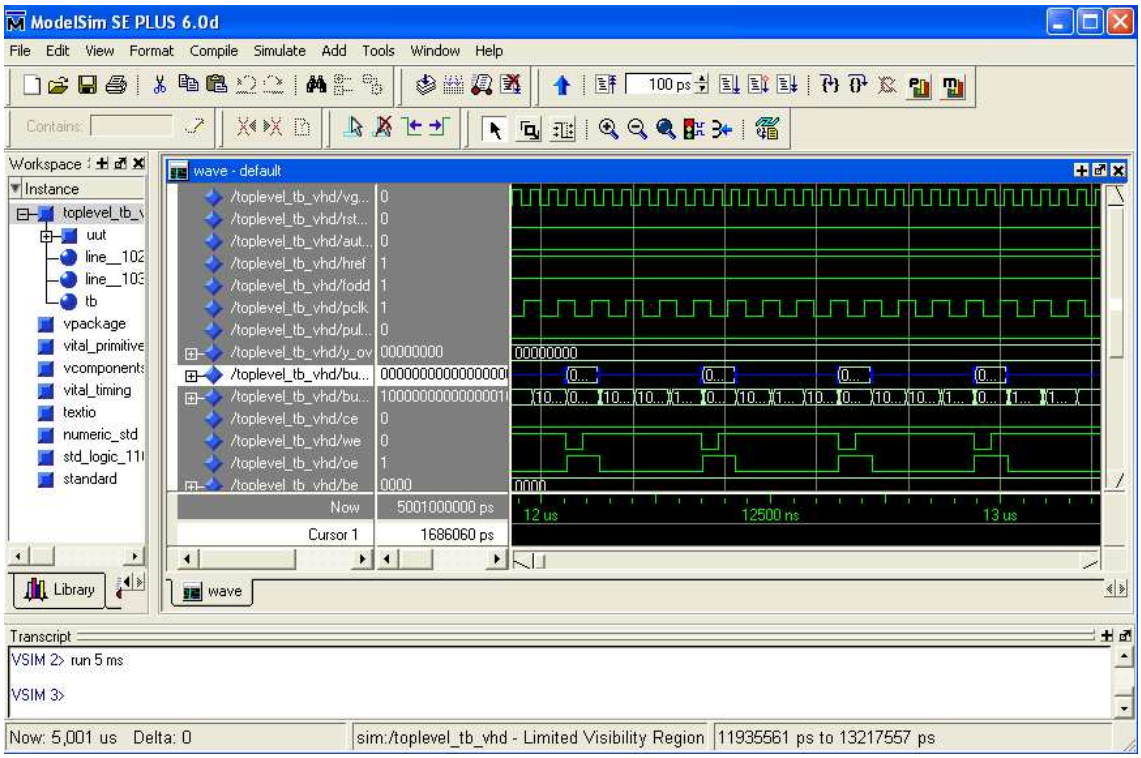


Figura 42. Simulación temporal en ModelSim 6.0d

9) Configurar la FPGA

9.1) Generar el fichero de programación

- Hacer **doble clic** sobre la opción **Generate Programing File** de la ventana de procesos.

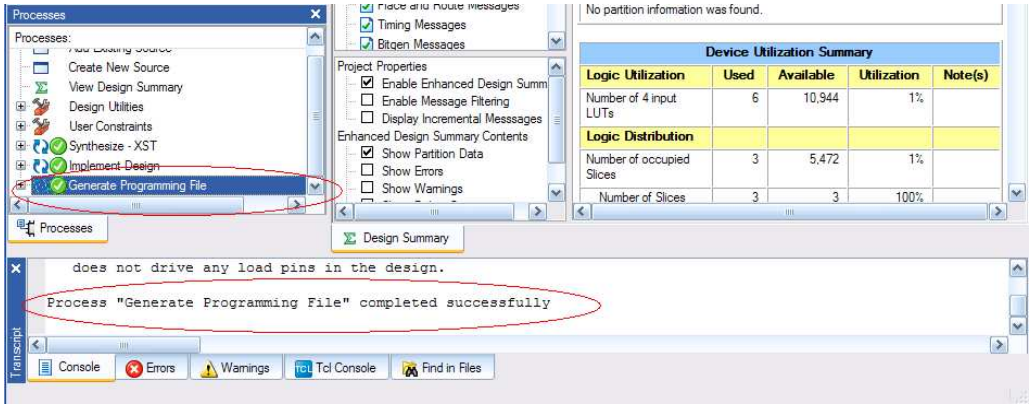


Figura 43. Proceso para crear el archivo de configuración

9.2) Configuración del hardware

- Conectar el transformador de 5V de continua a la entrada de alimentación de la placa.
- Conectar el cable de descarga **JTAG Paralelo IV** entre el PC y la placa.

Figura 44. Cable de descarga JTAG Parallel IV

9.3) Programar la FPGA

- Expandir en la ventana de procesos **Generate Programing File** y hacer **doble clic** sobre la opción **Configure Device (iMPACT).** Se abrirá el software de configuración iMPCAT

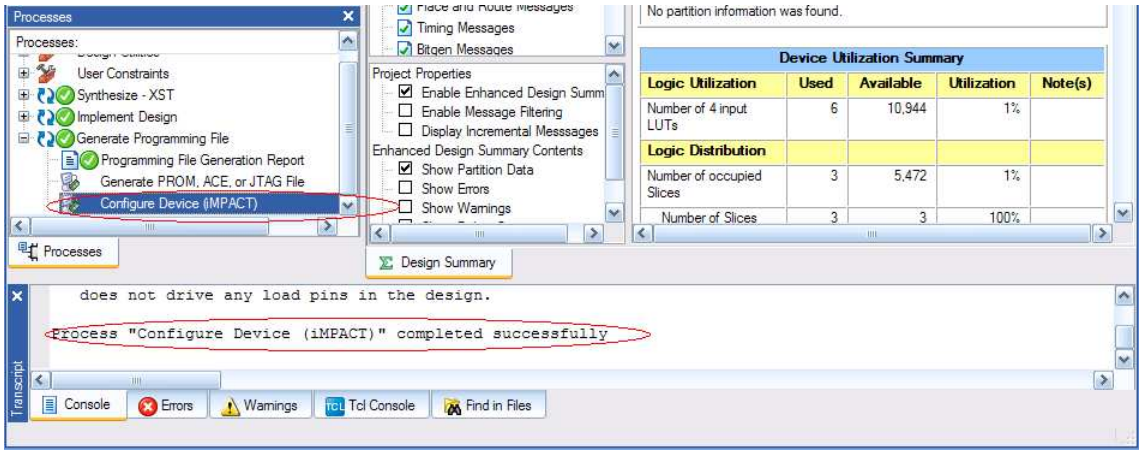


Figura 45. Abrir software de configuración de dispositivos iMPACT

- Seleccionar **Configure Devices Using Boundary Scan** como modo de configuración. Dentro de este modo seleccionar **Automatically conect to a cable and identify Boundary-Scan chain** para que la identificación de los componentes de la cadena sea automática. **Clic** sobre **Finish**.

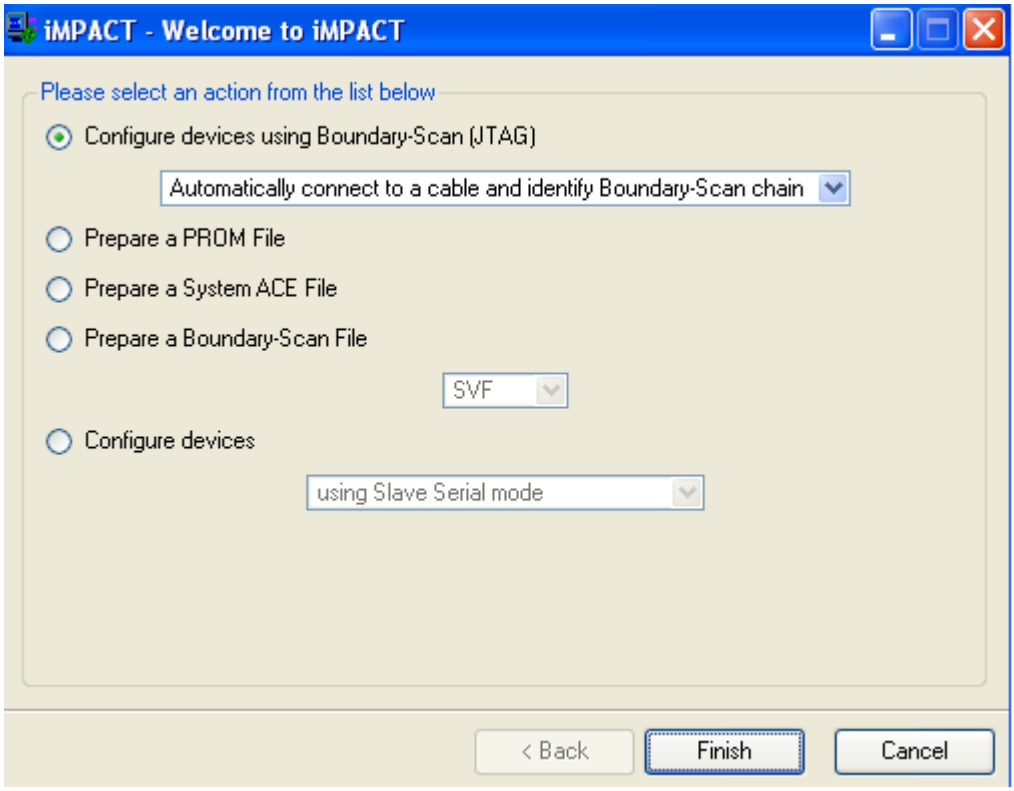


Figura 46. Cuadro de diálogo de iMPACT

- Aparece un mensaje que indica que hay 3 dispositivos. Hacer **clic** en **OK**.
- A continuación aparece el cuadro de diálogo **Assign New Configuration File**. Para asignar el archivo de configuración del dispositivo xc3s1500 seleccionamos la opción Bypass para los dos primeros dispositivos y sobre el tercero se selecciona el archivo TopLevel.bit y se hace clic en Open.

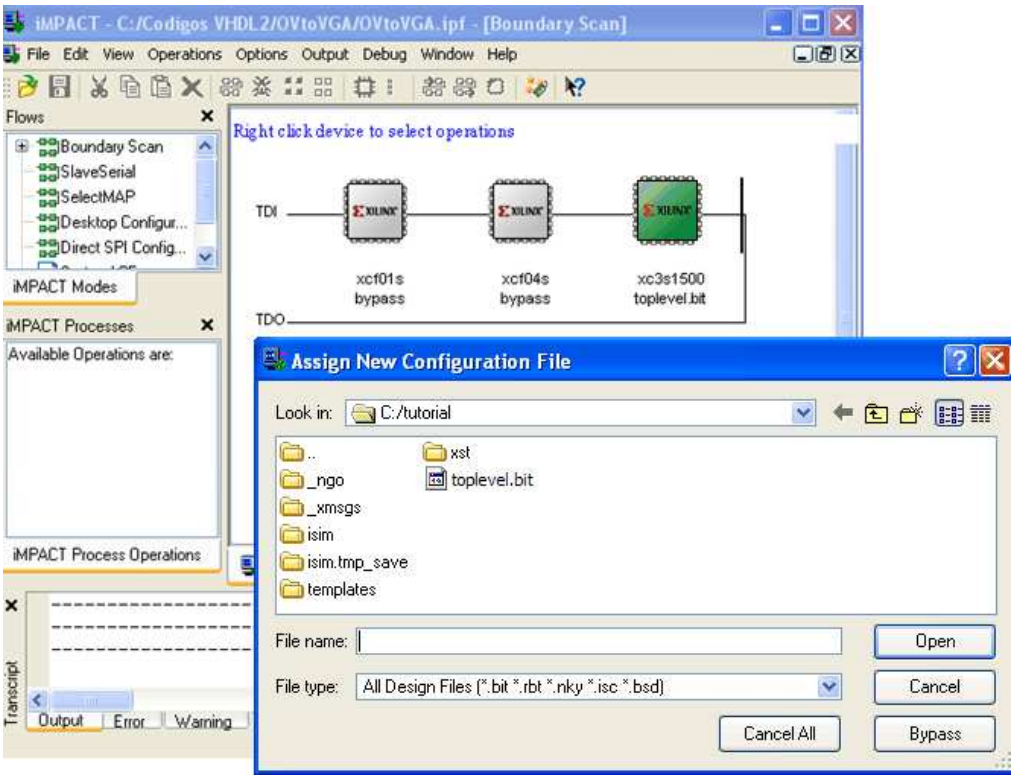


Figura 47. Asignación de nuevo archivo de configuración

- Obtenemos una identificación satisfactoria de todos los elementos de la cadena.

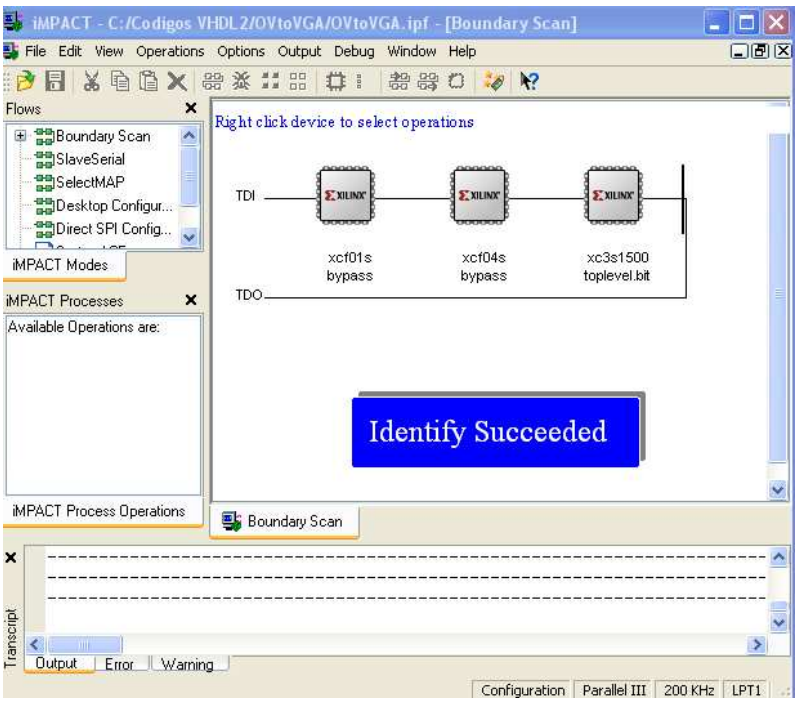


Figura 48. Identificación satisfactoria de los dispositivos

- **Clic** con el botón derecho sobre el dispositivo **xc3s1500** y seleccionar la opción **Program**. Se abre una ventana, pulsamos **OK** y ya esperamos a que se configure la FPGA.

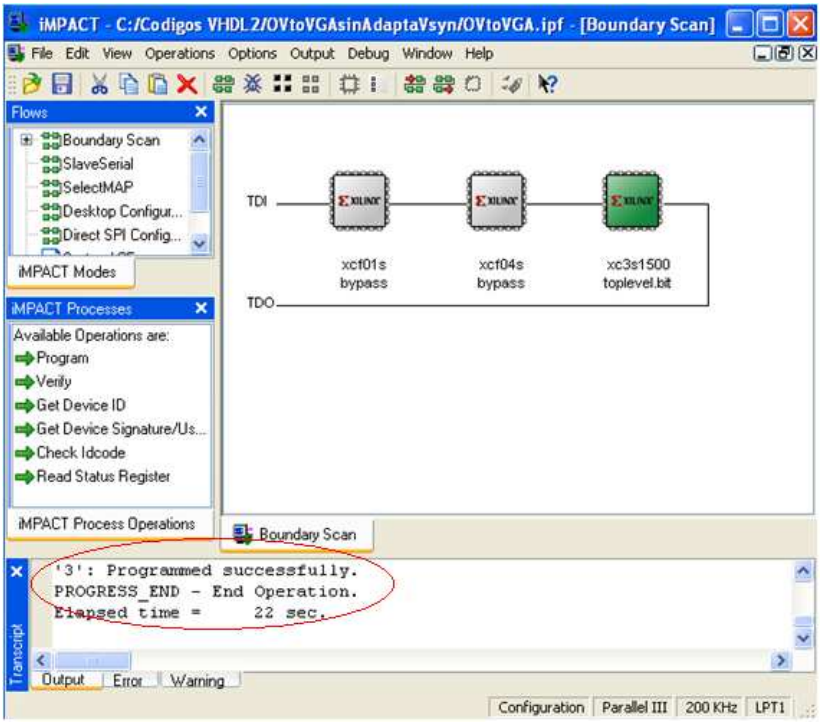


Figura 49. FPGA programada

2.3.2 Descripción de la placa

La placa utilizada para desarrollar nuestro proyecto es una Spartan 3 de Xilinx, en su modelo XC3S1500-FG456.

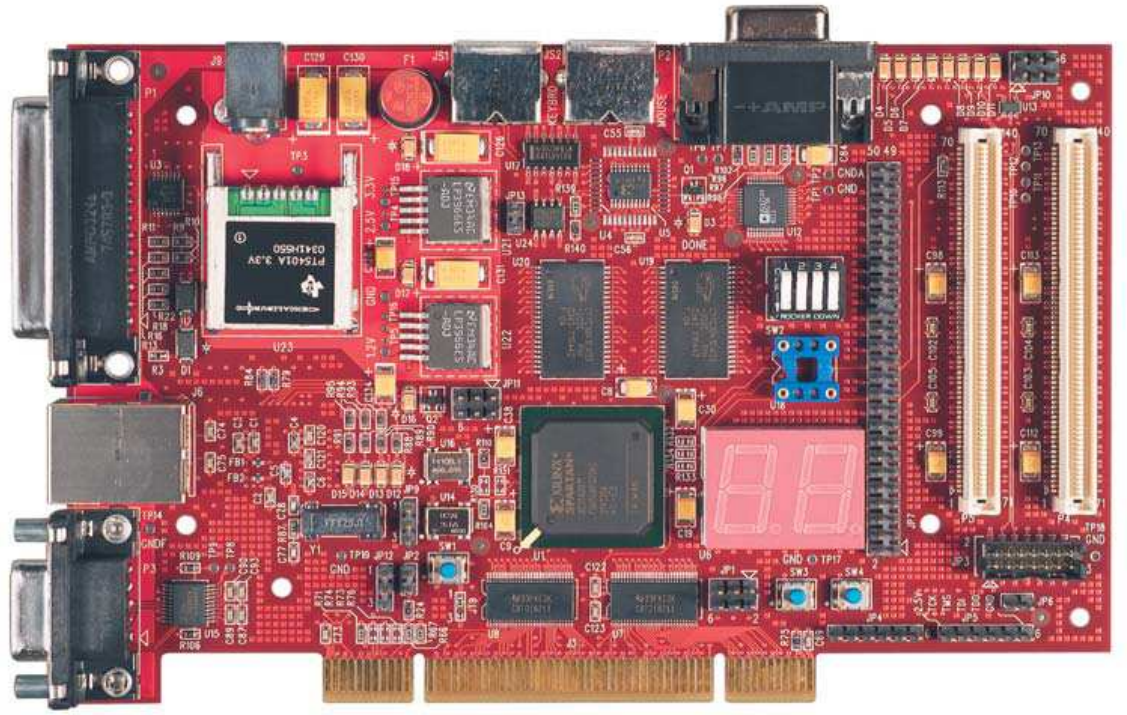


Figura 50. Placa Xilinx Spartan 3- XC3S1500-FG456

Características:

- FPGA
  - Xilinx Spartan-3 XC3S1500-FG456
- Board I/O Connectors
  - Dos buses de expansion de entrada/salida de propósito general
  - 50 – Pin 0.1” Header (4 LVDS pairs)
  - Bus PCI 32 bits (3.3V/5V universal)
- Memoria
  - Cypress SRAM Asíncrona -1MB
  - Atmel Serial EEPROM – 1Mbit 256kb (on 3S1500)
- Comunicación
  - 10/100 base T Ethernet
  - Dos puertos PS2
  - Puerto serie RS-232
- Vídeo
  - Vídeo RGB (Convertidor Analógico-Digital)
- Entradas/Salidas analógicas
  - DAC (para Xilinx AP-Note XAPP154)
  - A/D (para Xilinx AP-Note XAPP155)
- Alimentación
  - 10+ Wattios AC/DC Alimentación +5.0V
  - Módulo 3.3V 6A deTexas Instruments
  - National Linear regulators
- Configuration

- Xilinx XCF02S PROM (XCF04S+XCF01S)
- Circuitería para cable Paralelo III y soporte para cable Paralelo IV

Información de usuario

En esta sección se presenta información sobre cómo utilizar la Spartan-3. Trataremos aspectos como la configuración de los dispositivos FPGA o puesta en marcha los jumpers.

Alimentación

La placa se alimenta con un adaptador AC/DC a 5 V.

Configuración

Esta placa permite distintos modos para su configuración. En nuestro caso, hemos empleado el modo Boundary-San. Programaremos la FPGA mediante la herramienta iMPACT de Xilinx utilizando este modo.

Boundary-Scan

Programar la FPGA mediante este modo requiere un cable de descarga JTAG .

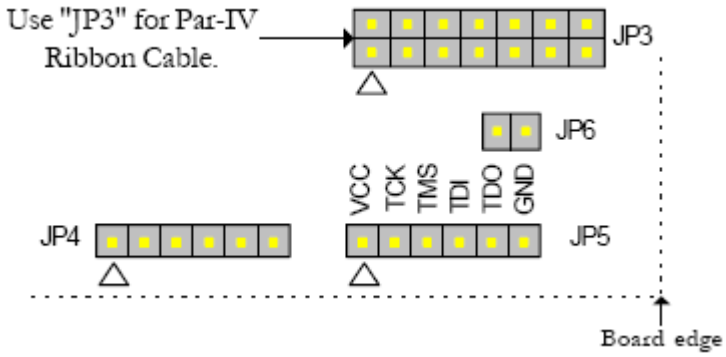
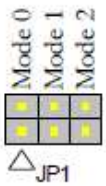


Figura 51. Configuración Boundary Scan

Ajuste de los Jumpers

Esta sección proporciona la descripción de los jumpers de la placa. A continuación, presentamos la lista de los jumpers ordenado por el número de JP.

JP1- Selección del modo de configuración. Utilizado para seleccionar el modo de configuración de la FPGA. Viene configurado por defecto en el modo Master Serial.



JP1	1-2	3-4	5-6
Config Mode	M0	M1	M2
Master Serial	0	0	0
Slave Serial	1	1	1
Master SelectMAP	1	1	0
Slave SelectMAP	0	1	1
Boundary Scan	1	0	1

Figura 52. JP1

JP2 - Habilita los pull-ups en los pines de entrada/salida durante la configuración. En pull-down se emplea la resistencia “R24”.

JP3 - Conector paralelo IV. Ver el modo de configuración Boundary – Scan

JP4 - Configuración del JTAG.

JP5 – Conector Flying Lead para la interfaz del JTAG.

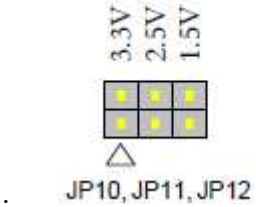
JP6 – JTAG TRST, fuerza TRST a baja.

JP7 – Pines de propósito general para entradas y salidas.

JP9 – Reloj de vídeo deshabilitado. El oscilador "U14"de 25.175 MHz proporciona el reloj para la interfaz VGA en la placa. El reloj está habilitado si la derivación se sitúa en los pines 1-2 y deshabilitado cuando está en los pines 2-3.

JP10 – Interfaz analógica de entrada/salida.

JP11 – Configuración de la tensión de referencia VCCO. Las distintas posibilidades se muestran en la Figura 53.



Jumper Position	1-2	3-4	5-6
I/O Voltage	3.3V	2.5V	1.2V

Figura 53. Selección de voltage de Entrada/Salida

JP12 – Activa disipador de potencia.

JP13 – Protección de escritura en EEPROM

Hardware

En esta sección se describe el hardware de la placa Spartan 3. Nos centraremos en aquellos que hayan sido utilizados en nuestro proyecto. La siguiente figura muestra el diagrama de bloques de la placa.

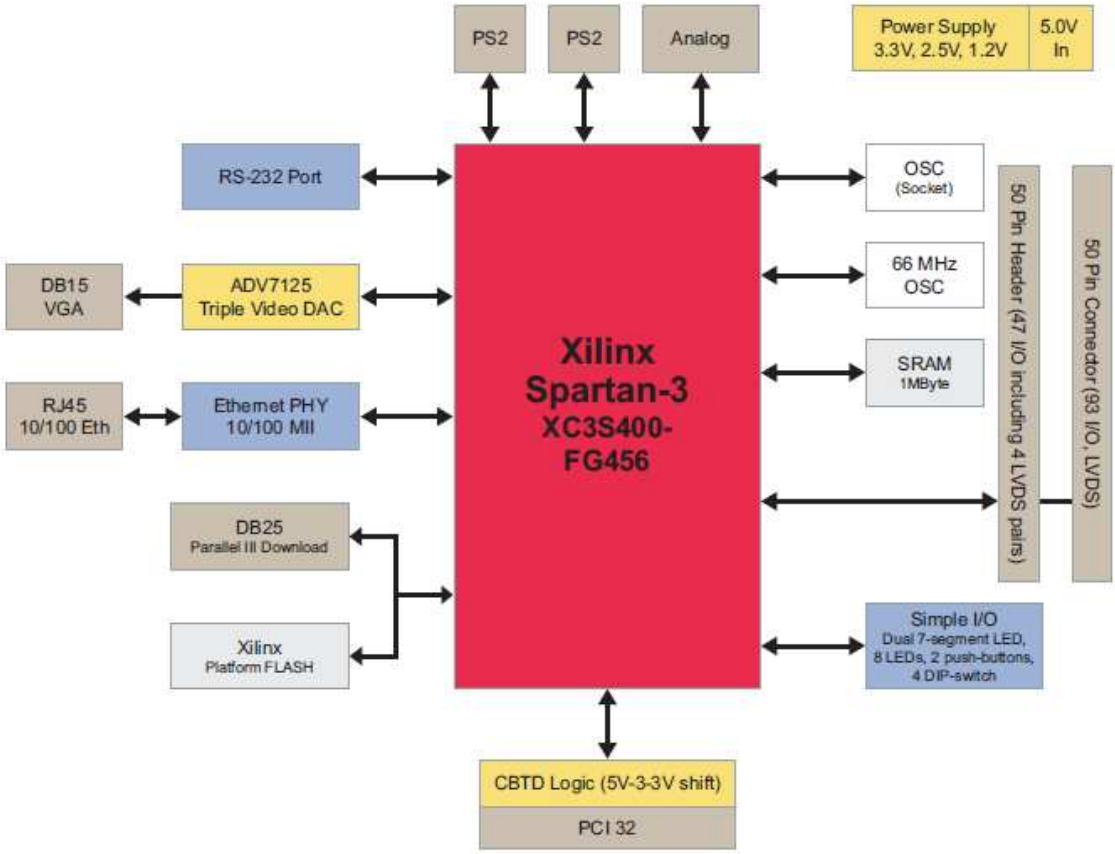


Figura 54. Diagrama de Bloques de la Spartan 3

Spartan 3 FPGA

En nuestro caso la FPGA empleada es la FG456 en su rango de densidad de integración más alto (3S1500). La placa ha sido limitada a 264 pines de entrada/salida.

Figura 55. Spartan 3 XC3S1500 (FG56)

Memorias

La placa Spartan 3 dispone de dos dispositivos de memoria SRAM Asíncrona y de un dispositivo EEPROM Serie.

SRAM Asíncrona

Los dos dispositivos integrados en la placa son Cypress del tipo CY7C1041CV33. Se trata de una memoria SRAM estática en tecnología CMOS, organizada en 256x1024 posiciones de 16 bits.

Figura 56. Chip CY7C1041CV33

En nuestro proyecto configuraremos las memorias para que trabajen en paralelo, de modo que podamos considerarlas como una sola memoria de datos de 32 bits. Ver figura
57.

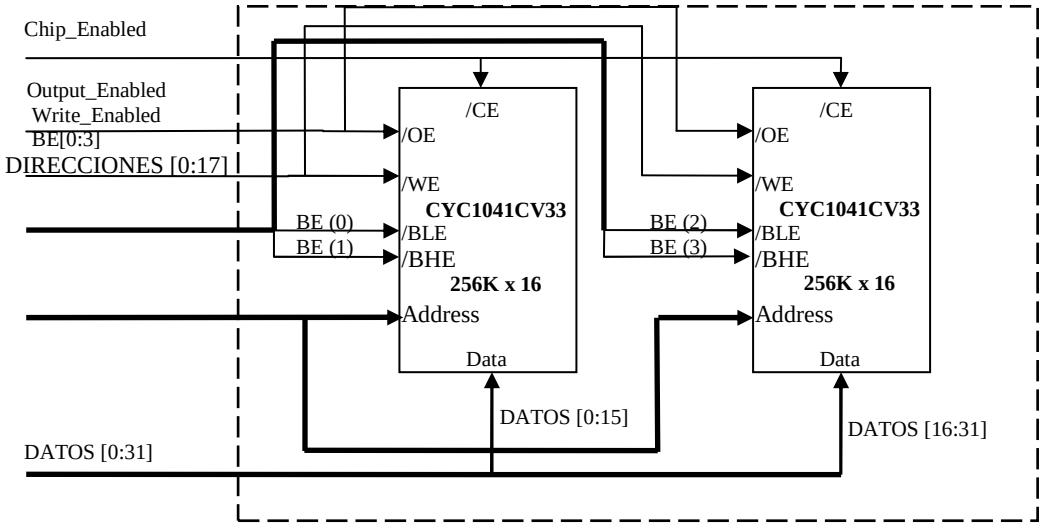


Figura 57. SRAM Asíncrona de tamaño 256x32 construida a partir de dos CYC1041CV33 en paralelo

EEPROM Serie

El dispositivo EEPROM de la placa es un Atmel del tipo AT24C256W-10SI. Se trata de una memoria ROM programable y borrable eléctricamente. Además es una memoria no volátil y una vez programada los datos pueden estar protegidos con la correcta configuración del JP13. Se utiliza para reconfigurar la FPGA.

Figura 58. Dispositivo de memoria EEPROM AT24C256W-10SI

Vídeo RGB

La señal de vídeo RGB puede ser conducida a un monitor VGA desde la placa, a través del convertidor ADV7125. Se trata de un convertidor Digital/Analógico de Triple vídeo. Su salida está conectada a un conector DB15 que soportan la mayoría de los monitores comercializados.

Figura 59. Salida de vídeo RGB. Convertidor ADV7125 a conector DB15 macho

En resumen, si tenemos información RGB y queremos sacarla por un monitor VGA, sólo tendremos que conectar las componentes de la señal VGA al convertidor ADV7125. Esto supone introducirle la información y generar las señales de control para trabajar con el estándar VGA.

CAPÍTULO 3. Sistema hardware para adaptar las temporizaciones cámara-monitor

3.1 INTRODUCCIÓN

El objetivo de este proyecto es adaptar la salida de la cámara OV7620 para poder ver las imágenes que capture en un monitor que funcione con el estándar VGA. Para ello será necesario resolver dos problemas de compatibilidad entre ambos dispositivos.

El primer problema que se plantea es transformar la resolución de la cámara, que es de 858x525 píxeles, a una resolución de 800x525 píxeles que utiliza el monitor.

El otro inconveniente es la adaptación de frecuencias (tasa de refresco), puesto que la cámara funciona a 30 Hz, es decir, captura 30 frames por segundo, mientras que el monitor funciona 60 Hz (pinta 60 frames por segundo).

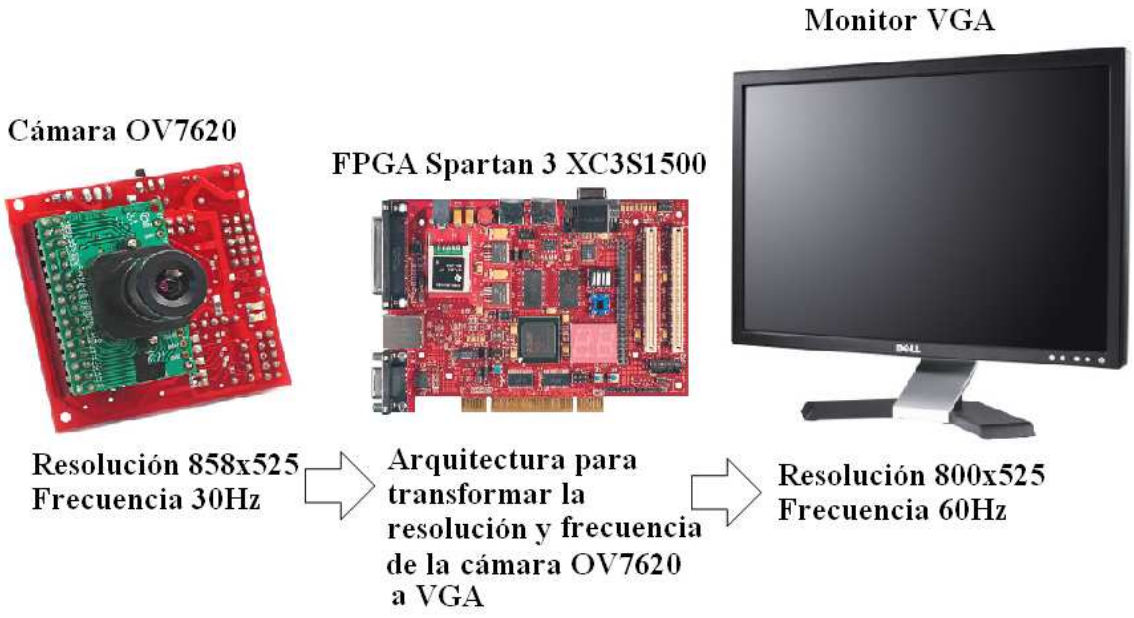


Figura 60. Objetivos del proyecto: adaptación de la resolución y frecuencia de la cámara OV7620 al estándar VGA

En realidad, este proyecto ha sido enfocado para dar soporte al procesado de las imágenes que recoge la cámara. Como línea futura este proyecto se unirá a otro que se encargue de este procesamiento a través de una red neuronal y que obtenga la silueta de la imagen capturada por la cámara. Por tanto, la entrada de información sería la que muestra la figura 61.

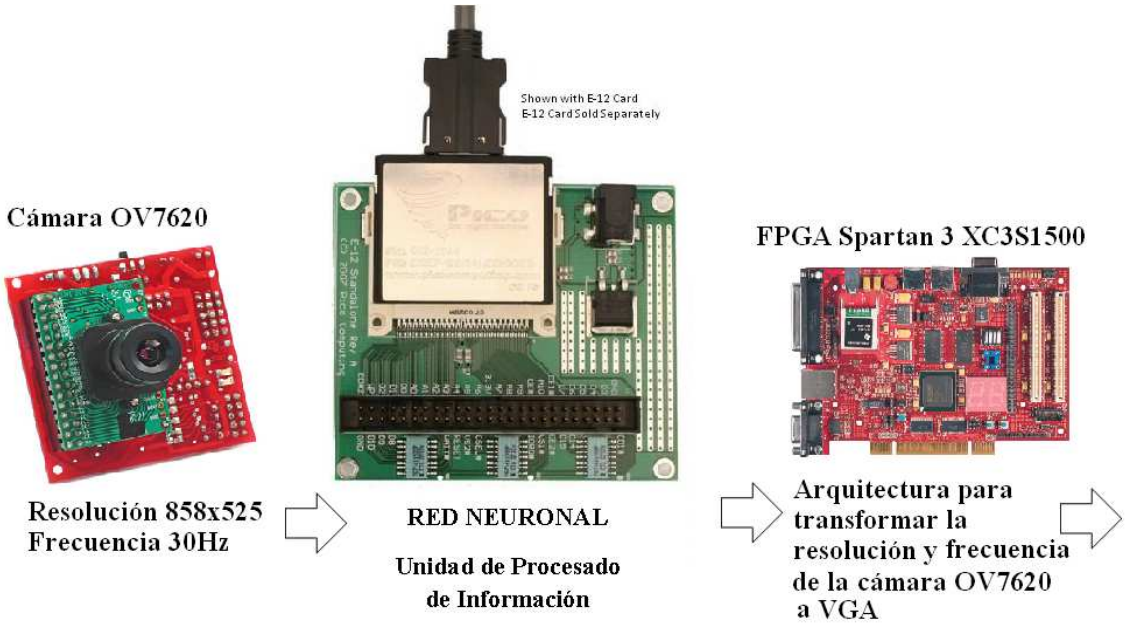


Figura 61. Posible disposición del proyecto con la fase de procesado de información

3.2 DESCRIPCIÓN GENERAL DE LA ARQUITECTURA

La arquitectura del proyecto se haya expuesta en el "Anexo B" del informe y resuelve todos los problemas citados en la Introducción. Los datos de entrada a la arquitectura proceden de la cámara así como las señales de sincronismo horizontal *y* vertical *(HSYN* y *VSYN)* que pone el circuito en funcionamiento. A continuación procedemos a describir el funcionamiento de la arquitectura de manera global, dejando los detalles de cada módulo para el siguiente apartado.

En primer lugar debemos saber que en esta arquitectura vamos a trabajar con dos "dominios de reloj" delimitados por dos relojes distintos: el *PCLK* (13.5 Mhz) procedente de la cámara Ov7620 y el *VGA_CLK* (25.175 Mhz) procedente de un oscilador de la FPGA y que será el reloj maestro para el estándar VGA. Por tanto, tendremos una primera parte de la arquitectura que funcionará con el reloj *PCLK* a la que denominaremos "dominio 1", y una segunda parte que lo hará con el *VGA_CLK* a la que nos referiremos como "dominio 2". La sincronización de ambas partes se hará mediante el módulo FIFO, que representa una cola FIFO capaz de ser escrita a una velocidad y leída a otra distinta.

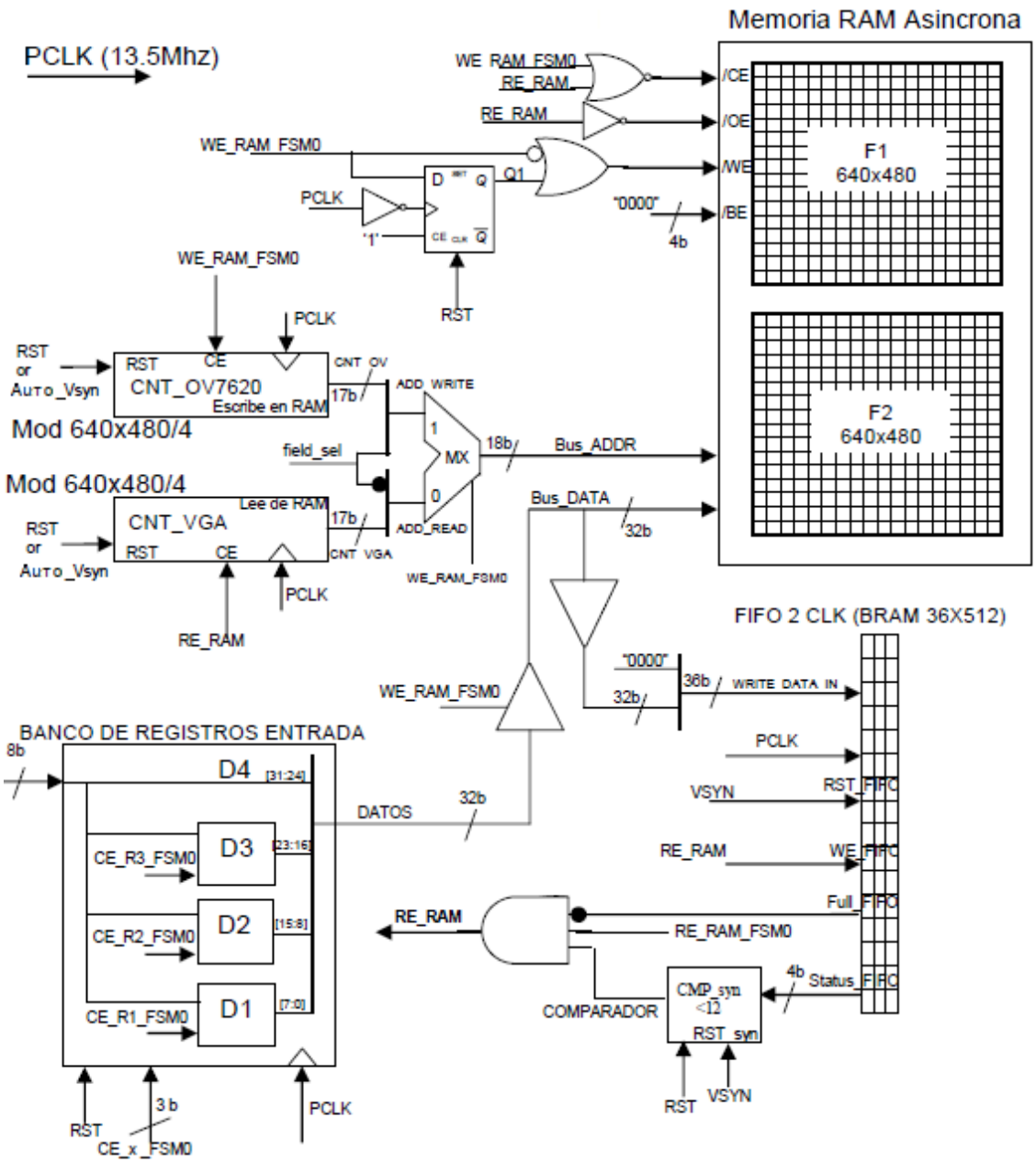


Figura 62. Arquitectura general. Zona 1. Funcionamiento con PCLK

El dominio 1 constituye principalmente la entrada y almacenamiento de datos en la arquitectura. En un principio los datos de 8 bits proporcionados por la cámara son almacenados en el Banco de Registros de Entrada, cuya función es la de concatenar estos datos para crear datos de 4 bytes, ya que el módulo de memoria trabaja con datos de este tamaño. Una vez que lo tenemos construido, el dato de 4 bytes es llevado a una memoria SRAM Asíncrona y almacenado en la posición que indique su bus de direcciones. Esta memoria es la encargada de solucionar el problema de la adaptación de frecuencias. Para ello dividimos la memoria en dos partes, de modo que podamos almacenar dos frames consecutivos en distintas zonas de memoria. Así, mientras guardamos el segundo frame que llega en una zona, leeremos dos veces el frame ya almacenado en la otra zona de memoria. De esta forma duplicamos la frecuencia de funcionamiento de la arquitectura general de 30 Hz a 60 Hz. Las direcciones de lectura y escritura de la memoria son proporcionadas por los contadores de lectura y escritura, a

cuya señal de cuenta se le concatena un bit que permite hacer la citada división de la memoria

Cuando procede el dato es leído de la memoria y almacenado en la cola FIFO. Ésta es el módulo encargado de sincronizar las dos zonas de la arquitectura. En ella se escribirá utilizando como señal de reloj la *PCLK* y se leerá empleando la señal *VGA_CLK*.

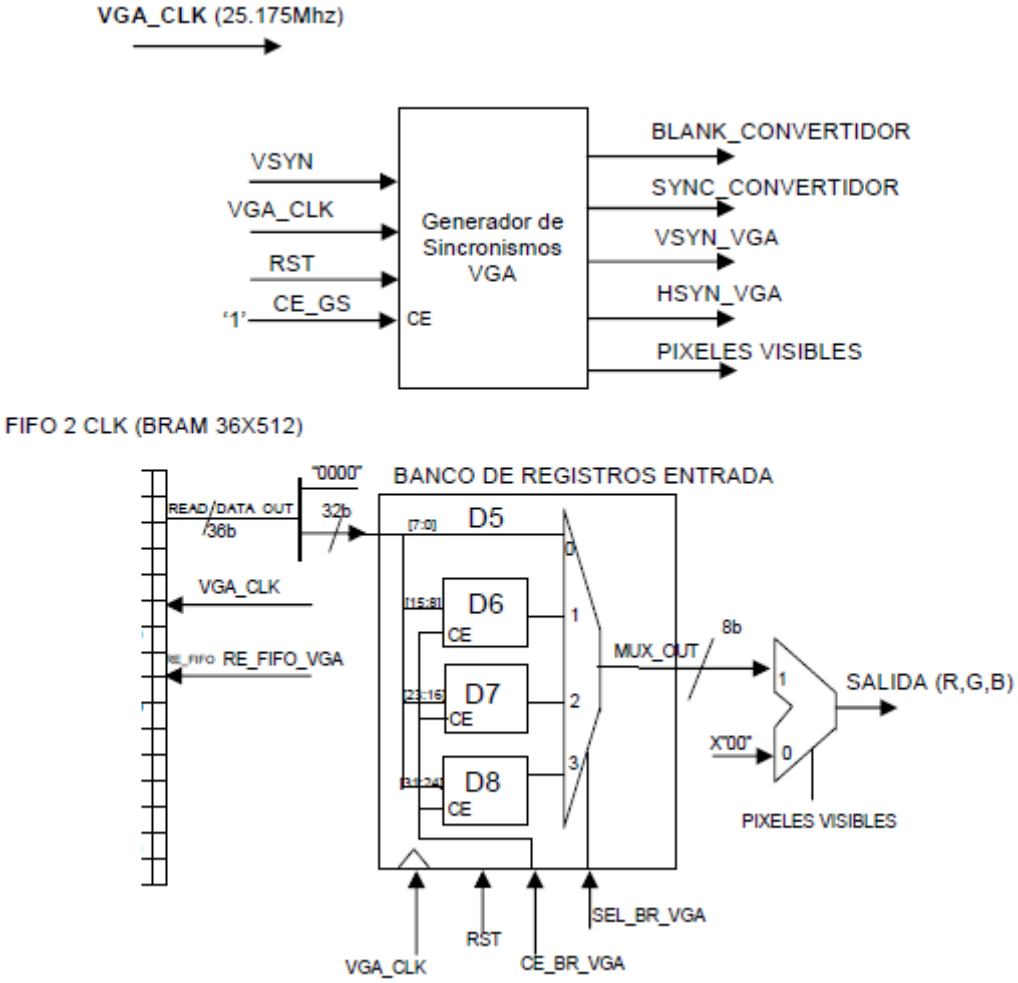


Figura 63. Arquitectura general. Zona 2. Funcionamiento con VGA_CLK.

El dominio 2 de la arquitectura representa la salida de la información en el estándar VGA. Por tanto, debe proporcionar las señales de sincronismo en este estándar y la información RGB. Las señales de sincronismo en estándar VGA son creadas por el módulo Generador de Sincronismos VGA a partir de la señal de sincronismo vertical *VSYN*. En cuanto a la salida de información, el dato almacenado en la FIFO es extraído de ella y almacenado en el Banco de Registros de Salida. La función de este módulo es deshacer la concatenación del Banco de Registros de Entrada y sacar la información en datos de 1 byte. Al final de la arquitectura tenemos un multiplexor que saca el dato de información por la salida *RGB* sólo cuando se trate de un píxel visible. De este modo, resolvemos el problema de la resolución, ya que tanto la cámara como el monitor trabajan con la misma información visible dentro de un frame (640x480 píxeles), por lo que bastará con no tener en cuenta las líneas y píxeles no visibles.

Por último, debemos comentar que la sincronización de toda la arquitectura se lleva a cabo mediante dos máquinas de estado, la FSM0 y la FSM1. La FSM0 generará

las señales que van a controlar todos los componentes del dominio 1, mientras que la FSM1 creará las señales de control del dominio 2.

3.3 DESCRIPCIÓN DE MÓDULOS

En este apartado vamos a describir cada uno de los módulos que componen la arquitectura general. Lo haremos de manera particular, de modo que para cada uno de ellos explicaremos su arquitectura interna, funcionamiento y situación en la arquitectura.

3.3.1 GENERADOR DE SINCRONISMOS PARA ESTÁNDAR VGA.

3.3.1.1 Utilidad y funcionalidad

Dentro de la arquitectura general del proyecto, el módulo generador de sincronismos adquiere una gran importancia, puesto que se encarga de generar las señales de sincronismo de un monitor compatible con el estándar VGA.

Antes de realizar un análisis en profundidad este módulo, debemos tener claros algunos conceptos que nos ayudarán a comprenderlo:

- La cámara OV7620 proporciona unos sincronismos cuya señal de sincronismo vertical se repite con una frecuencia de 30 Hz.
- El monitor utiliza el estándar VGA, que emplea unos sincronismos cuya señal de sincronismo vertical se repita con una frecuencia de 60 Hz.
- Por último, conviene repasar la forma y duración exactas de las señales de sincronismo, tanto de la cámara (Capitulo 2) como del estándar VGA (Capítulo 3).

El objetivo de este módulo es generar a partir del sincronismo vertical de la cámara (*VSYN*) los sincronismos vertical y horizontal del estándar VGA. Por tanto, generaremos dos sincronismos verticales en estándar VGA (con sus correspondientes sincronismos horizontales) por cada sincronismo vertical procedente de la cámara. La figura 64 muestra de manera muy simplificada el funcionamiento de este módulo:

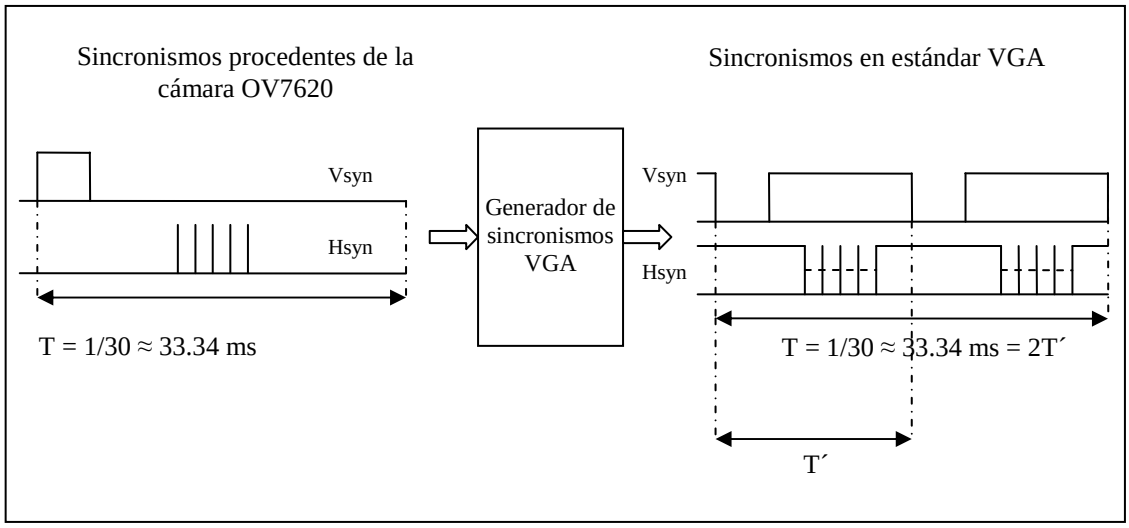


Figura 64. Esquema simplificado del funcionamiento del Generador de Sincronismos para estándar VGA

La figura 65 y la Tabla 7 muestran la disposición y descripción de las entradas y salidas del generador de sincronismos. Este módulo permanecerá activo todo el tiempo (señal de habilitación siempre en alta) esperando la llegada de un pulso en su entrada *VSYN*. Cuando este pulso se produce, comienzan a generarse en la salida los sincronismos horizontales y verticales correspondientes a dos frames completos en estándar VGA. Una vez generados, el módulo permanece en estado de espera ante la llegada de un nuevo pulso en la entrada *VSYN*.

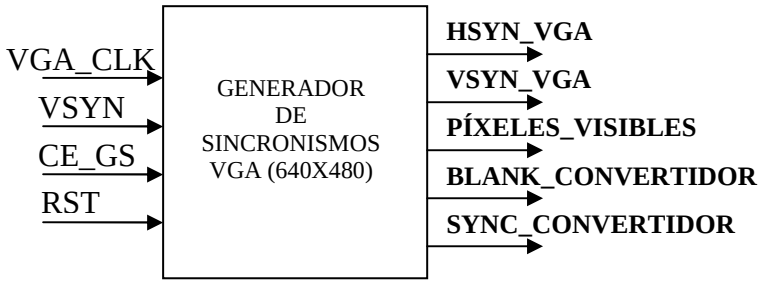


Figura 65. Generador de Sincronismos para estándar VGA

Nombre del Puerto	Dirección	Descripción
VSYN	Entrada	Señal de sincronismo vertical procedente del adaptador de sincronismos verticales
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
RST	Entrada	Señal de reset asíncrono del circuito
CE_GS	Entrada	Señal de activación del módulo. Estará siempre activa, es decir, a nivel alto
VSYN_VGA	Salida	Señal de sincronismo vertical del monitor compatible con el estándar VGA (Ver Capítulo 3.- Monitor VGA)
HSYN_VGA	Salida	Señal de sincronismo horizontal del monitor compatible con el estándar VGA (Ver Capítulo 3.- Monitor VGA)
BLANK_CONVERTIDOR	Salida	Señal de control del convertidor ADV7125
SYNC_CONVERTIDOR	Salida	Señal de control del convertidor ADV7125
PIXELES_VISIBLES	Salida	Señal que indica todos los píxeles visibles de un frame de VGA.

Tabla 7. Descripción señales de entrada y salida del Generador de Sincronismos

3.3.1.2 Situación del módulo en la arquitectura

El generador de sincronismos está conectado a la cámara, que le suministrará la señal de entrada *VSYN*. A partir de ella, el módulo generará las señales *HSYN_VGA*, *VSYN_VGA*, *BLANK_CONVERTIDOR* y *SYNC_CONVERTIDOR* que son llevadas al convertidor ADV7125 de la FPGA. Este módulo las convertirá en señales analógicas y las introducirá en el conector VGA de entrada al monitor. El generador de sincronismos proporciona también a su salida la señal *PIXELES_VISIBLES*, que se conectará al multiplexor de salida de información RGB como señal de control, proporcionando una salida con información cuando esté en alta y todo ceros cuando esté en baja.

3.3.1.3 Arquitectura interna

En referencia a la implementación interna del generador de sincronismos, hemos de saber que está compuesta por varios submódulos. Éstos han sido organizados de manera muy intuitiva, como se muestra en la Figura 66. Disponemos de un submódulo de entrada que activará un submódulo de control. Este último controlará la activación de los otros dos submódulos, cuya tarea consistirá en obtener las señales de sincronismo horizontal y vertical.

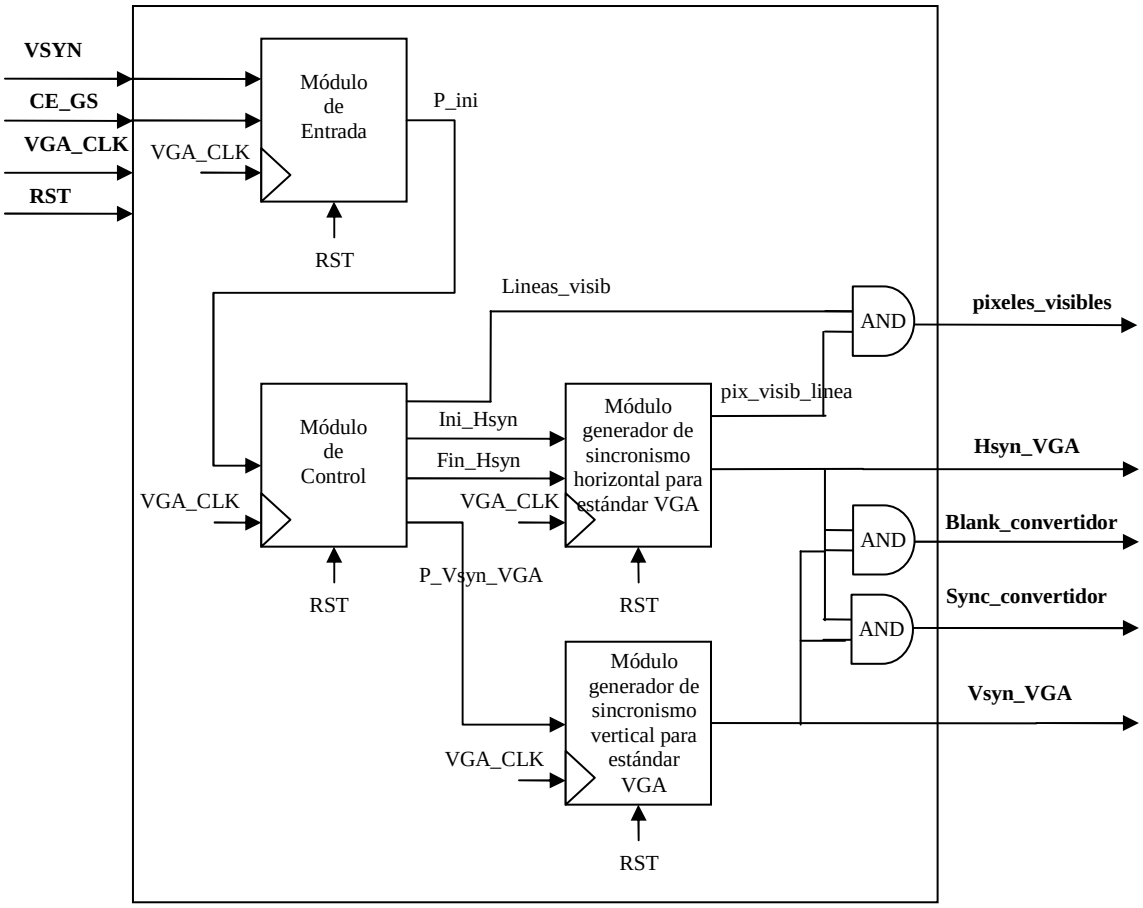


Figura 66. Esquema interno del Generador de Sincronismos para estándar VGA

La arquitectura interna totalmente detallada del Generador de Sincronismos se encuentra en el "Anexo C" de la memoria. Además, han sido incluidos un cronograma generalizado de su funcionamiento en el "Anexo D" y un cronograma con todas las señales en el "Anexos E". Antes de analizarlos, conviene realizar una descripción previa de cada uno de los submódulos que forman la arquitectura interna.

Submódulo de entrada

La función de este submódulo es la de sincronizar la señal de entrada *VSYN* con la señal de reloj *VGA_CLK*, y generar el pulso de salida que pondrá en funcionamiento el resto de submódulos.

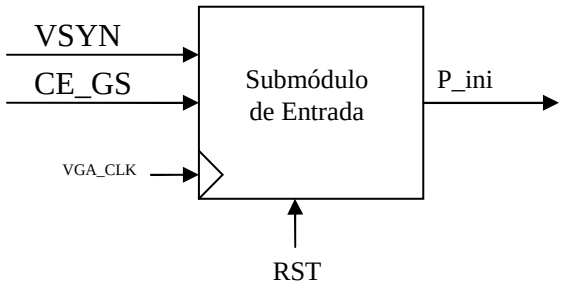


Figura 67. Submódulo de Entrada

Nombre del Puerto	Dirección	Descripción
VSYN	Entrada	Señal de sincronismo vertical procedente del adaptador de sincronismos verticales
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
RST	Entrada	Señal de reset asíncrono del circuito
CE_GS	Entrada	Señal de activación del módulo. Estará siempre activa, es decir, a nivel alto
P_ini	Salida	Señal de salida que se empleará para activar el módulo de control. Sólo se pondrá en alta durante un ciclo.

Tabla 8. Descripción de las señales de entrada y salida del submódulo de entrada

En la figura 68 se representa la arquitectura interna de este submódulo. Está compuesta por un componente para la sincronización de señales asíncronas, que sincronizará la entrada *VSYN* con la señal de reloj *VGA_CLK*; y por una máquina de estados que controla la aparición de un pulso a la salida *P_ini* que pondrá en funcionamiento el módulo de control.

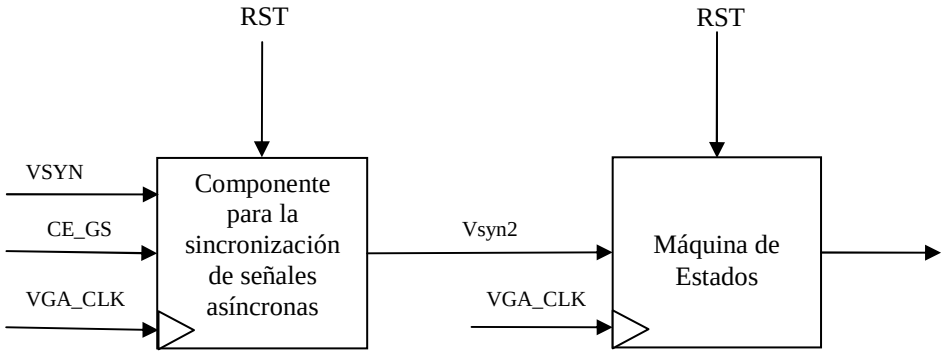


Figura 68. Esquema interno del submódulo de entrada. Sus componentes se explican en los apartados siguientes

1) Componente para la sincronización de señales asíncronas

Su función es la de sincronizar la señal de la entrada *VSYN* con la señal de reloj del circuito *VGA_CLK*. Para ello, tal como queda reflejado en la figura 69, se emplean 2 flip-flops tipo D dispuestos en serie.

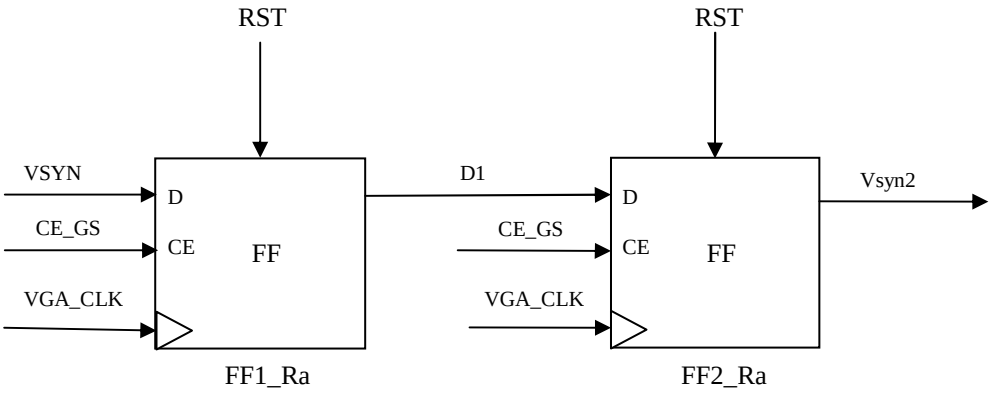


Figura 69. Componente para la sincronización de señales asíncronas formado por dos flip-flops tipo D en serie

2) Máquina de estados

Se trata de una maquina de Moore, puesto que el valor de la salida en cada instante es función únicamente del estado en el que se encuentra. A través de ella controlamos la aparición de un pulso a la salida *P_ini* de un ciclo de duración, que estará conectado a la entrada de activación del submódulo de control.

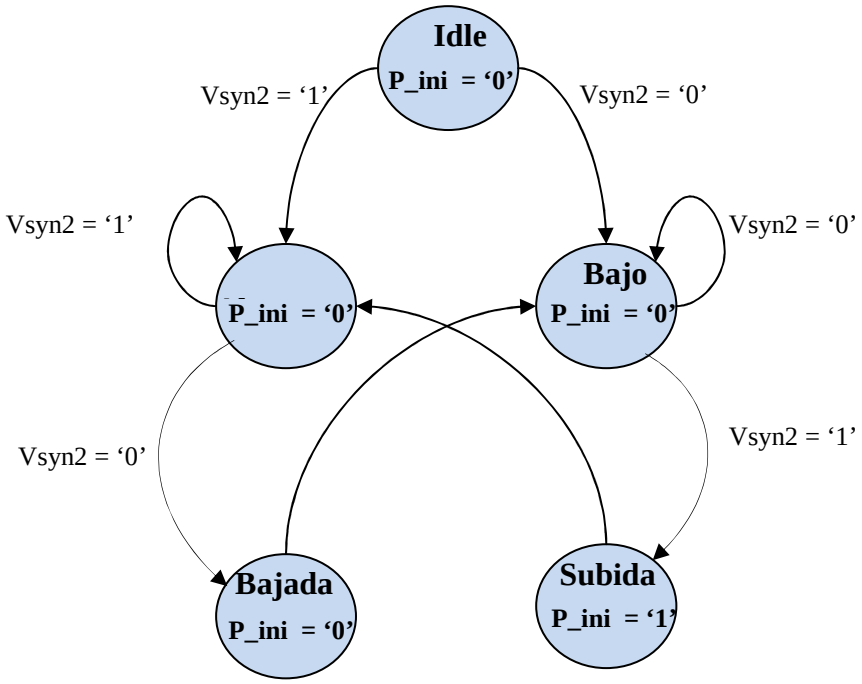


Figura 70. Diagrama de Estados

Analizando la figura 70 observamos como la salida *P_ini* se pone en alta cuando hay una transición de baja a alta en la señal de entrada *VSYN*(de estado Bajo a estado Subida), adquiriendo un valor bajo en el ciclo inmediatamente posterior (estado Alto), por lo que este pulso durará siempre un ciclo de reloj.

La máquina de estados ha sido implementada en VHDL utilizando de 3 procesos, dos combinacionales y uno secuencial. Los procesos combinacionales se utilizan, uno para calcular el estado siguiente en función del estado actual y la señal de entrada, y el otro, para obtener el valor de la salida en función del estado actual; mientras que el proceso secuencial es utilizado para registrar tanto el estado actual como la salida *P_ini*.

Submódulo de control

Su función es la de crear los pulsos que van a controlar los submódulos generadores de los sincronismos horizontal y vertical. Este módulo se activa con la llegada de un pulso en la entrada P_ini. Sus salidas Ini_Hsyn y Fin_Hsyn son las señales de Inicio y Fin que activan y desactivan el módulo generador de los sincronismos horizontales. En cuanto a su salida P_Vsyn_VGA, se utiliza para poner en funcionamiento el módulo generador de sincronismos verticales.

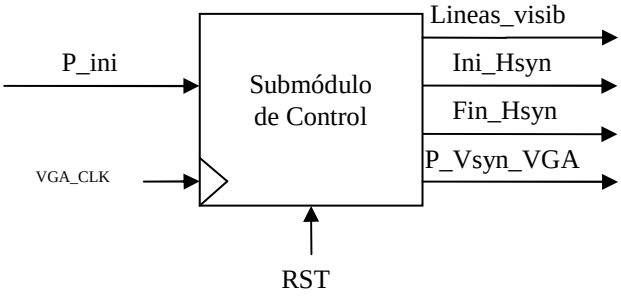


Figura 71. Submódulo de Control

Nombre del Puerto	Dirección	Descripción
P_ini	Entrada	Señal de activación (en alta) del módulo de control. Sólo se pondrá en alta durante un ciclo.
RST	Entrada	Señal de reset asíncrono del circuito.
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
Ini_Hsyn	Salida	Señal que al ponerse en alta, activa el módulo generador de sincronismos horizontales. Sólo se activará durante un ciclo.
Fin_Hsyn	Salida	Señal que al ponerse en alta, desactiva el módulo generador de sincronismos horizontales. Sólo se activará durante un ciclo.
P_Vsyn_VGA	Salida	Señal que al ponerse en alta, activa el módulo generador de sincronismos verticales. Sólo se activará durante un ciclo.
Lineas_visib	Salida	Señal que indica las líneas visibles dentro de

		un frame de VGA. Cuando la línea sea visible tomará un valor alto y uno bajo cuando no lo sea.

Tabla 9. Descripción de las señales de entrada y salida del submódulo de control

La arquitectura interna de este submódulo se representa en la figura 72. Está compuesta por un módulo denominado de Inicio/Fin, un contador y cinco comparadores. Este contador es el encargado de contar todos los píxeles de dos frames de salida VGA, es decir, 2x525x800 píxeles (ciclos de reloj), por lo que será un contador de 20 bits*.

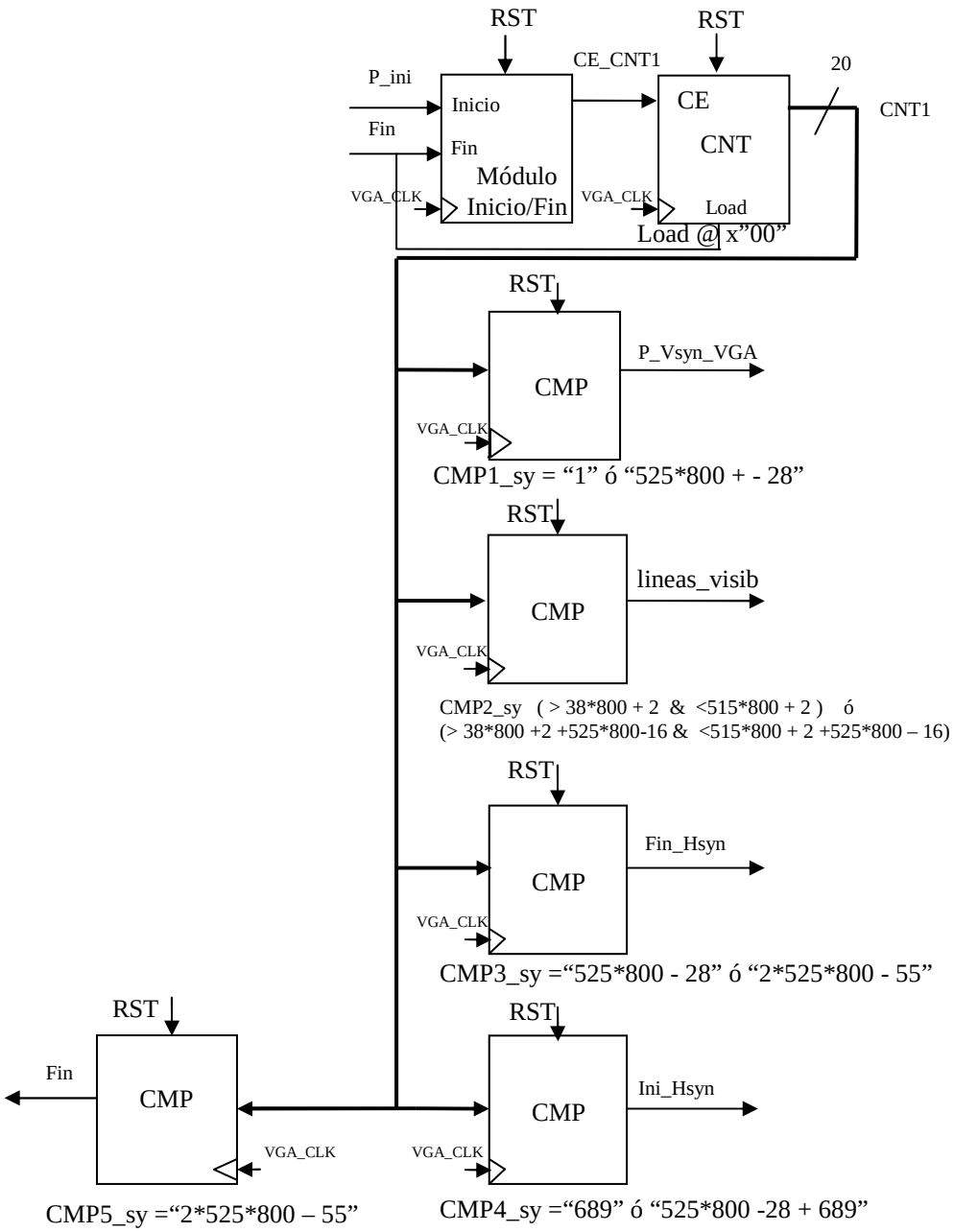


Figura 72. Esquema interno del submódulo de control.

* $N\ bits = ceil(\ln_2(2x525x800)) = ceil(19.68) = 20\ bits$

El funcionamiento es el siguiente: un pulso en la entrada *P_ini* provoca que el módulo Inicio/Fin ponga su salida en alta hasta que se active su señal de entrada *Fin*. Esta salida está conectada a la señal de activación del contador *CE_CNT1*, por lo que al ponerse en alta el contador comenzará su cuenta. A su vez la señal de cuenta *CNT1* será la entrada de todos los comparadores, que la compararán con sus genéricos para activar o no sus salidas. Finalmente, la salida del quinto comparador será la entrada *Fin* del módulo Inicio/Fin y la entrada de carga del contador, por lo que cuando se ponga en alta, se desactivará el contador y se cargará en su señal de cuenta el valor 0.

A continuación se presenta una descripción de los componentes utilizados. Se debe tener en cuenta que los submódulos generadores de los sincronismos horizontales y verticales (Apartado 3.3 y 3.4) van a tener una arquitectura muy similar a la de este submódulo, y van a hacer uso de estos mismos componentes.

1) Módulo Inicio/Fin

Este módulo pone en alta su salida ante un pulso en la señal de entrada *Inicio*, manteniéndola en este estado hasta que se produzca un pulso en la señal de entrada *Fin*. Por tanto, se trata de un conmutador con dos posiciones, la posición "on" en la que la salida está activa y la posición "of" en la que está a nivel bajo.

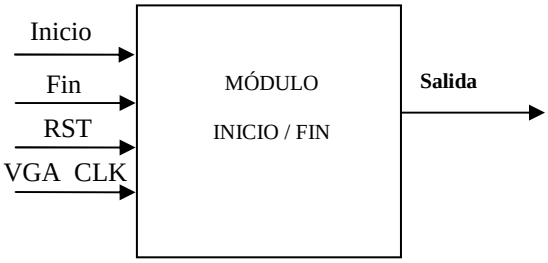


Figura 73. Módulo Inicio/Fin

Nombre del Puerto	Dirección	Descripción
Inicio	Entrada	Señal que al activarse pone en alta la salida, siempre y cuando la entrada Fin esté en baja
Fin	Entrada	Señal que al activarse pone en baja la salida, independientemente del valor de la entrada Inicio.
RST	Entrada	Señal de reset asíncrono del circuito.
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
Salida	Salida	Señal de salida del circuito. Tomará un valor alto o bajo dependiendo de la combinación de las entradas. También será utilizada como entrada de realimentación para determinar la salida en el ciclo siguiente.

Tabla 10. Descripción de las señales de entrada y salida del submódulo Inicio/Fin.

En la figura 74podemos ver cómo está compuesto por una LUT (lookup table) de tres entradas para implementar la función lógica de salida, y un flip-flop tipo D para registrarla.

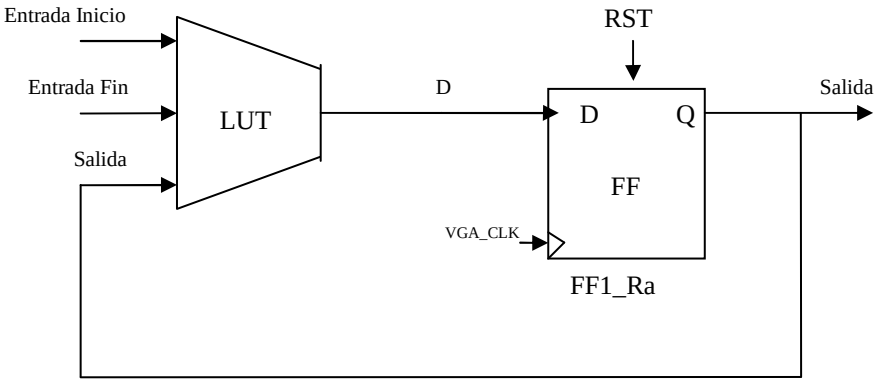


Figura 74. Esquema interno del módulo Inicio/Fin.

La tabla 11 muestra la configuración de la LUT.

ENTRADAS			**SALIDA LUT**
Entrada Inicio	*Entrada Fin*	*Salida*	*D*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Tabla 11. Tabla de verdad de la LUT del módulo Inicio/Fin

Como hemos comentado anteriormente la salida de la LUT y del circuito (tras ser registrada) se pondrá en alta siempre que la entrada *Inicio* esté en alta y la entrada *Fin* en baja; y cuando, estando ambas entradas en baja, la entrada *Salida* (representa la salida anterior del módulo) esté activa. Además, notamos que siempre que la entrada *Fin* esté activa, la salida de la LUT será baja, independientemente del resto de entradas.

2) Contador

Todos los contadores que utilizaremos en el generador de sincronismos serán síncronos ascendentes de un número determinado de bits*[1], con reset asíncrono y entradas de carga y habilitación síncronas. Una vez activa la señal de habilitación (*CE*) la señal de cuenta del contador comienza a contar de forma ascendente tomando como primer valor de cuenta el 0. La activación de la señal de carga provoca que la señal de cuenta tome el valor 0 en el siguiente ciclo.

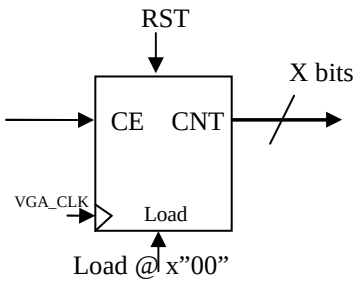


Figura 75. Contador ascendente con reset asíncrono y entradas de carga y habilitación síncronas

Notas:

1. El número de bits dependerá de la cantidad de ciclos de reloj que se quiera contar y teniendo en cuenta que cada píxel de la imagen tiene una duración de un ciclo, se determinará específicamente en cada submódulo. Se calcula aplicando la siguiente ecuación:

$$x\,bits = ceil(\ln_2(n^o\,ciclos))$$

siendo ceil (x) la función que devuelve el entero superior de x.

3) Comparadores

Los comparadores utilizados son síncronos y con reset asíncrono. Además, como señal de entrada tendrán la salida del contador, que contrastarán con sus genéricos de la manera que corresponda para obtener la salida.

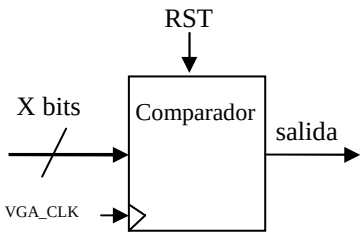


Figura 76. Comparador síncrono con réset asíncrono

Submódulo generador de sincronismos Vertical

Se encarga de generar la señal de sincronismo vertical para el estándar VGA. Un pulso en su entrada *P_Vsyn_VGA* pondrá en funcionamiento el módulo, generando en su salida dicho sincronismo.

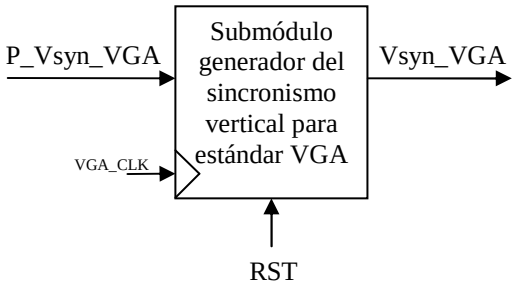


Figura 77. Módulo generador del sincronismo vertical para estándar VGA

Nombre del Puerto	Dirección	Descripción
P_Vsyn_VGA	Entrada	Señal que al ponerse en alta, activa el módulo generador de sincronismos verticales. Sólo se activará durante un ciclo.
RST	Entrada	Señal de reset asíncrono del circuito.
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
Vsyn_VGA	Salida	Señal de sincronismo vertical del estándar VGA

Tabla 12. Descripción de las señales de entrada y salida del generador del sincronismo vertical para estándar VGA.

La Figura 78 muestra la arquitectura interna de este submódulo. Está formado por un módulo Inicio/Fin, un contador y un comparador. Este contador es el encargado de contar 2 líneas (duración en alta de la señal de sincronismo vertical), es decir, 2*800 píxeles, por lo que será un contador de 11 bits*.

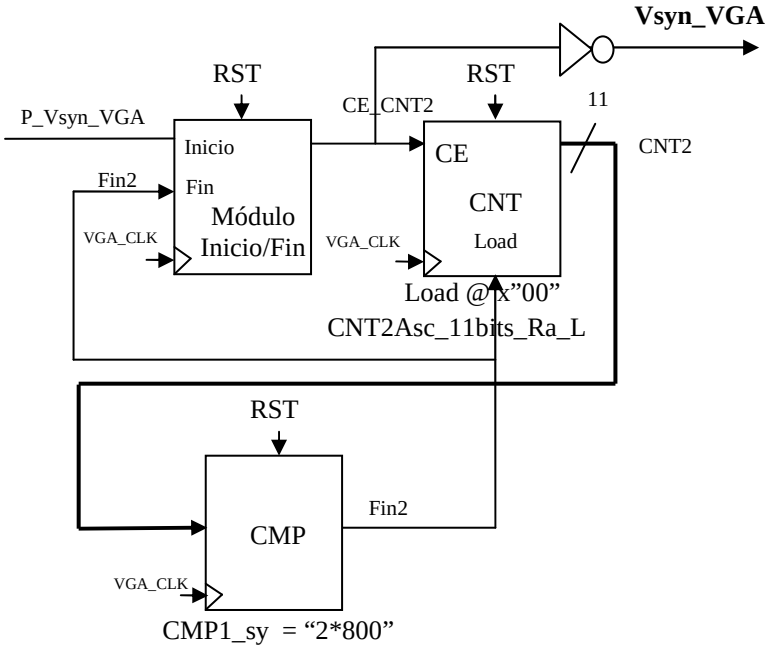


Figura 78. Esquema interno del módulo generador del sincronismo vertical para estándar VGA

* $N\ bits = ceil(\ln_2(2x800)) = ceil(10.64) = 11\ bits$

En referencia al funcionamiento de este módulo, comprobamos como es similar al del módulo de control, salvo que la señal de salida no será la salida de un comparador, sino la inversa de la señal de activación del contador. Por ello, un pulso en la entrada *P_Vsyn_VGA* provoca que el módulo Inicio/Fin ponga su salida en alta hasta que se active su señal de entrada *Fin*. Esta salida será la señal de activación del contador, por lo que al ponerse en alta el contador comenzará su cuenta. A su vez la señal de cuenta será la entrada del comparador, que la comparará con su genérico para activar o no su salida *Fin2*. Notamos como esta salida del comparador supone la entrada *Fin* del módulo Inicio/Fin y la entrada de carga del contador, de manera que cuando se ponga en alta desactivará el contador y cargará en su señal de cuenta el valor 0.

Submódulo generador de sincronismos Horizontal

Su función es la de generar la señal de sincronismo horizontal en el estándar VGA, así como una señal auxiliar que será utilizada para conseguir una señal que indique los píxeles visibles de un frame de VGA.

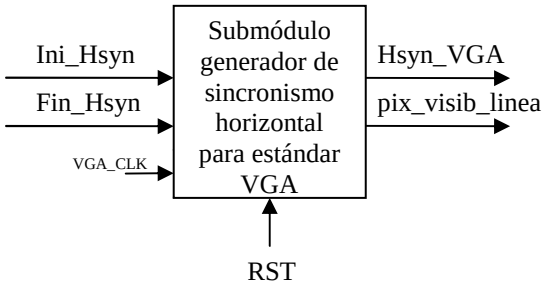


Figura 79. Módulo generador del sincronismo horizontal para estándar VGA

Nombre del Puerto	Dirección	Descripción
Ini_Hsyn	Entrada	Señal que al ponerse en alta, pone en funcionamiento este módulo. Sólo se activará durante un ciclo.
Fin_Hsyn	Entrada	Señal que al ponerse en alta, desactiva este módulo. Sólo se activará durante un ciclo.
RST	Entrada	Señal de reset asíncrono del circuito.
VGA_CLK	Entrada	Señal de reloj con la que trabaja el monitor. Su frecuencia nominal es de 25.175 Mhz.
Hsyn_VGA	Salida	Señal de sincronismo horizontal del estándar VGA.
pix_visib_linea	Salida	Señal que indica los píxeles visibles dentro de una línea. Cuando los píxeles sean visibles tomará un valor alto y un valor bajo cuando no lo sea.

Tabla 13. Descripción de las señales de entrada y salida del generador del sincronismo horizontal para estándar VGA

La arquitectura interna de este submódulo viene dada por la figura 80. Consta de un módulo Inicio/Fin, un contador y dos comparadores. En cuanto al contador se refiere, es el encargado de contar todos los píxeles de una línea, es decir, 800 píxeles, por lo que será un contador de 10 bits*.

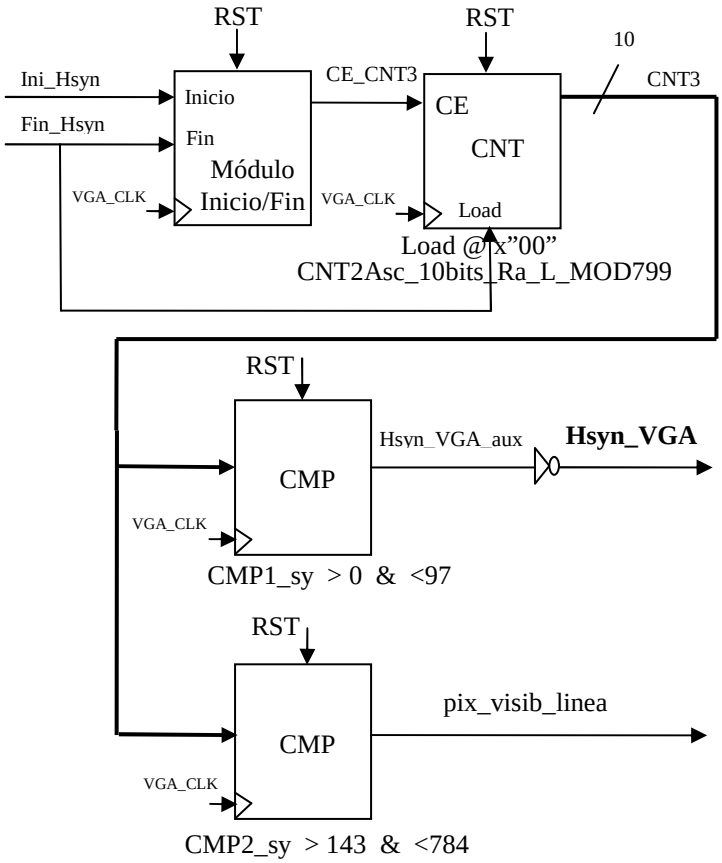


Figura 80. Esquema interno del módulo generador del sincronismo horizontal para estándar VGA.

El funcionamiento es similar al del módulo de control, salvo que sólo tenemos dos comparadores y que la señal de entrada *Fin* del módulo Inicio/Fin viene dada como entrada al módulo generador. Por tanto, un pulso en la entrada *Ini_Hsyn* provoca que el módulo Inicio/Fin ponga su salida en alta hasta que se active su señal de *fin*. Esta salida es la señal de activación del contador, por lo que al ponerse en alta el contador comenzará su cuenta. A su vez la señal de cuenta será la entrada de los dos comparadores que la compararán con sus genéricos para activar o no sus salidas. Finalmente, cuando se active la entrada *Fin_Hsyn*, se activarán la entrada *Fin* del módulo Inicio/Fin y la entrada de carga del contador, desactivándose el mismo y cargándose en su señal de cuenta el valor 0.

* $N\ bits = ceil(\ln_2(800)) = ceil(9.64) = 10\ bits$

Resto de componentes

Por último, notamos como la generación de las restantes señales de salida del generador de sincronismos en estándar VGA, se consigue mediante puertas AND, multiplicando las señales que correspondan.

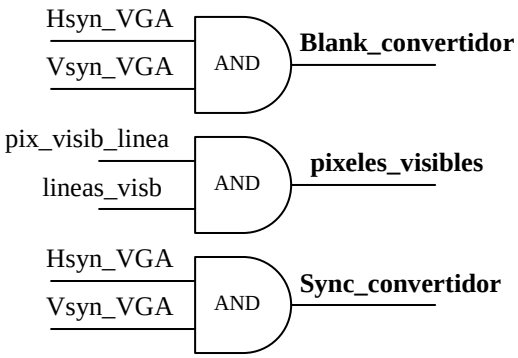


Figura 81. Generación de las señales Blank_Convertidor, Sync_convertidor y pixeles_visibles mediante puertas AND.

Nota: la señal *píxeles_visibles* muestra los píxeles visibles dentro de un frame de VGA. Se genera mediante la operación AND entre las señales *lineas_visib* y *Pix_visib_linea*. A continuación mostramos la forma que adquirirá dicha señal:

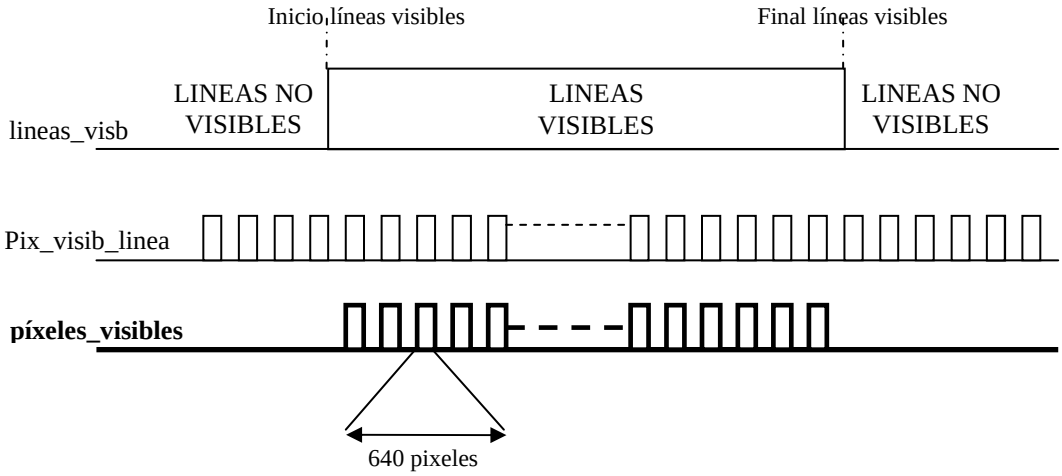


Figura 82. Generación de la señal *pixeles_visibles.*

3.3.2 MEMORIA SRAM ASÍNCRONA

SRAM representa la abreviatura de Static Random Access Memory (Memoria Estática de Acceso Alaeatorio). El hecho de ser estática quiere decir que no es necesario refrescar los datos, ya que sus celdas están formadas por flip-flops de transistores bipolares que mantienen el dato siempre y cuando estén alimentadas. El acceso aleatorio significa que la localización de las posiciones donde los datos son leídos o escritos no sigue ningún orden. Entre sus ventajas, destaca la velocidad en el acceso a los datos.

Este tipo de memorias se divide en dos tipos: síncronas y asíncronas. La memoria que hemos utilizado en nuestro proyecto es asíncrona, lo que indica que es independiente de la frecuencia de reloj, estando los datos de entrada y de salida controlados por la transición de las direcciones.

3.3.2.1 Utilidad y funcionalidad

La memoria SRAM Asíncrona es el elemento principal sobre el que gira toda la arquitectura. Gracias a este módulo es posible pasar de los 30 hz con los que funciona la cámara OV7620 a los 60 hz del monitor.

El hecho de que la cámara OV7620 funcione a 30 hz indica que se obtienen 30 frames de información por segundo, mientras que el monitor pintará 60 frames en ese segundo. La memoria es utilizada de manera que, cada frame procedente de la cámara que se almacene en ella, sea leído dos veces seguidas por el monitor, consiguiendo así duplicar la frecuencia de funcionamiento (de 30 a 60 hz). Por tanto, lo que hacemos para duplicar la frecuencia es pintar dos veces en el monitor la misma imagen procedente de la cámara.

3.3.2.2 Características

La memoria utilizada en nuestra arquitectura es una SRAM Asíncrona de datos de 32 bits y 32M Byte de capacidad de almacenamiento. Está formada por dos dispositivos SRAM Asíncrona, del tipo CY7C1041CV33, dispuestos en paralelo.

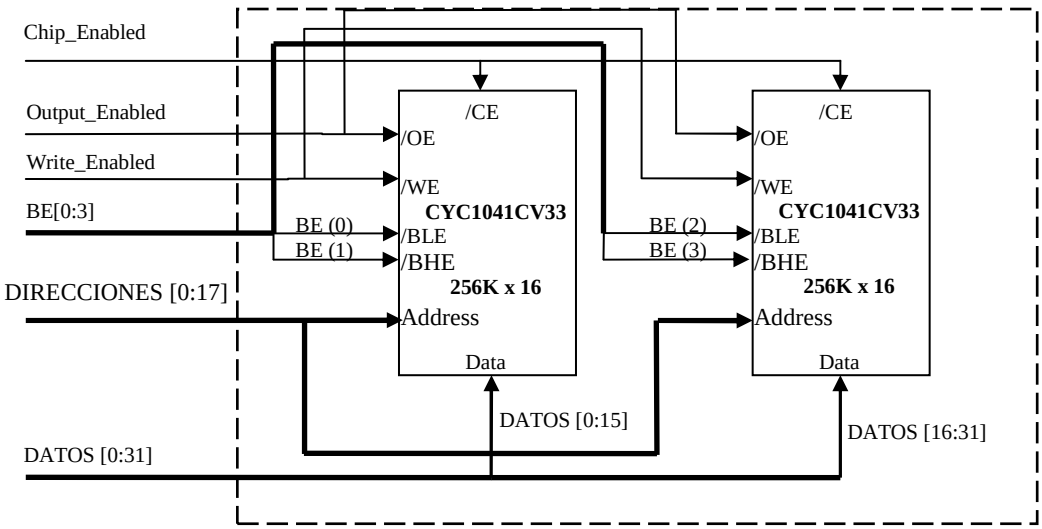


Figura 83. SRAM Asíncrona de tamaño 256x32 construida a partir de dos CYC1041CV33 en paralelo

A continuación profundizamos en las características del dispositivo CY7C1041CV33. Se trata de una memoria SRAM estática en tecnología CMOS, organizada en 256x1024 posiciones de 16 bits. Su velocidad máxima de operación la determina el tiempo que debe transcurrir desde que se produce un cambio en el bus de direcciones hasta que el dato es válido, siendo de 10 ns.

La figura 84 muestra el diagrama de bloques lógicos que implementan la memoria y la disposición de los pines de la misma.

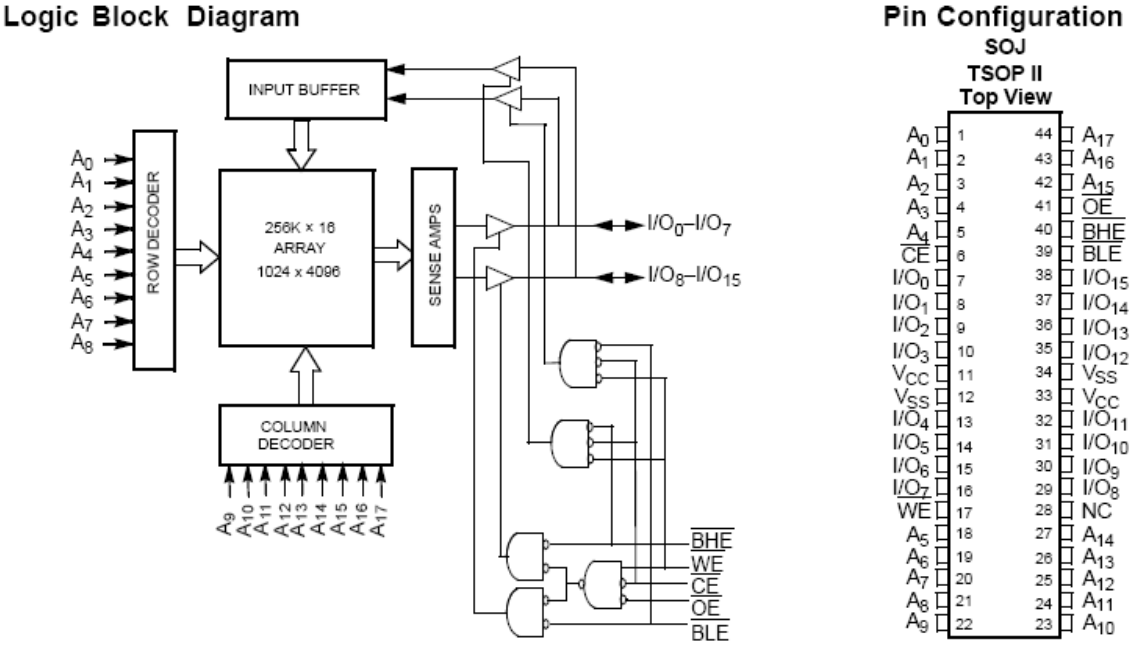


Figura 84. A la izquierda: Diagrama de bloques lógicos de la memoria CY7C1041CV33. A la derecha: Distribución de los pines del chip.

Para escribir en el dispositivo las entradas *Chip Enable* (/CE) y *Write Enable* (/WE) deben estar en baja. Si la entrada *Byte Low Enable* (/BLE) está en baja, los 8 bits menos significativos del bus de datos (pines I/O_0 al I/O_7) se escriben en la posición especificada por el bus de direcciones (pines A_0 al A_{17}). Si la entrada *Byte High Enable* (/BHE) está en baja, los 8 bits más significativos del bus de datos (pines I/O_8 al I/O_{15}) se escriben en la posición especificada por el bus de direcciones (pines A_0 al A_{17}).

Para leer de la memoria las entradas *Chip Enable* (/CE) y *Output Enable* (/OE) deben estar en baja, mientras que la entrada *Write Enable* (/WE) tiene que ser forzada a un valor alto. Si la entrada *Byte Low Enable* (/BLE) está en baja, el dato de la posición de memoria especificada por el bus de direcciones aparecerá en los 8 bits menos significativos del bus de datos (pines I/O_0 al I/O_7). Si la entrada *Byte High Enable* (/BHE) está en baja, el dato se obtiene los pines I/O_8 al I/O_{15}. Ver la tabla de verdad (Tabla 16) al final de este apartado para una descripción completa de los modos de lectura y escritura.

Todos los pines del bus de datos (I/O_0 al I/O_{15}) se pondrán en alta impedancia cuando el dispositivo esté desactivado (/CE en alta), las salidas estén deshabilitadas (/OE en alta), las entradas /BLE y /BHE estén deshabilitadas (ambas en alta), o durante la operación de Escritura (/CE y /WE en baja).

La tabla 14 presenta la definición de los pines:

Nombre del Pin	**Tipo de Entrada/Salida**	**Descripción**
$A_0 - A_{17}$	Entrada	Bus de direcciones utilizado para seleccionar una determinada posición en memoria.
$I/O_0 - I/O_{15}$	Entrada/Salida	Bus de datos bidireccional. Usado como bus entrada o salida dependiendo de la operación.
NC	No Conectado	Pin no conectado.
/WE	Entrada/Control	Entrada de habilitación de Escritura, activa en baja. En baja permite la Escritura, en alta permite la Lectura.
/CE	Entrada/Control	Entrada de habilitación del chip, activa en baja. En baja habilita el chip, en alta deshabilita el chip.
/BLE, /BHE	Entrada/Control	Entrada de selección de byte, activa en baja. /BLE controla los pines $I/O_0 - I/O_7$, /BHE controla los pines $I/O_8 - I/O_{15}$.
/OE	Entrada/Control	Habilitación de salida, activa en baja. Controla la dirección de los pines de Entrada/Salida (I/O). Cuando está en baja los pines I/O se comportan como salidas y cuando está en alta como entradas.
V_{SS}	Masa	Masa del dispositivo. Deberá estar conectada a la masa del sistema.
V_{CC}	Alimentación	Entrada de alimentación del chip.

Tabla 14. Definición de los pines del chip CY7C1041CV33

Los modos de funcionamiento (Lectura y Escritura) de la memoria quedan de manifiesto a partir de sus tiempos (Tabla 15) y cronogramas (figuras 85, 86, 87, 88 y 89).

Parámetro	Descripción	-10		Unidad
		Min.	Max.	
Ciclo de Lectura				
tpower	V_{cc} del primer acceso	1		µs
tRC	Tiempo del ciclo de Lectura	10		ns
tAA	Dirección hasta dato válido		10	ns
tOHA	Datos estables hasta cambio de dirección	3		ns
tACE	/CE baja hasta dato válido		10	ns
tDOE	OE baja hasta dato válido		5	ns
tLZOE	/OE baja hasta baja impedancia	0		ns
tHZOE	/OE alta hasta alta impedancia		5	ns
tLZCE	CE baja hasta baja impedancia	3		ns
tHZCE	/CE alta hasta alta impedancia		5	ns
tPU	/CE baja hasta Alimentación alta	0		ns
tPD	CE alta hasta Alimentación baja		10	ns
tDBE	Byte habilitado hasta dato válido		5	ns
tLZBE	Byte habilitado hasta baja impedancia	0		ns

tHZBE	Byte habilitado hasta alta impedancia		6	ns
Ciclo de Escritura				
tWC	Tiempo del ciclo de Escritura	10		ns
tSCE	CE baja hasta Fin de Escritura	7		ns
tAW	Dirección establecida hasta fin de Escritura	7		ns
tHA	Fin de Escritura hasta dirección establecida	0		ns
tSA	Dirección establecida hasta comienzo de Escritura	0		ns
tPWE	Anchura del pulso /WE	7		ns
tSD	Dato estable hasta fin de Escritura	5		ns
tHD	Fin de Escritura hasta final del dato estable	0		ns
tLZWE	/WE alta hasta baja impedancia	3		ns
tHZWE	/WE baja hasta alta impedancia		5	ns
tBW	Byte habilitado hasta fin de Escritura	7		ns

Tabla 15. Definición tiempos que intervienen en los ciclos de Lectura y Escritura

Figura 85. Ciclo de Lectura nº 1

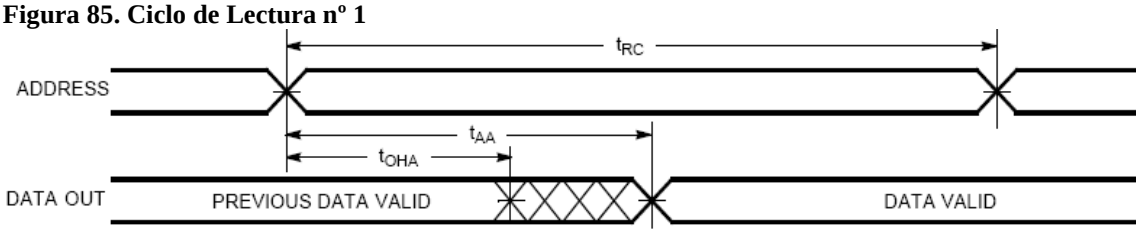


Figura 86. Ciclo de Lectura nº 2 (/OE controlada)

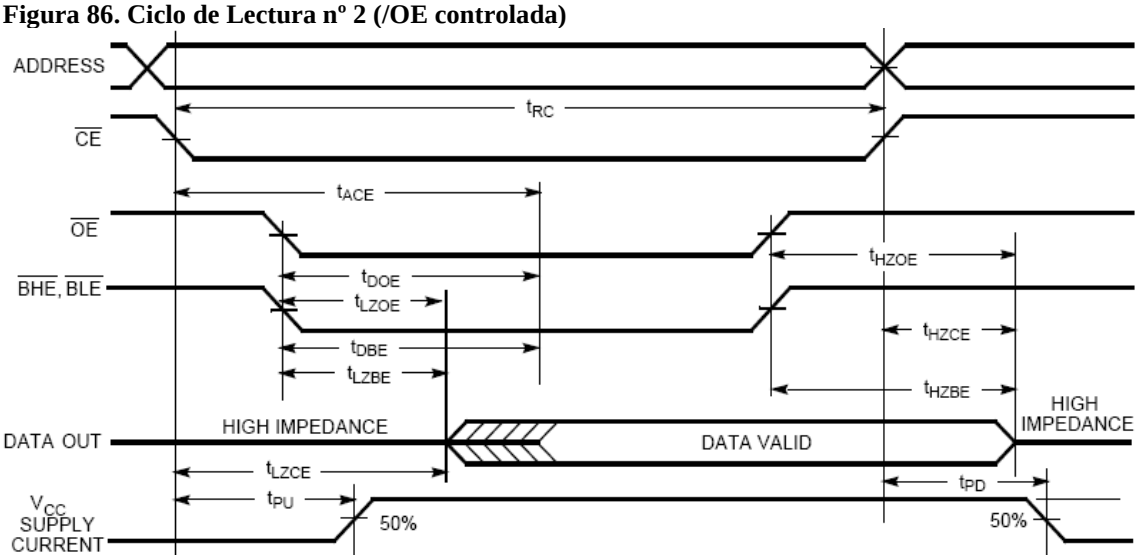


Figura 87. Ciclo de Escritura nº1 (/CE controlada)

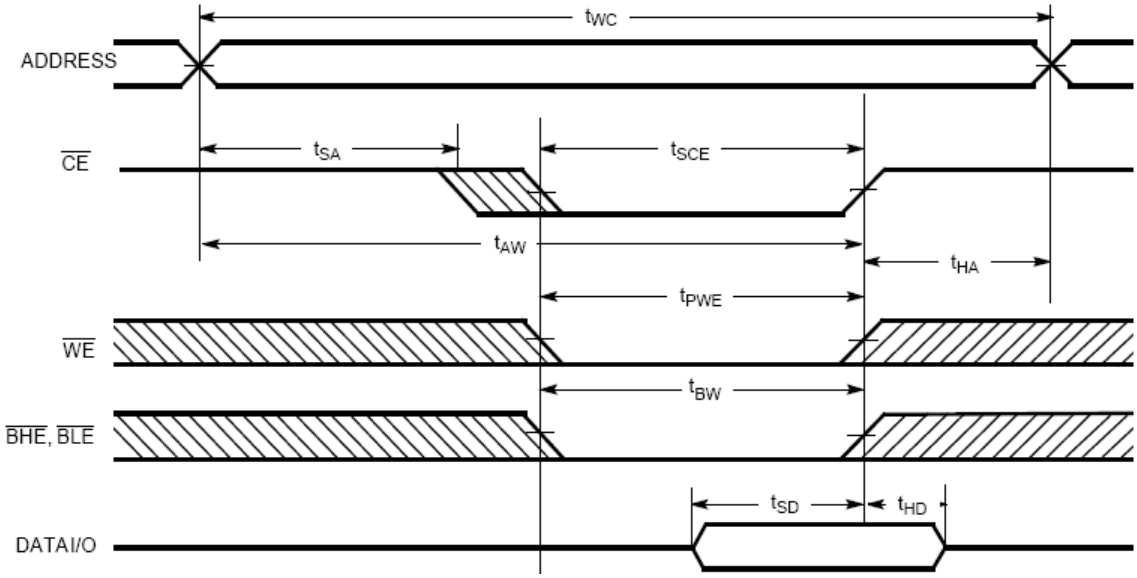


Figura 88. Ciclo de Escritura nº 2 (/BLE o /BHE controlados)

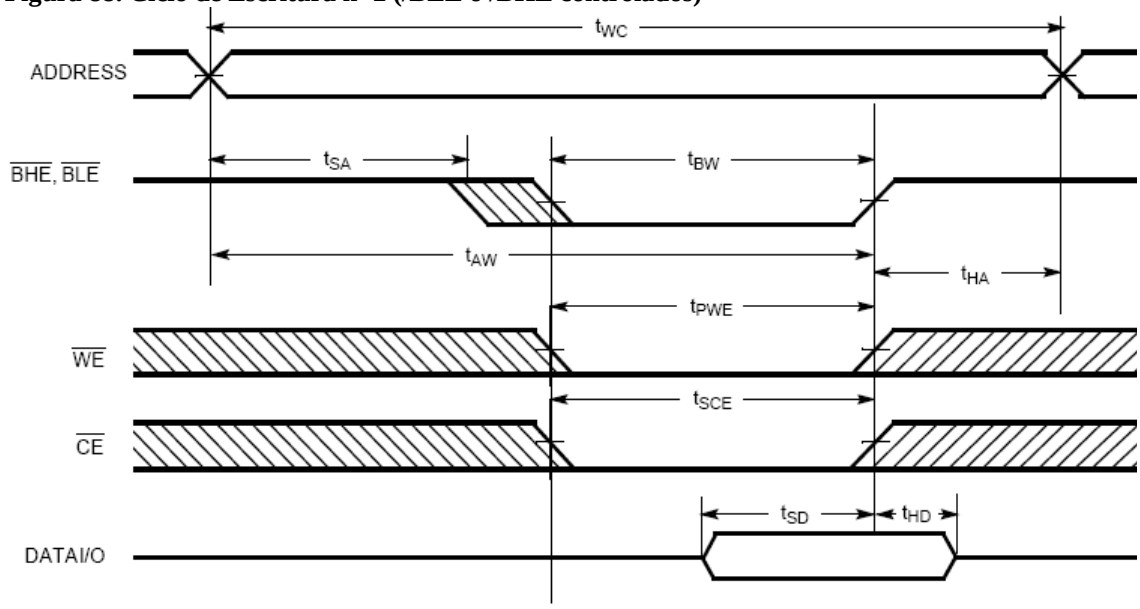
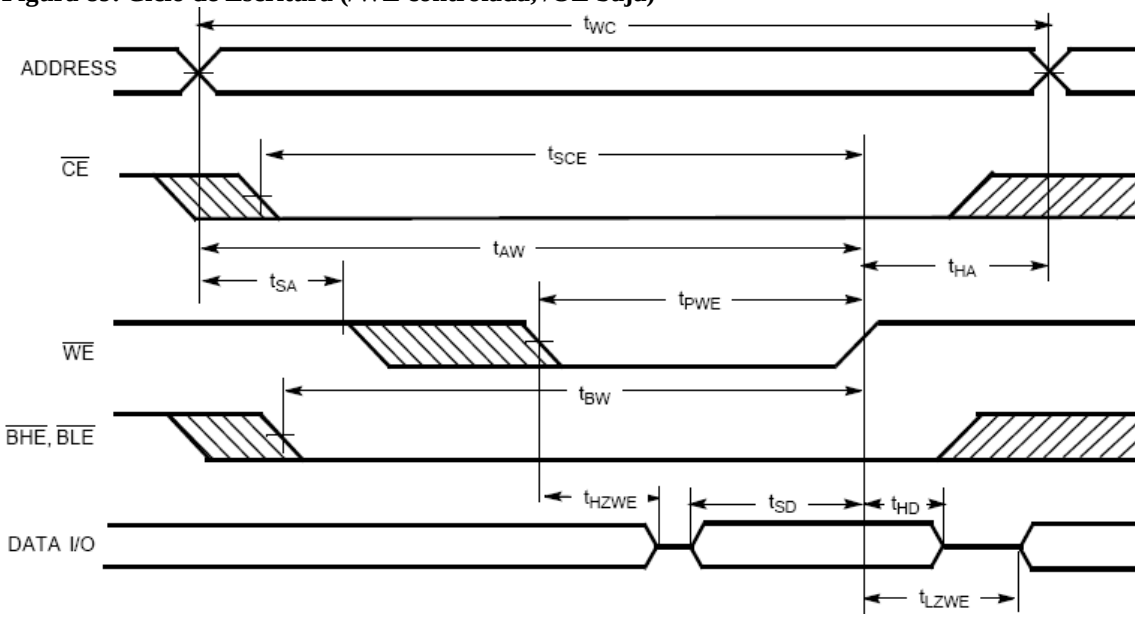


Figura 89. Ciclo de Escritura (/WE controlada, /OE baja)



Para tener una descripción completa de los modos de lectura y escritura de esta memoria es necesario construir su tabla de verdad (Tabla 16).

/CE	/OE	/WE	/BLE	/BHE	$I/O_0 – I/O_7$	$I/O_8 – I/O_{15}$	Modo	Alimentación
H	X	X	X	X	Alta impedancia	Alta impedancia	Apagado	Standby (I_{SB})
L	L	H	L	L	Salida datos	Salida datos	Lee todos los bits	Activa (I_{cc})
L	L	H	L	H	Salida datos	Alta impedancia	Lee sólo los 8 LSB	Activa (I_{cc})
L	L	H	H	L	Alta impedancia	Salida datos	Lee sólo los 8 MSB	Activa (I_{cc})
L	X	L	L	L	Entrada de datos	Entrada de datos	Escribe todos los bits	Activa (I_{cc})
L	X	L	L	H	Entrada de datos	Alta impedancia	Escribe sólo los 8 LSB	Activa (I_{cc})
L	X	L	H	L	Alta impedancia	Entrada de datos	Escribe sólo los 8 MSB	Activa (I_{cc})
L	H	H	X	X	Alta impedancia	Alta impedancia	Salidas deshabilitadas	Activa (I_{cc})

Tabla 16. Tabla de verdad. Especifica los modos de funcionamiento de la memoria.

Nota:

1. Nuestra memoria se configurará de manera que los modos empleados sean sólo los que se encuentran resaltados en gris.

3.3.2.3 Situación del módulo en la arquitectura

Conexiones

Como ya se ha comentado anteriormente la memoria utilizada en la arquitectura general es una SRAM Asíncrona de datos de 32 bits y 32M Byte de capacidad de almacenamiento. Tiene conectada al bus de direcciones la salida de un multiplexor 2 a 1 cuyas entradas son la señal de cuenta del contador de escritura con la señal *field_sel* concatenada como el bit más significativo y la señal de cuenta del contador de lectura a la que se le concatena la señal *field_sel* negada como el bit más significativo (figura 90). Estas señales de cuenta extendidas determinan la posición en memoria donde se escriben o se leen los datos.

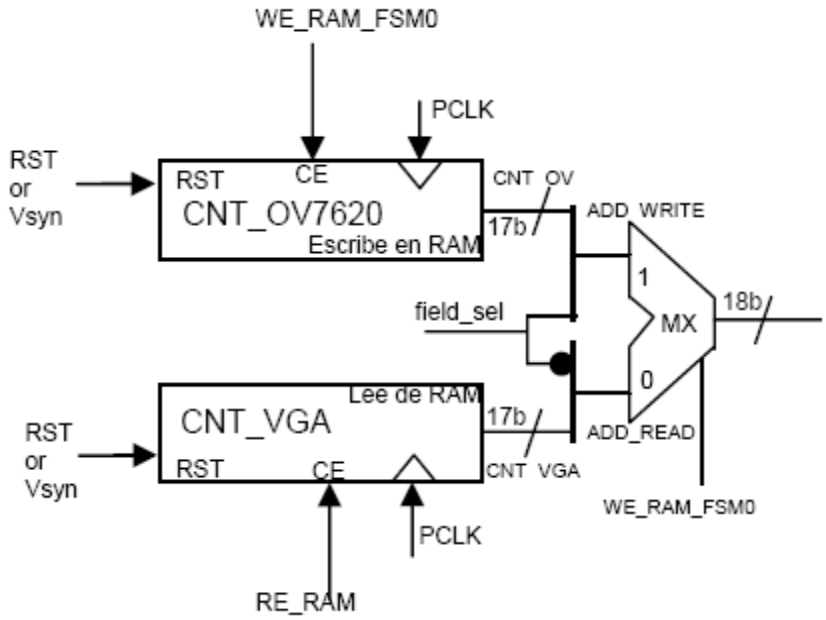


Figura 90. Generación del Bus de Direcciones de la memoria SRAM Asíncrona.

Nota:

1. Las características de los contadores se encuentran en la sección Resto de Componentes de este mismo capítulo.

El bus de direcciones es de 18 bits, pudiendo direccionar 262.144 celdas de memoria de 32 bits y dividirla en dos partes tal y como muestra la figura 91. Utilizaremos la división de memoria para almacenar dos frames consecutivos en distintas zonas de memoria, de manera que mientras guardamos el segundo un frame que llega en una zona, leeremos dos veces el frame ya almacenado en la otra zona de memoria. Esta es la base por la cual conseguimos duplicar la frecuencia funcionamiento de la arquitectura general.

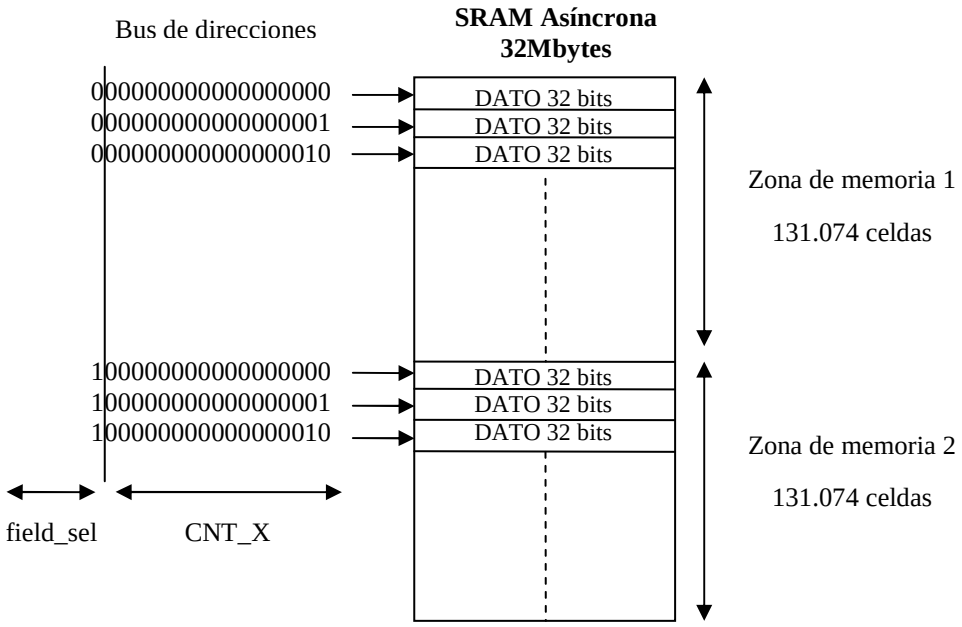


Figura 91. División de la memoria SRAM Asíncrona en dos zonas de memoria. Cada zona de memoria tiene capacidad suficiente para almacenar un frame.

El bus de datos está conectado a dos módulos a través de un buffer triestado controlado por la señal *WE_RAM_FSM0* (salida de la máquina de estados FSM0). Cuando se está escribiendo en la memoria el bus de datos se comporta como una entrada de información y, por tanto, se haya conectado al Banco de Registros de Entrada (*WE_RAM_FSM0* en alta). En el caso en el que se lee de memoria, el bus de datos se comporta como una salida de datos y se conecta a la entrada de datos de la cola FIFO.

La entrada *Byte Enabled* (/BE[0:3]) se conecta con todos sus bits a '0', de modo que en las operaciones de lectura y escritura se empleen siempre los 32 bits de información. Ver tabla de verdad de la memoria (Tabla 16).

En cuanto a la entrada Chip Enabled (/CE) de la memoria, debemos saber que ha sido conectada a la salida de una puerta NOR cuyas señales de entrada son *WE_RAM_FSM0* y *RE_RAM*. Así, el chip de memoria se habilitará cuando una de estas dos señales se encuentre en alta.

Como señal de entrada Write Enabled (/WE) tenemos la salida del circuito de la figura 92, formado principalmente por puertas lógicas y un registro tipo D de flanco descendente. Se trata de un circuito que retrasa en medio periodo de reloj *PCLK* (T_{PCLK} ÷ 2 = 74.07 ÷ 2 = 37.035 ns) la activación de la señal /WE, es decir, la activación de la señal *WE_RAM_FSM0* podrá en alta la señal /WE medio periodo más tarde. Con ello se consigue que tanto el bus de direcciones como el bus de datos tengan estabilizadas sus señales en el momento en de habilitación de la escritura. Además /WE sólo estará activa durante medio ciclo de reloj, que es tiempo suficiente para realizar la operación de Escritura (t_{WC} = 10 ns).

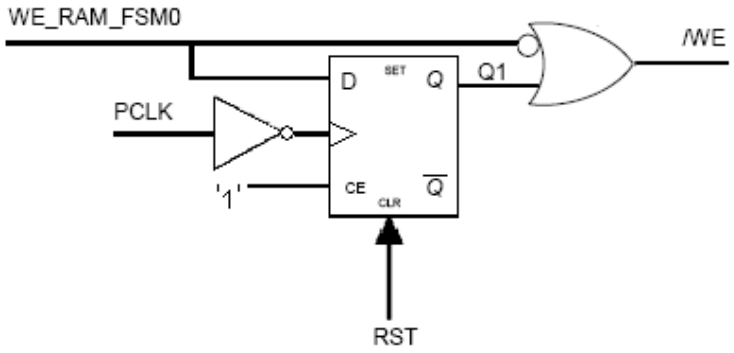


Figura 92. Circuito para retrasar la activación de la señal */WE* al activarse la señal *WE_RAM_FSM0*

Por último, la señal *RE_RAM* invertida será la entrada *Output Enabled* (/OE) de la SRAM. Cuando la señal *RE_RAM* esté en alta, a la entrada de /OE habrá un nivel bajo y el bus de datos se comportará como una salida.

Modo de funcionamiento

Todas estas conexiones hacen que la memoria SRAM Asíncrona quede controlada finalmente por las señales *WE_RAM_FSM0* procedente de la máquina de estados FSM0 y *RE_RAM* generada por el módulo FIFO y esta misma máquina de estados FSM0. Estas señales de control han sido producidas de manera que los ciclos de Lectura y Escritura en memoria se distribuyan como en la figura 93. En ella se muestra que estos ciclos sólo se van a producir cuando la información procedente de la cámara sea válida, es decir, cuando la señal *HREF* está en alta y *VSYN* en baja.

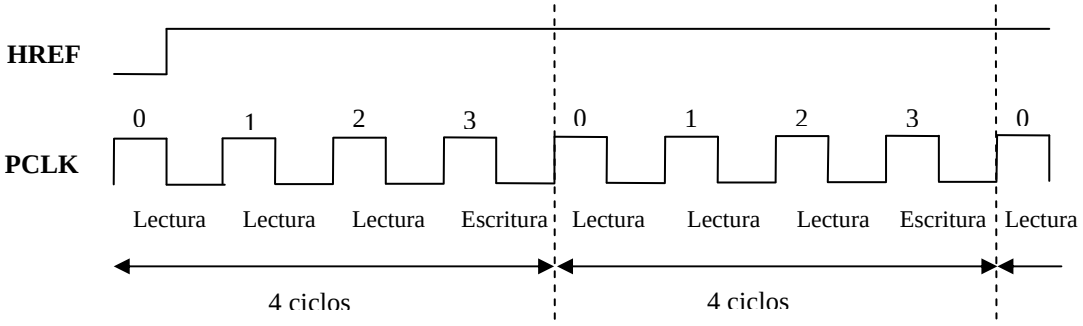


Figura 93. Ciclos de Lectura y Escritura de la SRAM

Observamos como existe una secuencia que se repite de 4 ciclos de duración. Los tres primeros ciclos son ciclos que permiten la operación de Lectura de memoria. Se permite esta operación en uno, en dos o en los tres ciclos, dependiendo de los requerimientos de la cola FIFO, puesto que es ésta junto con la máquina de estados FSM0 la que genera la señal de habilitación de lectura *RE_RAM*. En cuanto al último ciclo de reloj de la secuencia, en él siempre se va a realizar la operación de Escritura en memoria.

3.3.3 COLA FIFO

3.3.3.1 Utilidad y funcionalidad

Una cola FIFO es un subsistema de memoria donde una secuencia de datos puede ser escrita y recuperada exactamente en el mismo orden. No se requiere direccionamiento explícito, y las operaciones de lectura y escritura pueden ser completamente independientes usando relojes distintos.

En nuestro caso hemos utilizado una memoria FIFO cuyo código VHDL ha sido descargado de la web de XilinX. Se trata de una memoria FIFO asíncrona de 511 celdas de memoria de 36 bits que vamos a utilizar en su modo de lectura y escritura con relojes independientes. Este modelo soporta escrituras de 200 Mhz y lecturas de hasta 500 Mhz. También es posible hacerla funcionar con un solo reloj, pero nos centraremos en el modo de operación empleado.

La figura 94 muestra la FIFO como una "caja negra". Observamos el suministro de datos de 36 bits y los relojes de lectura y escritura, y sus correspondientes entradas de habilitación. Los datos de salida tienen la misma anchura que los datos de entrada, a diferencia del bloque RAM donde las anchuras pueden ser distintas.

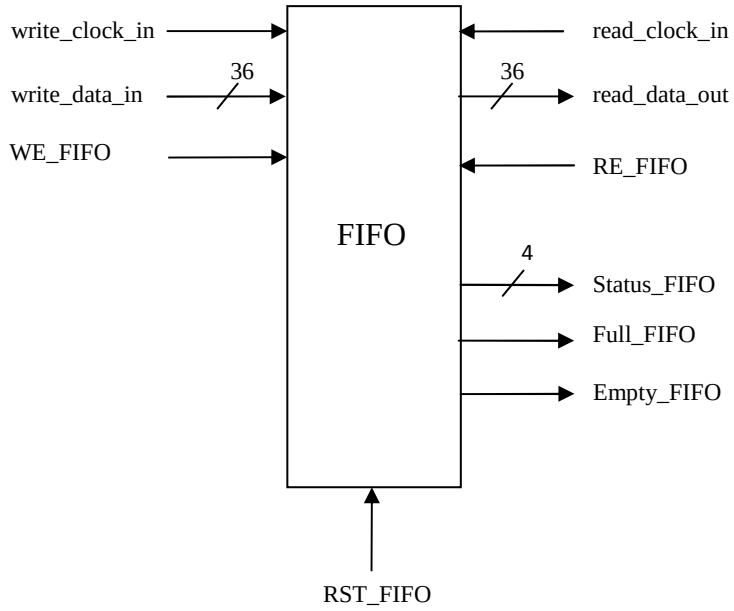


Figura 94. Esquema de la cola FIFO

La tabla 17 presenta la definición de los puertos:

Nombre del Puerto	**Dirección**	**Descripción**
write_clock_in	Entrada	Señal de reloj empleada para la escritura. Se conectará a la señal *PCLK*, reloj procedente de la cámara cuya frecuencia nominal es de 13.5 Mhz.
write_data_in	Entrada	Bus de datos de entrada. Tiene un tamaño de 36 bits[*1]
WE_FIFO	Entrada	Señal que en estado alto habilita la escritura en memoria.
RST_FIFO	Entrada	Señal de reset asíncrono del circuito. Al ponerse en alta borra el contenido de la memoria.
read_clock_in	Entrada	Señal de reloj empleada para la lectura. Se conectará a la señal *VGA_CLK*, reloj procedente de la cámara cuya frecuencia nominal es de 25.175 Mhz.
read_data_out	Salida	Bus de datos de salida. Tiene un tamaño de 36 bits[*2]
RE_FIFO	Salida	Señal que en estado alto habilita la lectura de los datos almacenados en memoria.
Full_FIFO	Salida	Señal que indica si la memoria está llena o no. Cuando están ocupadas las 511 celdas de memoria toma un valor alto, para valores inferiores de ocupación toma un valor bajo.
Empty_FIFO	Salida	Señal que indica si la memoria está vacía o no. Toma un valor alto cuando no hay ningún dato almacenado.
Status_FIFO	Salida	Señal de 4 bits que indica el porcentaje de llenado del buffer de memoria.

Tabla 17. Definición de los puertos de entrada y salida de la cola FIFO

Notas:

1. En nuestro caso la señal de información es solo de 32 bits, por lo que le concatenaremos cuatro ceros en las posiciones más significativas antes de introducirla como entrada *write_data_in* de la FIFO.

2. La salida *write_data_in* proporciona 36 bits de los cuales nos quedaremos con los 32 bits menos significativos.

En cuanto a los procesos de lectura y escritura en memoria, la figura 95 proporciona el diagrama temporal que determina los valores que deben adquirir las señales de la FIFO.

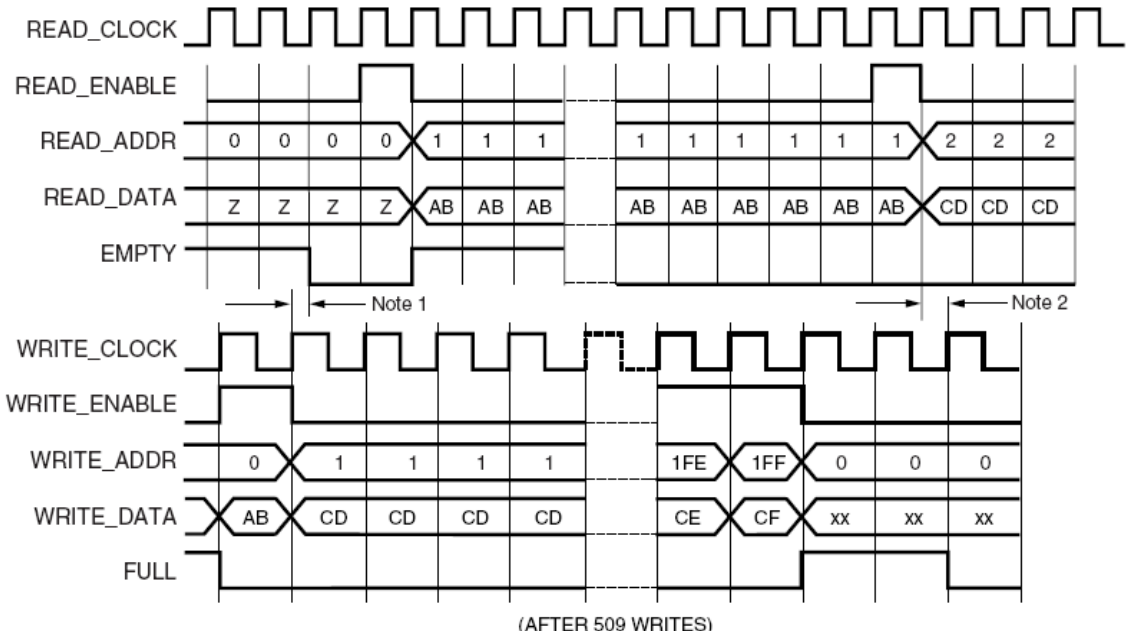


Figura 95. Diagrama temporal de los procesos de lectura (imagen superior) y escritura (imagen inferior)

Notas:

1. La señal *Empty_FIFO* se pondrá baja si la dirección de escritura se encuentra estable antes del flanco ascendente del reloj de lectura. Si no, se pondrá baja un ciclo después
2. La señal *Full_FIFO* se pondrá baja si la dirección de lectura se encuentra estable antes del flanco ascendente del reloj de escritura. Si no, se pondrá baja un ciclo después

La señal *Status_FIFO* indica como de lleno está el buffer de memoria: ½, ¼, etc, tal y como se muestra en la tabla 18. Generar esta señal en su versión asíncrona es una tarea compleja y, por tanto, requiere más lógica. La salida de la *Status_FIFO* tendrá un ciclo de latencia en las operaciones de escritura, y dos ciclos de latencia en las de lectura.

Bit 3	Bit 2	Bit 1	Bit 0	Cnt_FIFO/Status_FIFO
1	1	1	1	15/16 de ocupación
1	1	1	0	7/8 de ocupación
1	1	0	1	13/16 de ocupación
1	1	0	0	3/4 de ocupación
1	0	1	1	11/16 de ocupación
1	0	1	0	5/8 de ocupación
1	0	0	1	9/16 de ocupación
1	0	0	0	1/2 de ocupación
0	1	1	1	7/16 de ocupación
0	1	1	0	3/8 de ocupación
0	1	0	1	5/16 de ocupación
0	1	0	0	1/4 de ocupación
0	0	1	1	3/16 de ocupación
0	0	1	0	1/8 de ocupación

0	0	0	1	1/16 de ocupación
0	0	0	0	< 1/16 de ocupación

Tabla 18. Descripción de las señales Status_FIFO y Cnt_FIFO (contador interno de la FIFO)

En nuestro caso, se ha establecido como criterio de configuración, que la FIFO no se llene nunca por encima del 75 % (3/4 full en la tabla anterior). El cumplimiento de dicho criterio se asegura a través de la señal *RE_RAM* que determinará los ciclos de lectura en la memoria RAM y de escritura en la FIFO. Esta señal se obtiene a través del esquema de la figura 96, y sólo se pondrá en alta cuando se cumplan al mismo tiempo las siguientes condiciones: la FIFO no esté llena (señal *full_FIFO* en baja), nos encontremos en un ciclo escritura en la memoria RAM (*RE_RAM_FSM0* en alta) y la cola FIFO se encuentre por debajo del 75% de ocupación (salida del comparador en alta).

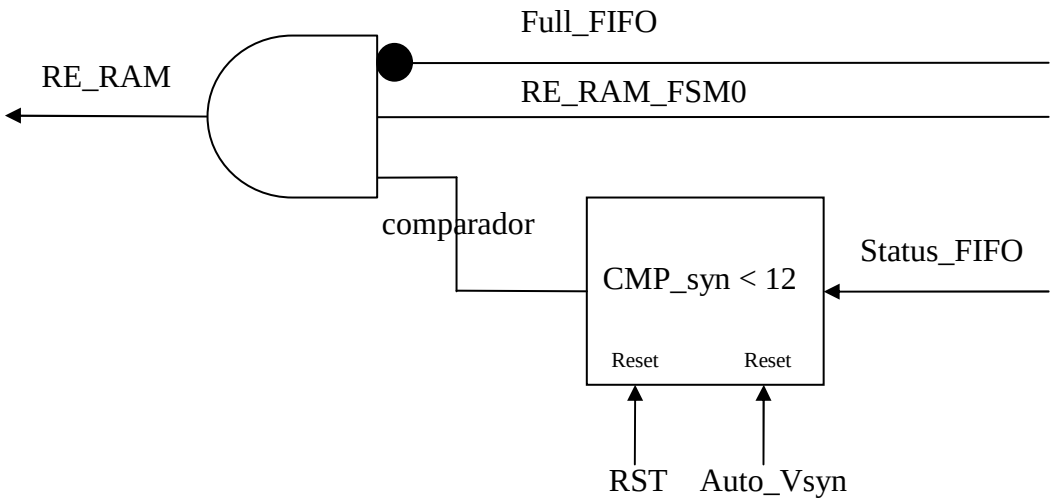


Figura 96. Generación de la señal de habilitación de lectura en la memoria RAM y escritura en la cola FIFO

3.3.3.2 Situación del módulo en la arquitectura

Para comprender el papel que juega el módulo FIFO en la arquitectura general del proyecto, debemos saber que en esta arquitectura vamos a trabajar con dos relojes distintos, el *PCLK* (13.5 Mhz) procedente de la cámara Ov7620 y el *VGA_CLK* (25.175 Mhz) procedente de un oscilador de la FPGA. Por tanto, tendremos una primera parte de la arquitectura general que funcionará con un reloj y una segunda parte que lo hará con otro. El módulo FIFO va a ser el encargado de sincronizar ambas partes.

Por un lado, tenemos las conexiones de la FIFO correspondientes a la primera zona, donde la señal de reloj de escritura *write_clock_in*, está conectada al reloj *PCLK*; y la señal de entrada de datos *write_data_in* es el bus de datos de la memoria RAM Asíncrona concatenado con 4 bits a 0 en la posición más significativa. Además, la FIFO se reseteará cada vez que la cámara suministre un nuevo frame (VSYN se pone a 1).

Por otra parte, tenemos las conexiones correspondientes a la segunda zona, donde la señal de reloj de escritura *write_clock_in*, está conectada al reloj *VGA_CLK*; y la señal de salida de datos *read_data_in* se lleva al Banco de registros de salida.

En cuanto al resto de conexiones de la FIFO, la señal de habilitación de escritura *WE_FIFO* se conecta a la señal *RE_RAM*, mientras que la de habilitación de lectura *RE_FIFO* lo hace a la señal *RE_FIFO_VGA* correspondientemente. La generación de la señal *RE_FIFO_VGA* viene dada por la máquina de estados FSM1 de la arquitectura general que procedemos a explicar en el apartado "Maquinas de estado" de este mismo capítulo.

3.3.3.3 Arquitectura interna

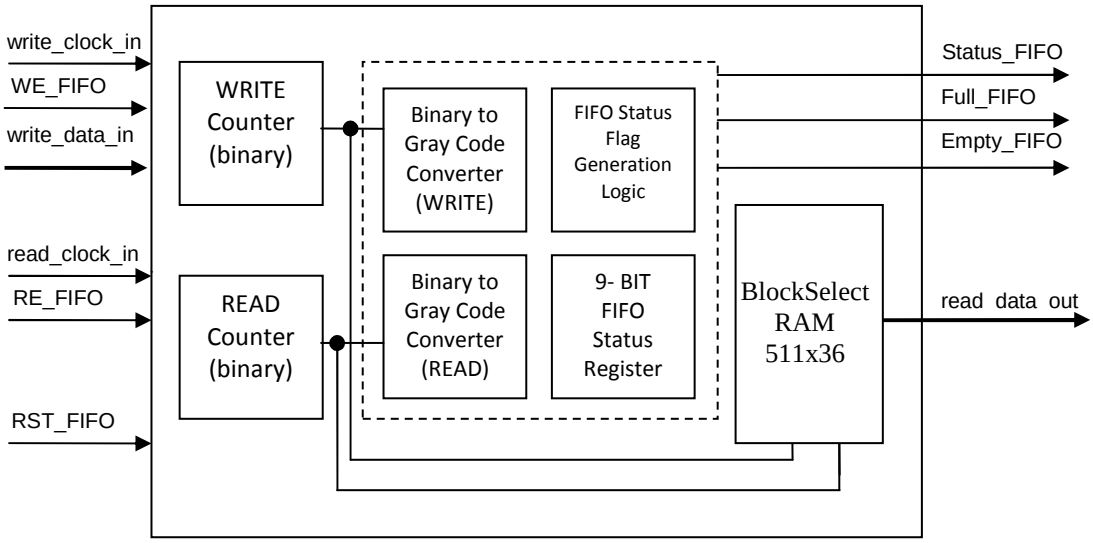


Figura 97. Diagrama de bloques de la FIFO

El esquema interno de la FIFO queda definido por la figura 97, donde podemos ver que en su modo de lectura y escritura con relojes independientes, utiliza dos contadores ascendentes binarios de 9 bits cuya cuenta representa las direcciones de acceso al bloque de memoria RAM. Estas direcciones binarias de los contadores son también entradas de unos codificadores, que las codificarán en código Gray para crear varios punteros de direcciones (read_addrgray, read_nextgray, read_lastgray, write_addrgray, write_nextgray) y generar con ellos las salidas *Full_FIFO y Empty_FIFO* lo más rápidamente posible. Con ello se pretende asegurar que estas señales estén siempre limpias, es decir, que no entren en un estado desconocido, ya que al ser asíncronas entre sí podrían producirse errores. En el peor caso las señales *Full_FIFO y Empty_FIFO* estarán activas el ciclo de reloj más largo, pero esto no generará error.

Cuando el puntero de lectura (read_addrgray) sea igual al de escritura (write_addrgray), la FIFO estará vacía, mientras que cuando el puntero de escritura (write_addrgray) sea igual al siguiente de lectura (read_nextgray), la FIFO estará llena, teniendo almacenada 511 datos.

3.3.4 MÁQUINAS DE ESTADO

El funcionamiento conjunto de todos los módulos estudiados anteriormente queda controlado por dos máquinas de estado, las FSM0 y FSM1. A partir de estas máquinas se crearán las señales de control de la arquitectura general. Procedemos a analizarlas independientemente.

3.3.4.1 Máquina de estado FSM0

La máquina de estados FSM0 es la encargada de generar las señales de control de la zona de la arquitectura controlada por el reloj de la cámara (*PCLK*). Determina el valor de las señales de habilitación del Banco de Registros de Entrada (*CE_RX_FSM0*). También crea las señales *WE_RAM_FSM0* y *RE_RAM_FSM0* (a partir de ella se genera *RE_RAM*) encargadas de controlar el funcionamiento sincronizado de la memoria SRAM, la cola FIFO y los contadores de Lectura y Escritura.

En la figura 98 se muestra el diagrama de estados de esta máquina. Claramente es de tipo Moore, ya que el valor de la salida en cada instante es función únicamente del estado en el que se encuentra. El cambio de un estado a otro está asociado a las señales de sincronismo de la cámara *HREF* y *VSYN*). El reset coincide con el reset general de la arquitectura (*RST*).

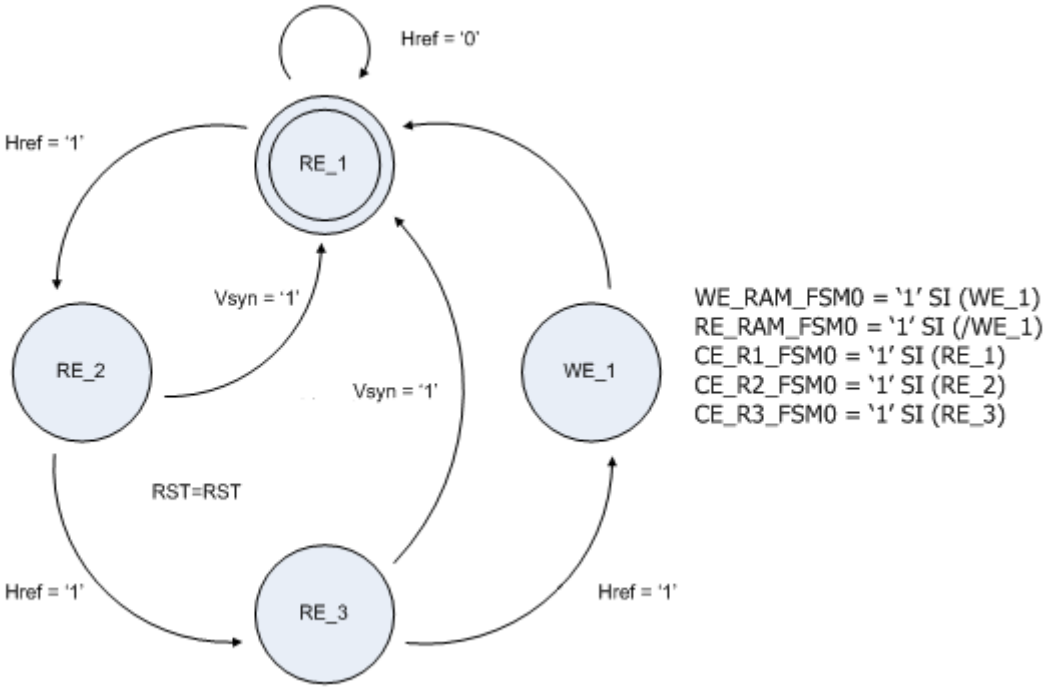


Figura 98. Diagrama de estados de la FSM0

La figura 99 representa el cronograma que surge de la máquina de estados FSM0. Está compuesto por dos tipos de ciclos, el de Lectura y el de Escritura, formando

una secuencia de 4 ciclos (4 estados de la FSM0) que se repite mientras se cumpla que la información procedente de la cámara sea válida, es decir, cuando las señales *HREF* y la señal *VSYN* en baja. También se expone el valor que adquieren las señales de control de la SRAM y la FIFO al iniciarse esta secuencia.

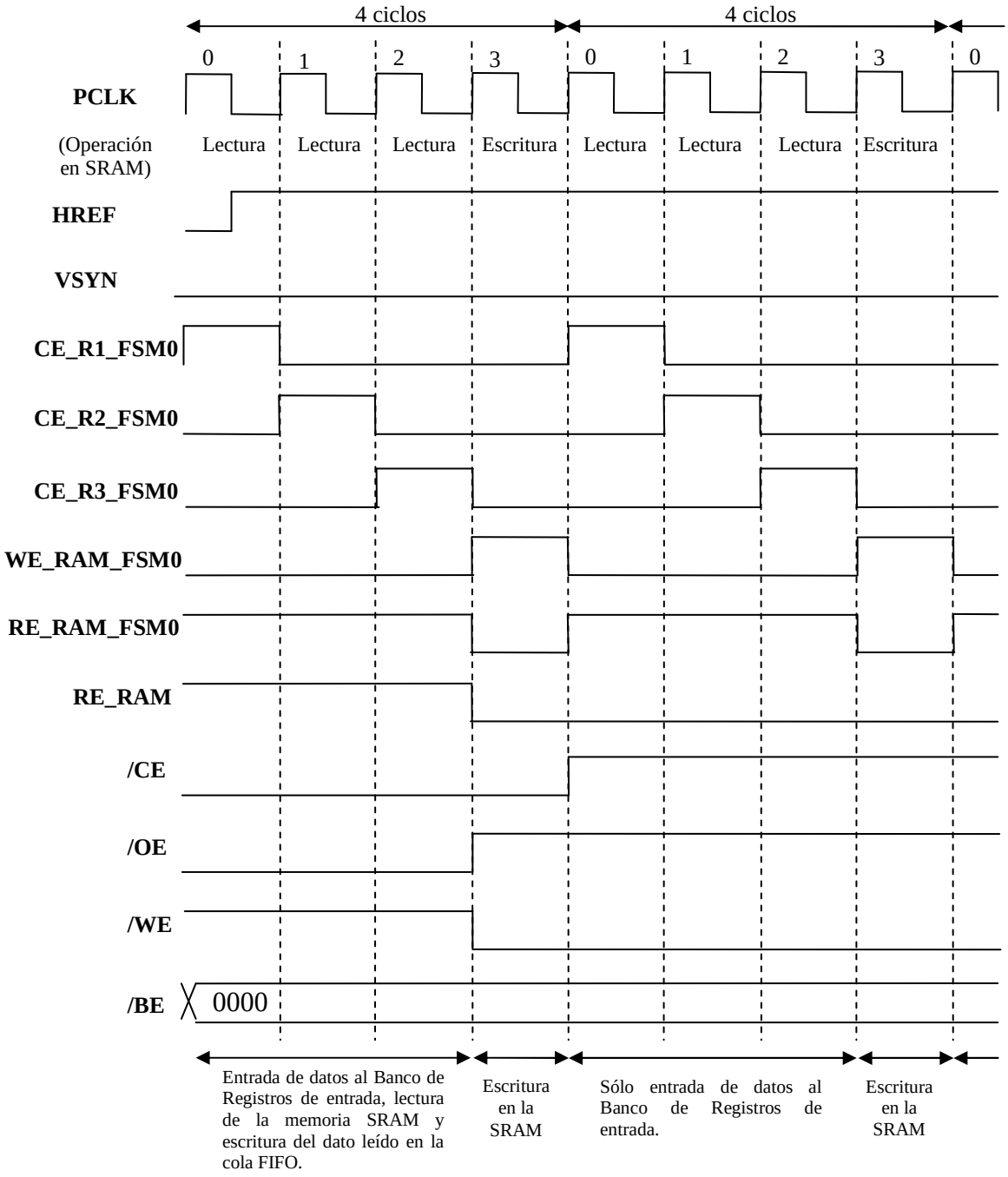


Figura 99. Cronograma de la FSM0. Control de la entrada y almacenamiento de información.

Los ciclos denominados de Lectura indican cuando es posible leer un dato de la memoria SRAM y almacenarlo en la cola FIFO. Para ello la señal *WE_RAM_FSM0* estará siempre en baja y la señal *RE_RAM_FSM0* en alta en este tipo de ciclos, siendo la señal *RE_RAM* (generada en el módulo FIFO) la que decidirá si se lee o no., independientemente de lo anterior, las señales de habilitación del Banco de Registros de

Entrada (*CE_R1_FSM0*, *CE_R2_FSM0* y *CE_R3_FSM0*) se van activando en sus correspondientes ciclos para tener cargado el dato de entrada a la RAM en el ciclo de Escritura. Por tanto, el ciclo de Lectura se puede utilizar de 2 maneras: carga dato en Banco de Registros y lee de memoria y almacena en la FIFO (*RE_RAM* en alta); o sólo carga dato en Banco de Registros, sin leer de la SRAM (*RE_RAM* en baja).

Los ciclos de Escritura se utilizan para escribir un nuevo dato en la memoria SRAM. La señal de habilitación de escritura *WE_RAM_FSM0* tomará un valor alto y las señales de habilitación del Banco de Registros (*CE_R1_FSM0*, *CE_R2_FSM0* y *CE_R3_FSM0*) y de habilitación de lectura (*RE_RAM_FSM0*) tomarán un valor bajo.

En referencia a la implementación de la FSM0 en VHDL, se ha hecho utilizando 3 procesos, dos combinacionales y uno secuencial. Los procesos combinacionales se emplean, uno para calcular el estado siguiente en función del estado actual y la señal de entrada, y el otro, para obtener el valor de la salida en función del estado actual; mientras que el proceso secuencial es utilizado para registrar tanto el estado actual como las salidas (*WE_RAM_FSM0*, *RE_RAM_FSM0* y *CE_RX_FSM0*).

3.3.4.1 Máquina de estado FSM1

La máquina de Estados FSM1 produce las señales de control de la zona de la arquitectura general controlada por la señal de reloj *VGA_CLK*. Proporciona un valor a la señal de habilitación (*CE_BR_VGA*) y al bus de selección del multiplexor del Banco de Registros de Salida (*SEL_BR_ VGA [1:0]*). También determina cuando es posible extraer datos de la cola FIFO con la señal *RE_FIFO_VGA*.

Su diagrama de estados se presenta en la figura 100 y en él podemos ver como la transición entre estados está condicionada por las señales *PIXELES_VISIBLES* (indica los píxeles visibles de un frame con un valor alto) y *Vsyn_VGA* (sincronismo vertical procedente del generador de sincronismos).

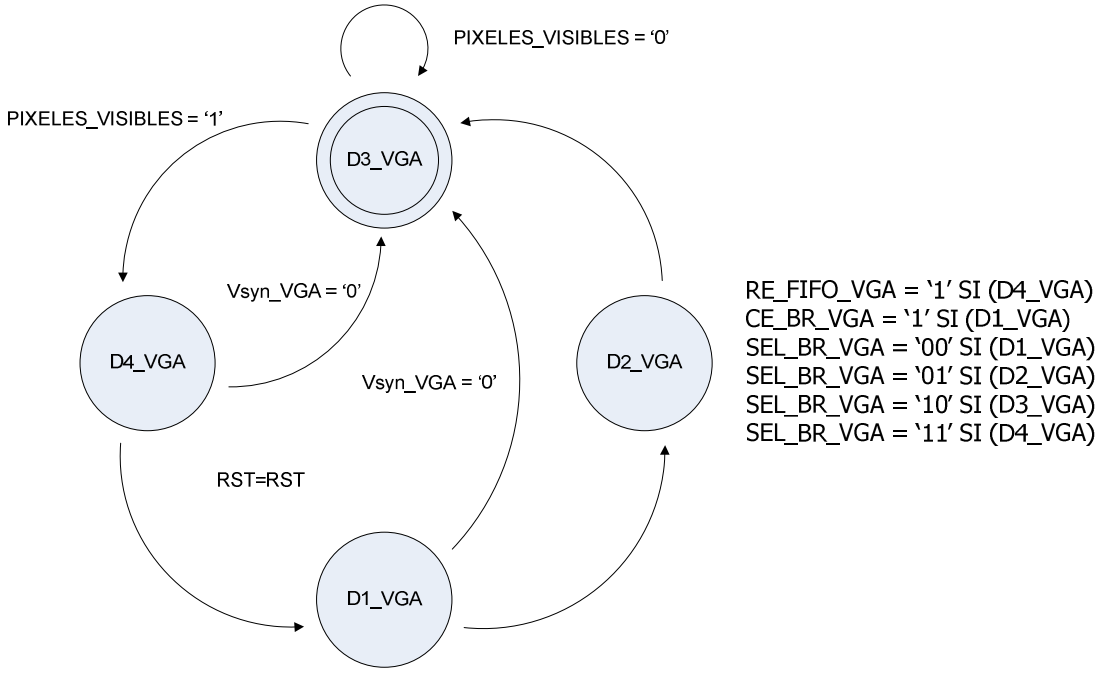


Figura 100. Diagrama de estados de la FSM1

A continuación presentamos el cronograma (figura 101) generado por esta máquina de estados y a partir del cual podemos controlar la salida de datos de la arquitectura general.

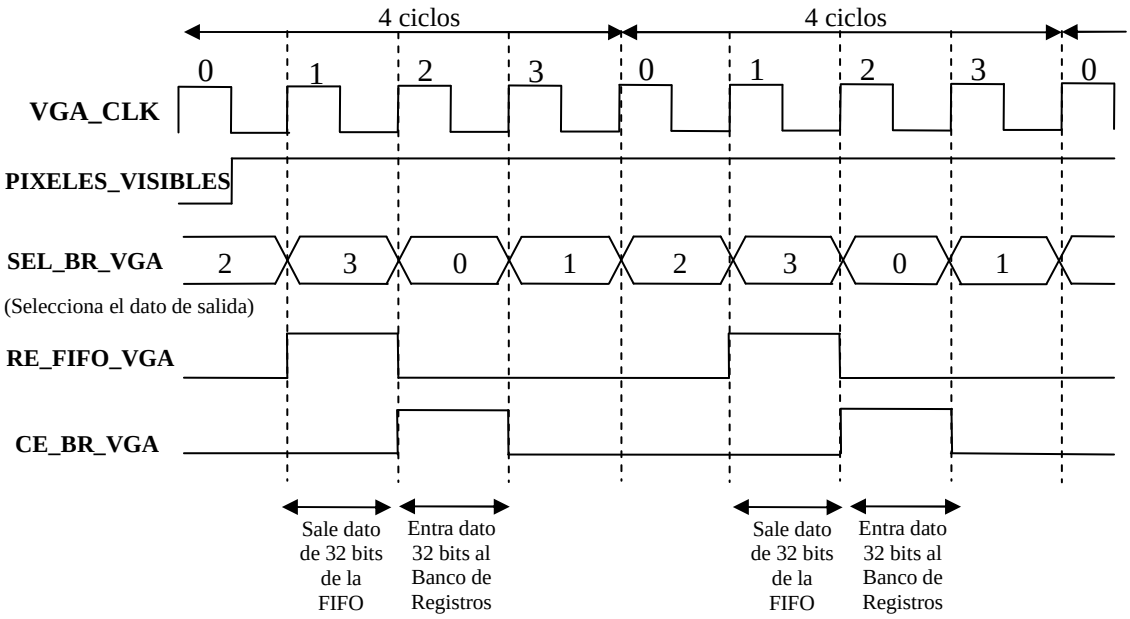


Figura 101. Cronograma de la FSM1. Control de la salida de datos.

Por último, en cuanto a la implementación de la FSM1 en VHDL, debemos comentar que se han utilizando 3 procesos, dos combinacionales y uno secuencial. Los procesos combinacionales se emplean, uno para calcular el estado siguiente en función del estado actual y la señal de entrada, y el otro, para obtener el valor de la salida en función del estado actual; mientras que el proceso secuencial es utilizado para registrar tanto el estado actual como las salidas (*CE_BR_VGA*, *SEL_BR_VGA [1:0]* y *RE_FIFO_VGA*).

3.3.5 RESTO DE COMPONENTES

3.3.5.1 Generador Filed_sel

La señal *field_sel* es utilizada por la memoria SRAM Asíncrona para seleccionar la zona de la memoria en la que se quiere trabajar. La idea es que cada vez que llegue un nuevo frame procedente de la cámara se invierta el valor de esta señal y se permuten las zonas de lectura y escritura en la memoria.

El Generador Field_sel se construye utilizando un flip-flop tipo D y un inversor, tal y como muestra la figura 102. Su señal de reloj está conectada a la señal de sincronismo vertical *VSYN*. Esta señal se pone en alta cierto tiempo con la llegada de cada nuevo frame. Por tanto, cada vez que llega un nuevo frame de la cámara se produce un flanco en la señal de reloj y la señal de salida *field_sel* invierte su valor, manteniéndolo hasta la llegada del siguiente frame.

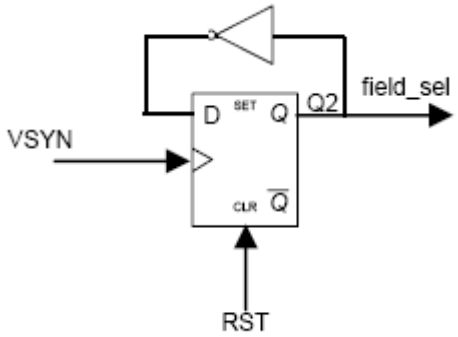


Figura 102. Estructura del Generador Field_sel

3.3.5.2 Control de reset asíncrono

Este módulo es el encargado de activar la señal de reset *RST* de la arquitectura. Cuando *RST* toma un valor alto se resetean todos los módulos utilizados en la arquitectura general excepto las memorias SRAM y la FIFO que sólo se borran desconectándolas de la alimentación.

La figura 103 muestra su implementación interna. Dispone de dos entradas: la señal *RST_button* y la señal *FODD*. La señal *RST_button* está conectada al switch 3 de la FPGA, ejerciendo así de un reset manual. La señal *FODD* procede de la cámara OV7620 e indica si la cámara está trabajando en modo progresivo (*FODD* en baja) o en modo entrelazado (*FODD* en alta). Como en nuestro caso queremos trabajar siempre en modo progresivo, si por algún motivo la cámara cambia a modo entrelazado se activará el reset del circuito. Se trata por tanto de un reset automático.

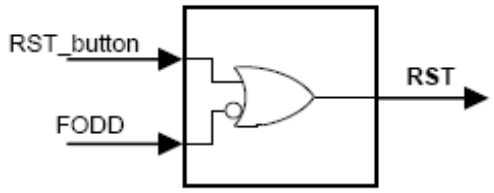


Figura 103. Control de Réset Asíncrono

3.3.5.3 Banco de registros

Banco de Registros de Entrada

La cámara OV7620 proporciona la información en datos de 8 bits (un píxel), mientras que la memoria SRAM Asíncrona almacena datos de 32 bits. El Banco de Registros de entrada es el encargado de construir datos de 32 bits a partir de datos de 8 bits, siendo el módulo de compatibilidad entre la cámara y la memoria.

Se implementa mediante 3 registros tipo D, como podemos ver en la figura 104. El reloj utilizado es el *PCLK* de la cámara, ya que este módulo se encuentra en la zona de la arquitectura sincronizada con ella. Las señales de habilitación de los registros *CE_X_FSM0* son producidas por la máquina de estados FSM0 lo que implica una sincronización con la memoria. También posee una entrada de reset que irá conectada al reset general del circuito (RST). Su bus de salida *Datos [31:0]* se conecta al bus de datos de la SRAM a través de un buffer triestado controlado por la señal *WE_RAM_FSM0.*

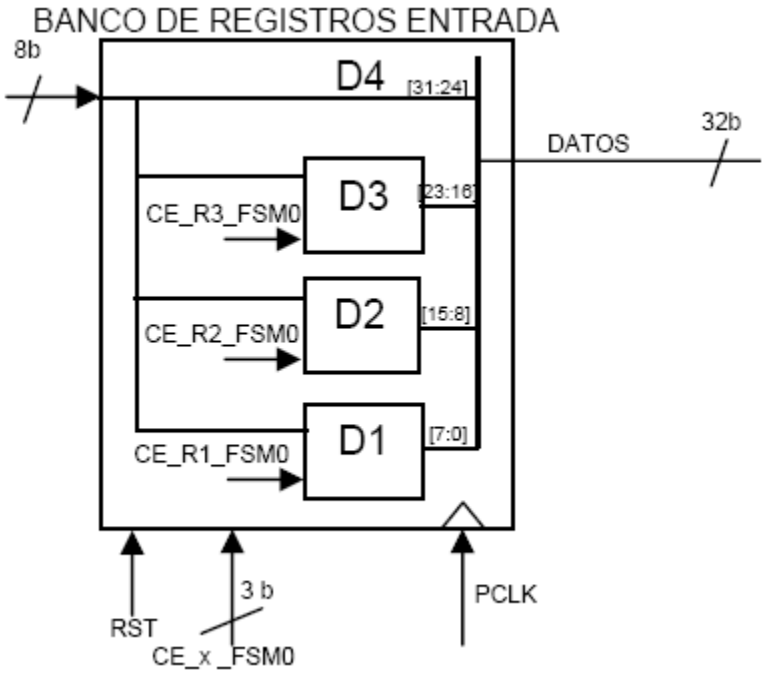


Figura 104. Banco de Registros de Entrada

Para poder explicar su funcionamiento nos ayudamos de la figura 105. Observamos explícitamente la sincronización que tiene este módulo con el módulo de memoria. En los ciclos de Lectura el bus de datos de la memoria se comporta como una salida de datos y el buffer triestado se comporta como una alta impedancia. Por tanto, es en estos tres ciclos de Lectura en los que los datos son registrados. En el primer ciclo se registra el dato 1 en el Registro D1 cuya salida se corresponde con *Datos [7:0]*, en el segundo ciclo se registra el dato 2 en el D2 (*Datos [15:8]*), y en el tercero el dato 3 en el D3 (*Datos [23:16]*). De esta forma, en el cuarto ciclo de la secuencia, con la llegada del dato 4 (*Datos [31:24]*) obtenemos la salida de 32 bits. Este cuarto ciclo es el ciclo de Escritura en memoria, por lo que el buffer triestado se comporta como un conductor y la información del bus de salida *Datos* del Banco de Registros de entrada se almacena en la SRAM.

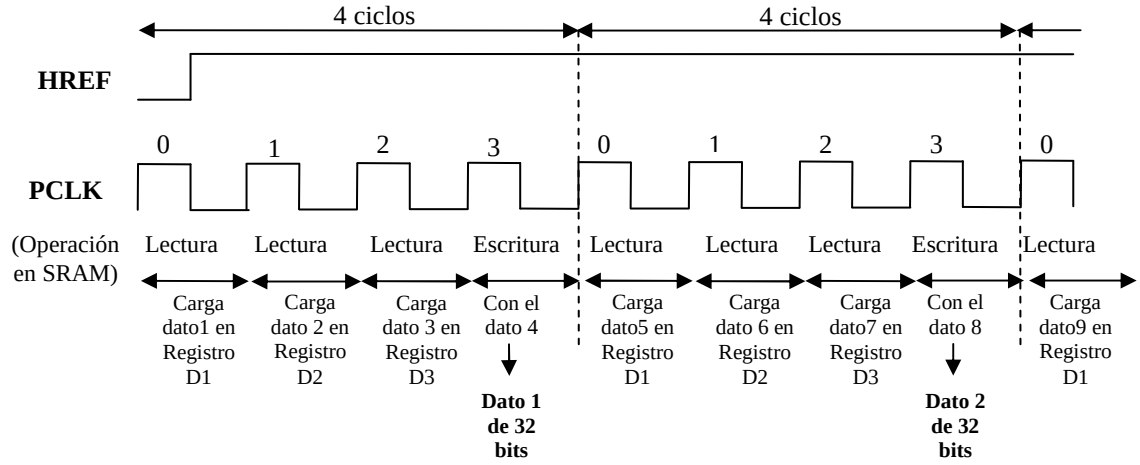


Figura 105. Cronograma que muestra el funcionamiento del Banco de Registros de Entrada

Banco de Registros de Salida y multiplexor de Salida

La señal de salida (*SALIDA*) de información de la arquitectura general está conectada a las entradas R, G y B del convertidor ADV7125. Este convertidor trabaja con datos de 8 bits, por lo que es necesario un circuito que adapte los datos de 32 bits que proporciona la cola FIFO con las entradas RGB del convertidor. Este circuito se haya expuesto en la figura 106 y consta de dos módulos: el Banco de Registros de Salida y un multiplexor 2 a 1.

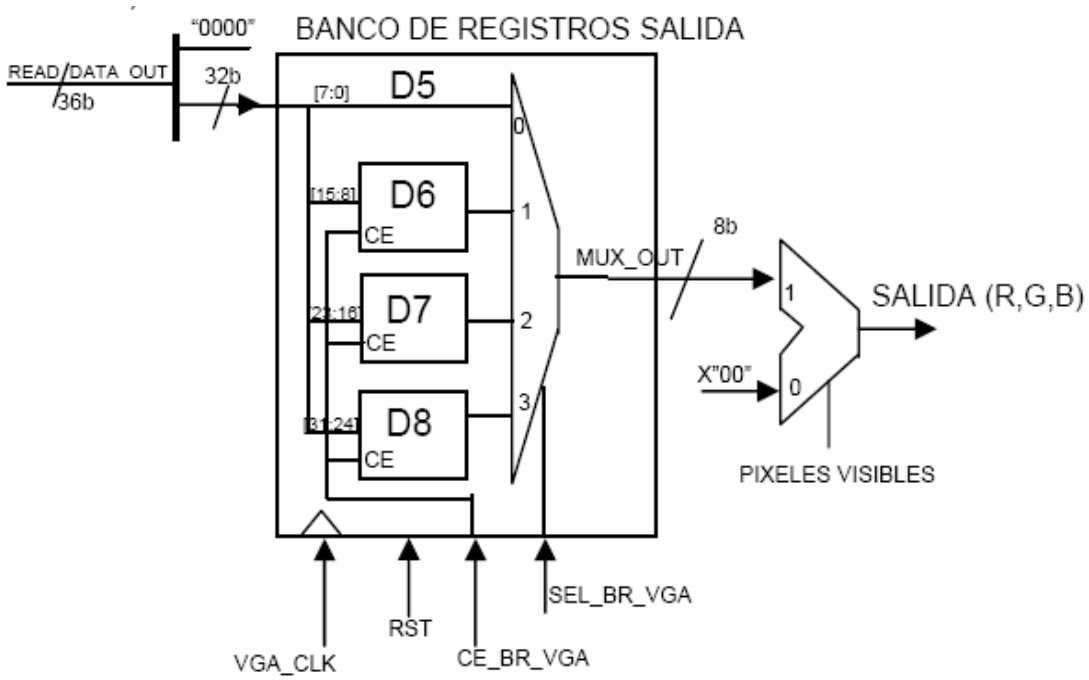


Figura 106. Circuito de adaptación entre la información procedente de la FIFO (32 bits) y la información de entrada al monitor VGA (8bits).

El Banco de Registros de Salida está formado por 3 registros tipo D y un multiplexor 4 a 1. El reloj utilizado es el *VGA_CLK* procedente de un oscilador interno de la FPGA, de modo que este módulo se encuentra en el segundo dominio de la arquitectura completa. Las entradas de información de los registros están conectadas al bus de datos de entrada tal y como muestra en la figura 106. En referencia a las entradas de habilitación de los registros, están controladas por la señal *CE_BR_VGA* que proporciona la máquina de estados FSM1. Las salidas de los registros D8, D7 y D6 están conectadas respectivamente a las entradas 3, 2 y 1 del multiplexor, dejando como entrada 0 los 8 bits menos significativos del dato 32 bits del bus de entrada.

El multiplexor de salida es el encargado de llevar la información a la señal de salida *SALIDA[7:0]*. Su señal de selección es la señal *PIXELES_VISIBLES* que indica los píxeles visibles de un frame. Cuando toma un valor bajo significa que el pixel en cuestión no es visible y por tanto la información que enviemos no es relevante (no se pintará en pantalla). En este caso selecciona la entrada 0 que está conectada a un dato todo ceros. Cuando la señal *PIXELES_VISIBLES* toma un valor alto significa que el pixel es visible, siendo necesario enviar información válida. En esta ocasión el multiplexor selecciona la entrada 1 que se encuentra conectada a la salida de información del Banco de Registros de Salida.

Para poder explicar el funcionamiento conjunto de estos dos módulos nos ayudaremos de la figura 107. En ella distinguimos una secuencia que se repite cíclicamente una vez que comienzan los píxeles visibles. El ciclo 0 proporciona en la señal SALIDA el dato almacenado en el registro D7. En el ciclo 1 saldrá el dato almacenado en el registro D8 y se obtendrá en el bus de entrada del Banco de Registros un nuevo dato de 32 bits procedente de la FIFO. En el ciclo 2 se cargarán los 24 bits más significativos del dato de 32 bits en los registros D6, D7 y D8; y se sacarán los 8 bits menos significativos (D5) por la salida. Por último, en el ciclo 3 saldrá el dato almacenado en el registro D6.

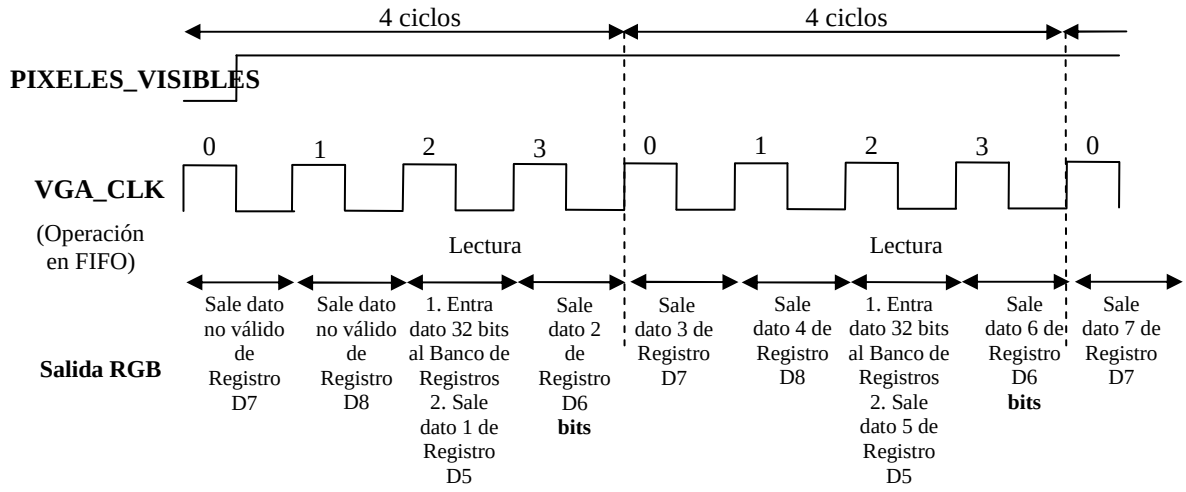


Figura 107. Cronograma que muestra la salida de datos de la arquitectura general. Para ello se emplean el Banco de Registros de Salida y un multiplexor 2 a 1.

Nota:

- El número de pixeles visibles de una línea es de 640 pixeles. Este número es exactamente divisible por 4 (640÷4 = 120). Por tanto, por cada línea que haya que pintar en pantalla se repetirá la secuencia completa de la figura 107 en 120 ocasiones.

3.3.5.4 Contadores de lectura y escritura

Los Contadores de Lectura y Escritura son idénticos: síncronos ascendentes de 17 bits, de módulo 76800[*1] y con reset asíncrono. Ambos tienen como señal de reloj el reloj procedente de la cámara *PCLK* y como entrada de reset la operación OR entre las señales *RST* (reset general del circuito) y *VSYN* (señal de sincronismo vertical de la cámara que indica en alta el comienzo de un nuevo frame). Una vez activa la señal de habilitación (*CE*) la señal de cuenta del contador comienza a contar de forma ascendente tomando como primer valor de cuenta el 0 y hasta llegar a 76799.

Nota:

1. Un frame capturado por la cámara tiene 640 (líneas visibles) × 480 (pixeles visibles) = 307200 píxeles visibles. Puesto que cada píxel es un dato de 8 bits y la memoria almacena datos de 32 bits, sólo será necesario direccionar 640×480÷4 = 76.800 posiciones de memoria.

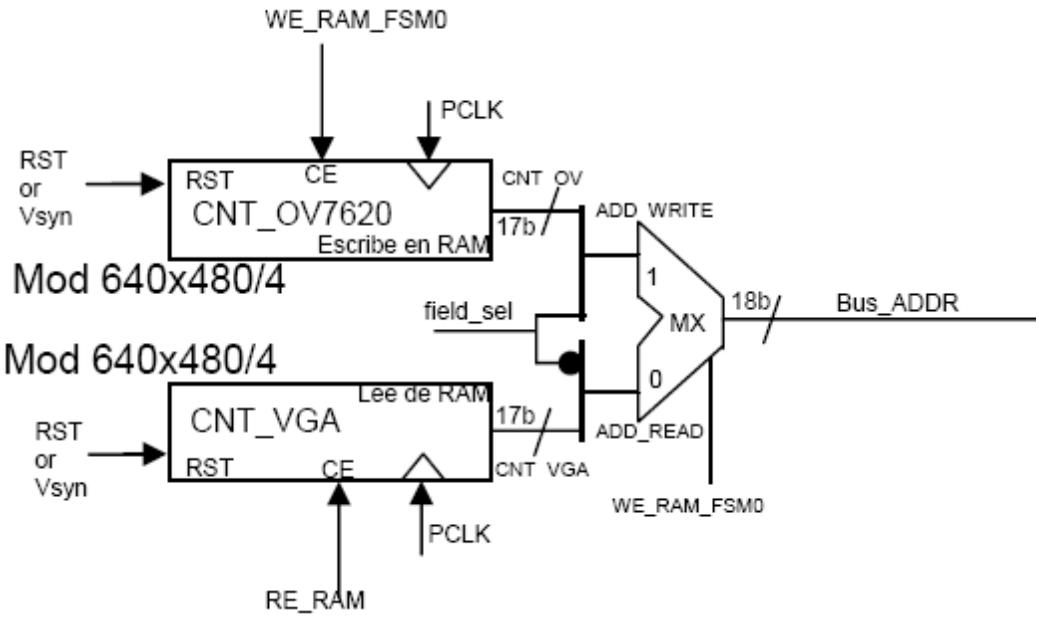


Figura 108. Conexionado de los contadores de Lectura y Escritura

La diferencia entre ambos contadores radica en sus señales de habilitación (ver figura 108). Ambos conectan su señal de cuenta (*CNT_OV* el contador de Escritura y *CNT_VGA* el de Lectura) a un multiplexor que los lleva al bus de direcciones de la memoria SRAM Asíncrona.

El Contador de Lectura determina la dirección de memoria de la que se va a leer. Como señal de habilitación tiene la señal *RE_RAM* generada por el módulo FIFO, que se activa como máximo 3 de cada 4 ciclos de reloj como queda reseñado en la figura 109.

El Contador de Escritura se encarga de determinar la dirección de memoria en la que se va guardar el dato de información. Como señal de habilitación tiene la señal *WE_RAM_FSM0* de la máquina de estados FSM0. Esta señal se activa 1 de cada 4 ciclos de reloj.

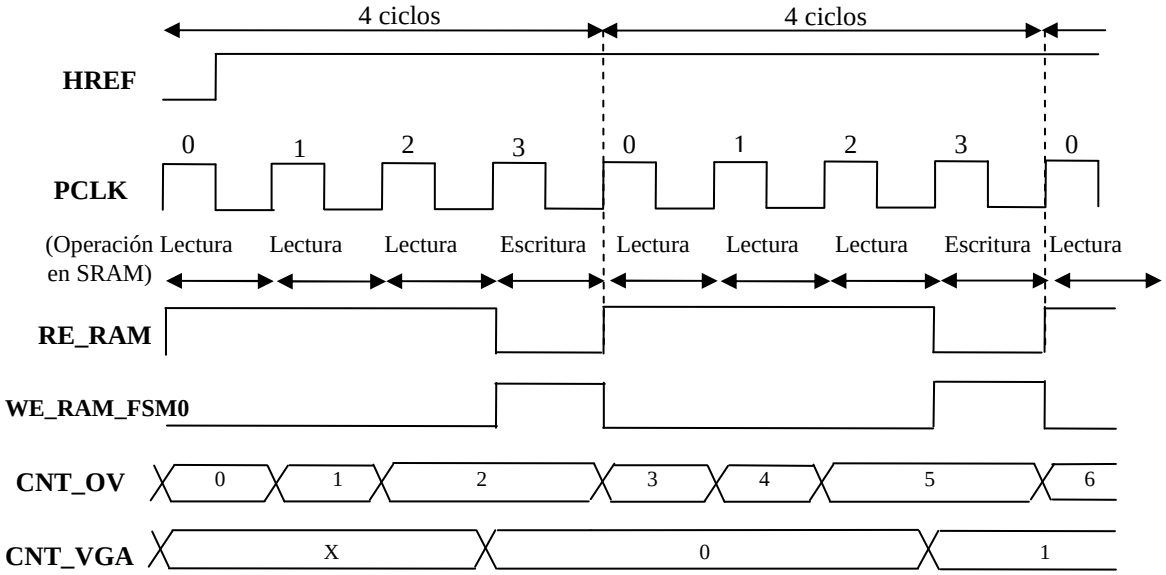


Figura 109. Cronograma de los contadores de Lectura (CNT_VGA) y de Escritura (CNT_OV)

3.3.5.5 Convertidor ADV7125

Las señales de salida de la arquitectura (*R, G, B, BLANK_CONVERTIDOR, SYNC_CONVERTIDOR, VSYN_VGA* y *HSYN_VGA*) componen una señal de vídeo VGA. Para mostrar la información en el monitor se han llevado estas señales al convertidor ADV7125 de la placa que, con su salida conectada a un conector DB15 macho, proporciona las señales analógicas que hacen funcionar dicho monitor. A continuación, se presentan las características más relevantes del datasheet del convertidor.

Figura 110. Salida de vídeo RGB. Convertidor ADV7125 a conector DB15 macho

Se trata de un convertidor Digital/Analógico de Triple velocidad. Está fabricado con tecnología CMOS y compuesto por tres convertidores de 8 bits con salidas complementarias, una entrada TTL estándar y una circuitería para el voltaje de referencia. Sus señales de entrada son las que componen la señal VGA más una entrada de reloj que conectaremos a VGA_CLK.

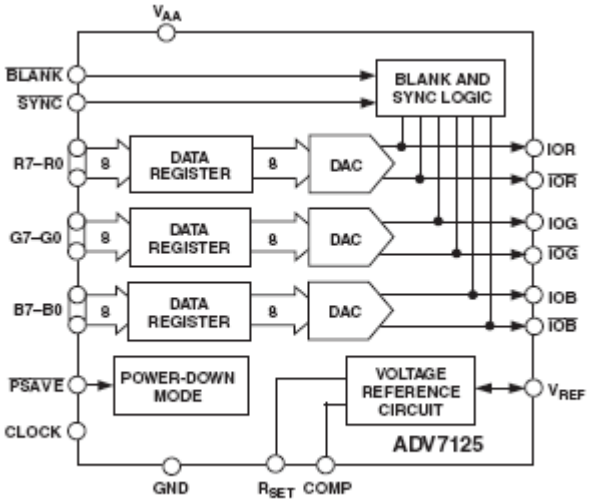


Figura 111. Diagrama de bloques del convertidor Digital-Analógico ADV7125

La siguiente tabla presenta la descripción de sus pines:

Nombre del Puerto	**Dirección**	**Descripción**
GND	Masa	Masa del chip.
R0-R7,G0-G , B0-B7	Entrada	Entradas de datos R,G y B. El dato (píxel) es registrado en el flanco ascendente de la señal de reloj. R0, G0 y B0 son los bits menos significativos.
/BLANK	Entrada	Entrada compuesta de control de *blank*. Un cero en esta entrada de control pone las salidas IOR, IOB e IOG a nivel *blanking*. Se registra en el flanco ascendente de la señal de reloj. Cuando /BLANK está a cero los pixeles de entrada R0-R7, G0-G7 y B0-B7 son ignorados.
/SYNC	Entrada	Entrada compuesta de control de sincronía. Un cero en esta entrada desconecta la fuente de corriente. Está internamente conectada a la salida analógica IOG. /SYNC no anula ninguna señal de control o entrada de datos, por lo que, debe activarse durante el intervalo de *blanking*. Se registra en el flanco ascendente de la señal de reloj. Si el canal verde no necesita información de sincronismo, /SYNC debe ponerse a cero.
V_{AA}	Entrada	Fuente de alimentación analógica. Todos los pines V_{AA} del ADV7125 tienen que estar conectados.
CLOCK	Entrada	Señal de reloj de entrada. En su flanco ascendente se registran las entradas de control /SYNC y /BLANK y las de datos R0-R7, G0-G7 y B0-B7.
/IOR,/ IOG,/ IOB	Salida	Salidas de corriente diferencial Roja, Verde y Azul. Si no son necesarias, estas salidas complementarias deben conectarse a masa.
IOR, IOG, IOB	Salida	Salidas de corriente Roja, Verde y Azul. Están capacitadas para conectarse a un cable coaxial de

		75Ω. Todas estas señales deben tener la misma carga de salida.
COMP	Salida	Pin de compensación para el amplificador interno de referencia.
VREF	Entrada	Voltaje de referencia de entrada para los DACs o voltaje de referencia para las salidas.
RST	Entrada	Señal de reset.
/PSAVE	Entrada	Pin de control de potencia. Cuando se pone en alta se reduce la potencia consumida por el ADV7125

La relación temporal entre las entradas digitales y las salidas analógicas se refleja en la figura 112.

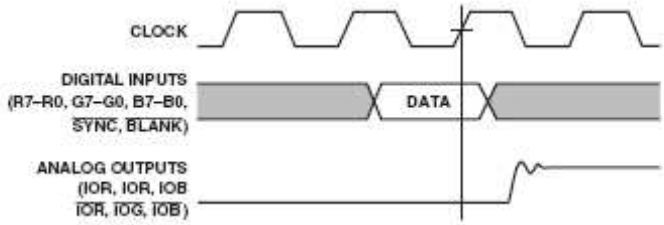


Figura 112. Entrada / Salida de vídeo

Por último, se muestra la señal de salida de vídeo RGB del convertidor:

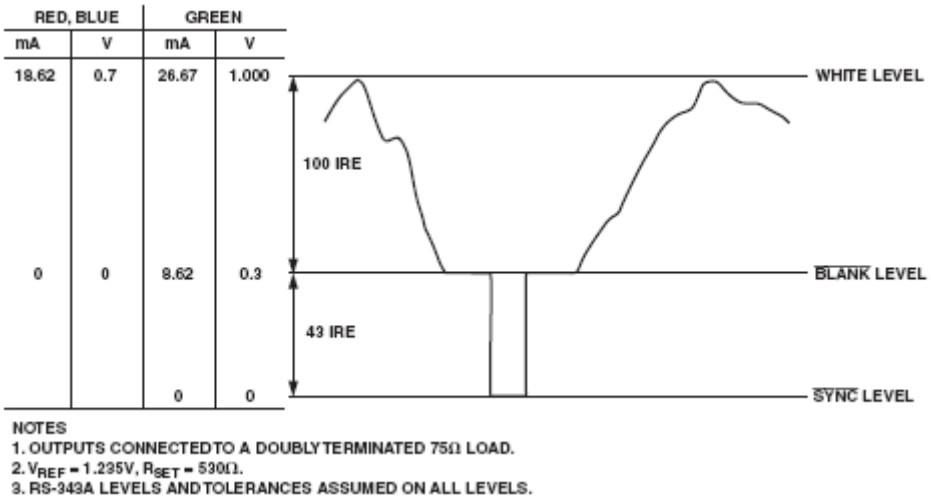


Figura 113. Salida analógica de vídeo

3.4 SIMULACIONES

En este apartado describiremos la simulación de un testbench realizado en "ModelSim", analizando e interpretando las señales más significativas de nuestra arquitectura. Simularemos la entrada de datos a la memoria SRAM, la generación de las señales de sincronismo VGA y la salida de información RGB a través de la cola FIFO.

A continuación, mostramos el código VHDL del testbench simulado:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY toplevel_tb_vhd IS
END toplevel_tb_vhd;

ARCHITECTURE behavior OF toplevel_tb_vhd IS
	COMPONENT toplevel
	PORT(
		vga_clk : IN std_logic;
		RST_button : IN std_logic;
		HREF : IN std_logic;
		VSYN : IN std_logic;
		Y_ov : IN std_logic_vector(7 downto 0);
		FODD : IN std_logic;
		pclk : IN std_logic;
		pulsador_reset_camara : IN std_logic;
		Bus_data : INOUT std_logic_vector(31 downto 0);
		Bus_addr : OUT std_logic_vector(17 downto 0);
		ce : OUT std_logic;
		we : OUT std_logic;
		oe : OUT std_logic;
		be : OUT std_logic_vector(3 downto 0);
		reset_camara : OUT std_logic;
		R : OUT std_logic_vector(7 downto 0);
		G : OUT std_logic_vector(7 downto 0);
		B : OUT std_logic_vector(7 downto 0);
		Datos_tb: out  std_logic_vector(31 downto 0);
		pixeles_visibles_tb :OUT std_logic;
		WE_RAM_FSM0_tb :OUT std_logic;
		RE_RAM_FSM0_tb :OUT std_logic;
		RE_RAM_tb :OUT std_logic;
		CE_R1_FSM0_tb :OUT std_logic;
		CE_R2_FSM0_tb :OUT std_logic;
		CE_R3_FSM0_tb :OUT std_logic;
		RE_FIFO_VGA_tb :OUT std_logic;
		CE_BR_VGA_tb :OUT std_logic;
		SEL_BR_VGA_tb :OUT std_logic_vector(1 downto 0);
		Sync_convertidor : OUT std_logic;
		Blank_convertidor : OUT std_logic;
		Vsyn_VGA_out : OUT std_logic;
		HSYN_VGA_out : OUT std_logic
```

```
            );
        END COMPONENT;
        --Entradas
        SIGNAL vga_clk :  std_logic := '0';
        SIGNAL RST_button :  std_logic := '0';
        SIGNAL HREF :  std_logic := '0';
        SIGNAL VSYN :  std_logic := '0';
        SIGNAL FODD :  std_logic := '1';
        SIGNAL pclk :  std_logic := '0';
        SIGNAL pulsador_reset_camara :  std_logic := '0';
        SIGNAL Y_ov :  std_logic_vector(7 downto 0) := (others=>'0');
        --Bidireccionales
        SIGNAL Bus_data :  std_logic_vector(31 downto 0);
        --Salidas
        SIGNAL Bus_addr :  std_logic_vector(17 downto 0);
        SIGNAL Datos_tb :  std_logic_vector(31 downto 0);
        SIGNAL pixeles_visibles_tb : std_logic;
        SIGNAL WE_RAM_FSM0_tb : std_logic;
        SIGNAL RE_RAM_FSM0_tb : std_logic;
        SIGNAL RE_RAM_tb : std_logic;
        SIGNAL CE_R1_FSM0_tb : std_logic;
        SIGNAL CE_R2_FSM0_tb : std_logic;
        SIGNAL CE_R3_FSM0_tb : std_logic;
        SIGNAL RE_FIFO_VGA_tb : std_logic;
        SIGNAL CE_BR_VGA_tb : std_logic;
        SIGNAL SEL_BR_VGA_tb : std_logic_vector(1 downto 0);
        SIGNAL ce :  std_logic;
        SIGNAL we :  std_logic;
        SIGNAL oe :  std_logic;
        SIGNAL be :  std_logic_vector(3 downto 0);
        SIGNAL reset_camara :  std_logic;
        SIGNAL R :  std_logic_vector(7 downto 0);
        SIGNAL G :  std_logic_vector(7 downto 0);
        SIGNAL B :  std_logic_vector(7 downto 0);
        SIGNAL Sync_convertidor :  std_logic;
        SIGNAL pixeles_visibles :  std_logic;
        SIGNAL Blank_convertidor :  std_logic;
        SIGNAL Vsyn_VGA_out :  std_logic;
        SIGNAL HSYN_VGA_out :  std_logic;
        constant periodo_vga : time:= 39.722 ns;
        constant periodo_pclk : time:= 74.074 ns;

begin

        uut: toplevel PORT MAP(
                vga_clk => vga_clk,
                RST_button => RST_button,
                VSYN => VSYN,
                HREF => HREF,
                Y_ov => Y_ov,
                pclk => pclk,
                Bus_data => Bus_data,
                Bus_addr => Bus_addr,
                ce => ce,
                we => we,
```

```
            oe => oe,
            be => be,
            FODD => FODD,
            pulsador_reset_camara => pulsador_reset_camara,
            reset_camara => reset_camara,
            R => R,
            G => G,
            B => B,
            Datos_tb=>Datos_tb,
            pixeles_visibles_tb => pixeles_visibles_tb,
            WE_RAM_FSM0_tb => WE_RAM_FSM0_tb,
            RE_RAM_FSM0_tb => RE_RAM_FSM0_tb,
            RE_RAM_tb => RE_RAM_tb,
            CE_R1_FSM0_tb => CE_R1_FSM0_tb,
            CE_R2_FSM0_tb => CE_R2_FSM0_tb,
            CE_R3_FSM0_tb => CE_R3_FSM0_tb,
            RE_FIFO_VGA_tb => RE_FIFO_VGA_tb,
            CE_BR_VGA_tb => CE_BR_VGA_tb,
            SEL_BR_VGA_tb => SEL_BR_VGA_tb,
            Sync_convertidor => Sync_convertidor,
            Blank_convertidor => Blank_convertidor,
            Vsyn_VGA_out => Vsyn_VGA_out,
            HSYN_VGA_out => HSYN_VGA_out
);

--Señales de reloj
            vga_clk<=not vga_clk after periodo_vga/2;  --reloj periódico
            pclk<=not pclk after periodo_pclk/2;  --reloj periódico

tb : PROCESS
BEGIN
             HREF <= '0';
             RST_button <='1';
             wait for 10*periodo_pclk;
             RST_button <='0';
             wait for 10*periodo_pclk;

             HREF <= '1'; --1er sincronismo de la CNN (particular)
             wait for 639*periodo_pclk;
             HREF <= '0';
             wait for 218*periodo_pclk;
             for i in 1 to 478 loop
               HREF <= '1';
               wait for 640*periodo_pclk;
               HREF <= '0';
               wait for 218*periodo_pclk;
             end loop;
             wait for 45*858*periodo_pclk;

             for j in 1 to 4 loop -- 4 sincronismos completos de la CNN
                          for k in 1 to 480 loop
                                       HREF <= '1';
                              wait for 640*periodo_pclk;
                                       HREF <= '0';
                              wait for 218*periodo_pclk;
```

```
                        end loop;
                    wait for 45*858*periodo_pclk;
                end loop;


            wait;
        END PROCESS;
        PROCESS -- Proceso para la entrada de datos
        BEGIN
                wait for periodo_pclk/2;
                for h in 1 to 525*858*4 loop
                        Y_ov <= "00000000";
                        wait for periodo_pclk;
                        Y_ov <= "00000001";
                        wait for periodo_pclk;
                        Y_ov <= "00000010";
                        wait for periodo_pclk;
                        Y_ov <= "00000011";
                        wait for periodo_pclk;
                end loop;

            wait;
        END PROCESS;

END;
```

Características del testbench:

Se ha proporcionado a las entradas *VSYN y HREF* la misma forma y duración que adquirirá cuando nuestra arquitectura esté conectada a la cámara (ver figura 114).

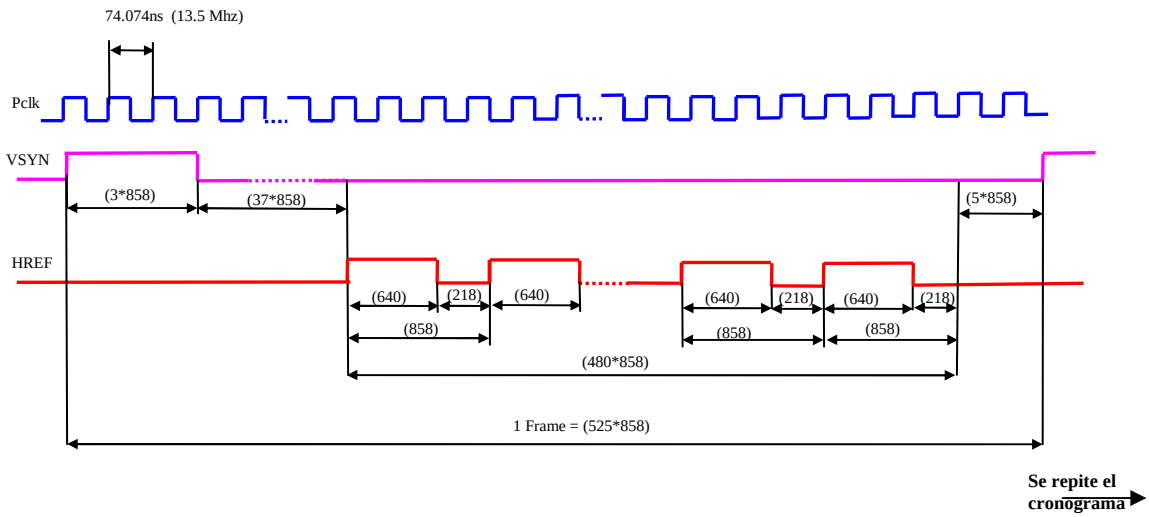


Figura 114. Forma y duración de la señal de sincronismo horizontal procedente de la cámara OV7620.

- En cuanto a la entrada de datos hemos creado un bucle, de manera que el bus de entrada cambie su valor cada ciclo de reloj *PCLK*, siguiendo la siguiente secuencia: “00000000”, “00000001”, 00000010” y 00000011”.

- La memoria SRAM Asíncrona está integrada en la FPGA, por lo que no es un componente de VHDL y no podemos simular su funcionamiento. En lugar de ello, analizaremos los valores que adquirirán sus señales de control y buses de datos y direcciones.
-

A continuación procedemos a describir las simulaciones mencionadas anteriormente.

3.4.1 Entrada de datos

La figura 115 muestra los elementos de la arquitectura implicados en la entrada de datos, mientras que la figura 116 presenta el valor que van a ir adquiriendo las señales de control implicadas.

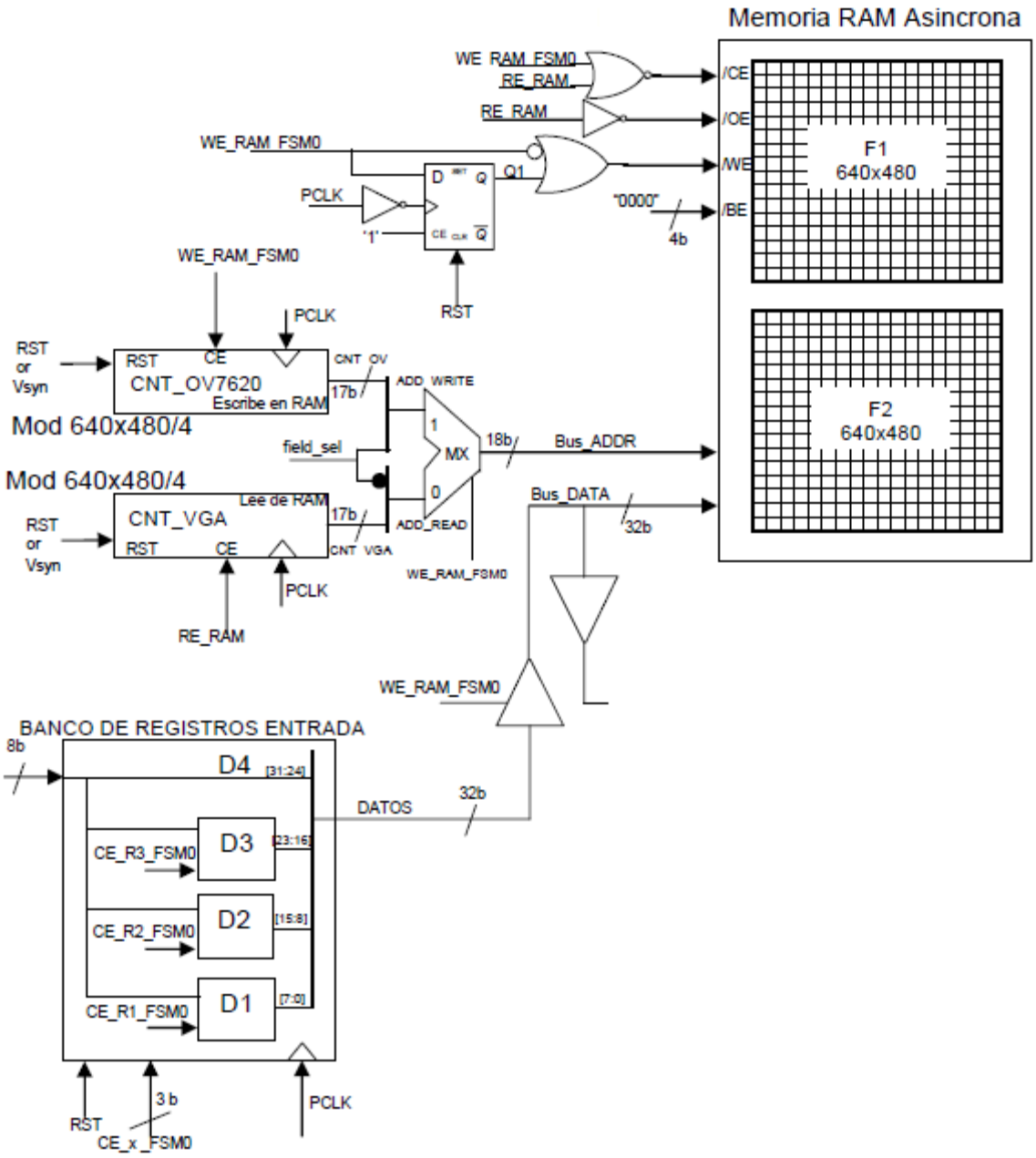


Figura 115. Arquitectura para la entrada de datos en la memoria SRAM Asíncrona

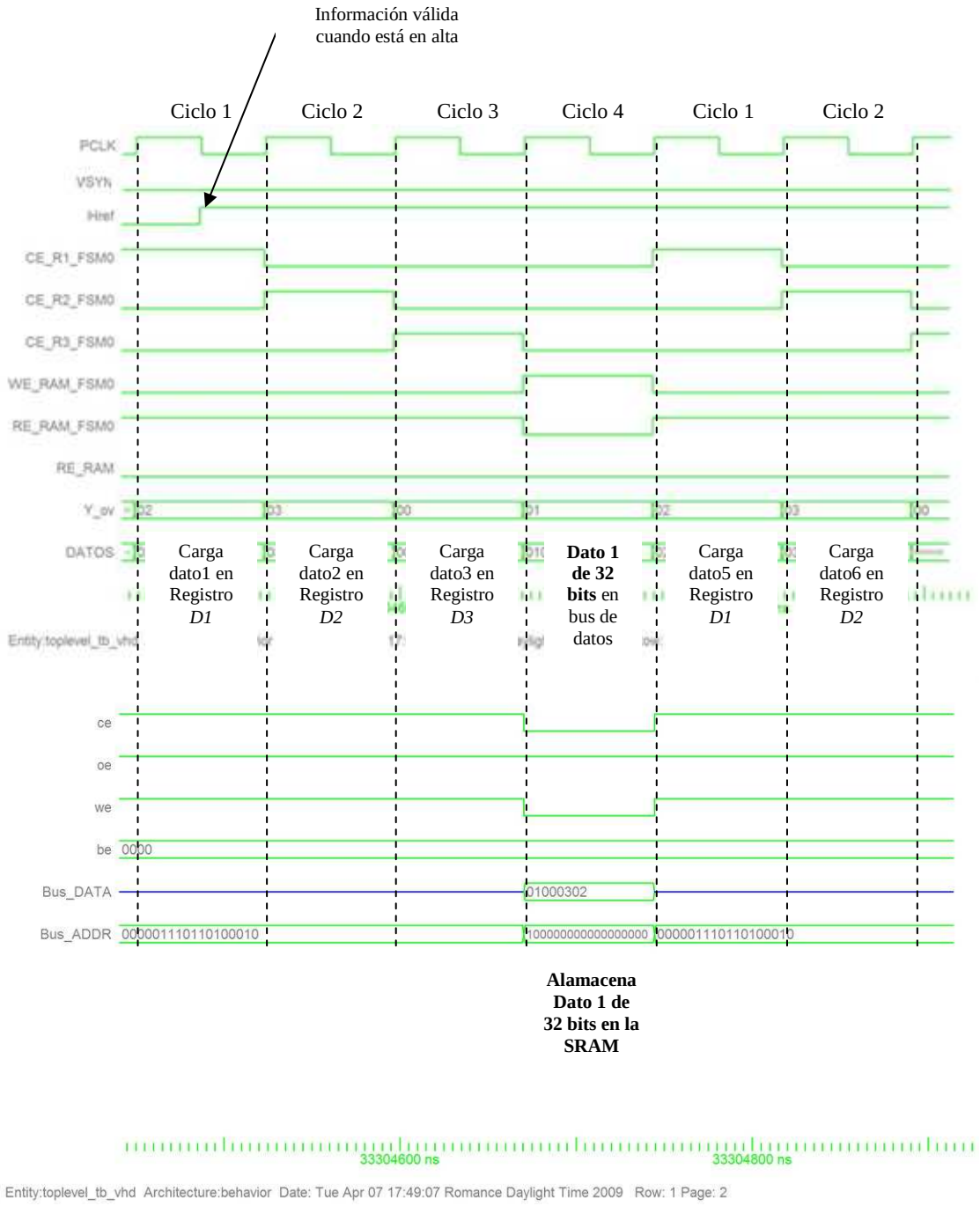


Figura 116. Cronograma de la entrada de datos en la memoria SRAM procedentes de la cámara.

Cuando la cámara proporciona información válida (información visible) se cumplen las condiciones para la puesta en marcha de la máquina de estados FSM0, generando las señales necesarias para controlar la escritura en memoria. En la figura 116 se observa la carga de la información válida en el Banco de Registros de Entrada, con la activación en los tres primeros ciclos de las señales *CE_R1_FSM0*, *CE_R2_FSM0* y *CE_R3_FSM0* respectivamente. El cuarto ciclo es el de Escritura en memoria. En este ciclo se permite el paso de la información al bus de datos *Bus_Datos* de la SRAM y se establece el valor de sus señales de control en la configuración de "Escribe todos los bits en memoria"*[1]. Por tanto, en esta simulación el primer dato de

información procesada será almacenado en la posición de memoria "1000000000000000", que corresponde a la primera posición de la zona 2 de memoria*[2].

Notas:

1. Configuración para escribir en la memoria SRAM Asíncrona:

/CE	/OE	/WE	/BLE	/BHE	$I/O_0 – I/O_7$	$I/O_8 – I/O_{15}$	Modo	Alimentación
L	X	L	L	L	Entrada de datos	Entrada de datos	Escribe todos los bits	Activa (I_{CC})

2. Recordar que la memoria SRAM está dividida en dos zonas mediante el bit que representa la señal *field_sel*.
3. El cronograma de la figura 116 se repite mientras la información que tenemos a la entrada de la arquitectura sea válida.

3.4.2 Generación de sincronismos VGA

En esta sección comprobamos como la señal de sincronismo vertical (VSYN) que proporciona la cámara actúa como entrada para generar los sincronismos de salida en estándar VGA. Las siguientes figuras muestran la arquitectura y los resultados de su simulación.

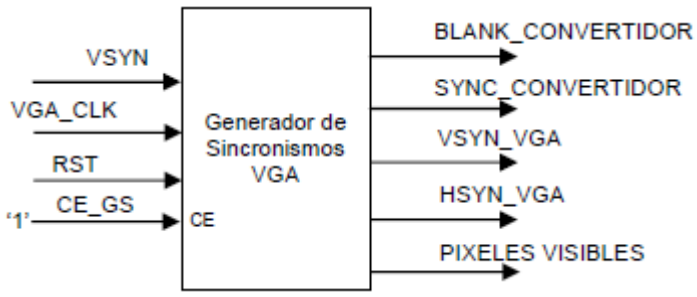


Figura 117. Generador de Sincronismos VGA

Notas:

1. La forma y duración de las señales que vamos a analizar han sido expuestas con detalle en el Capítulo 3. Por tanto, en estudio de las simulaciones nos limitaremos a indicar los tiempos pero no a explicarlos.

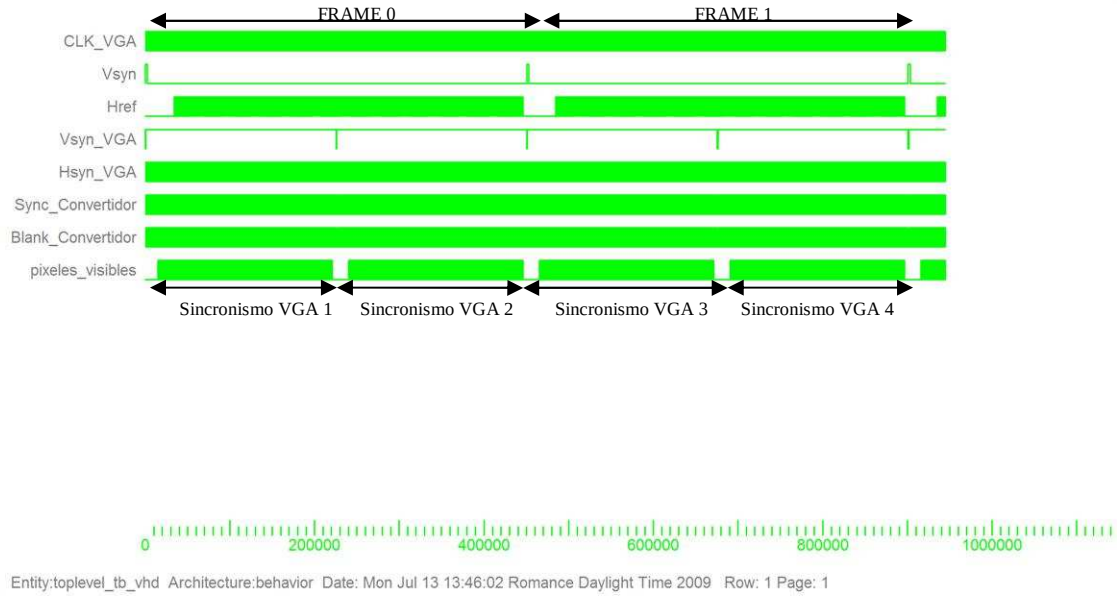


Figura 118. Cronograma de los sincronismos horizontal y vertical tanto en formato Cámara OV7620 como en el estándar VGA.

En esta figura se han simulado los dos primeros frames procedentes de la cámara. Comprobamos como a partir del primer frame se pone en funcionamiento toda la arquitectura. Por cada sincronismo vertical (*VSYN*) de la cámara de la cámara se generan dos sincronismos para el estándar VGA, lo que nos permite solucionar el problema de duplicar la frecuencia en la arquitectura general.

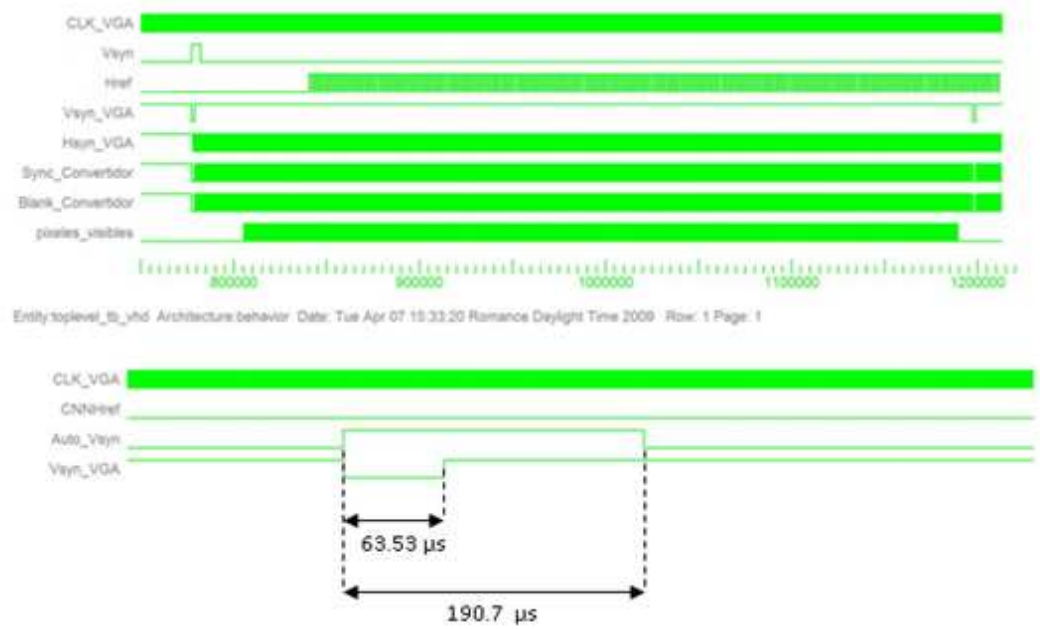


Figura 119. Arriba: sincronismos correspondientes a un frame para estándar VGA. Abajo: Duración temporal de los pulsos de mayor tamaño de las señales *VSYN* y *Vsyn_VGA* (sincronismos verticales en uno y otro estándar).

3.4.3 Salida de datos

La parte de la arquitectura general que interviene en la salida de datos queda reflejada en la figura 120.

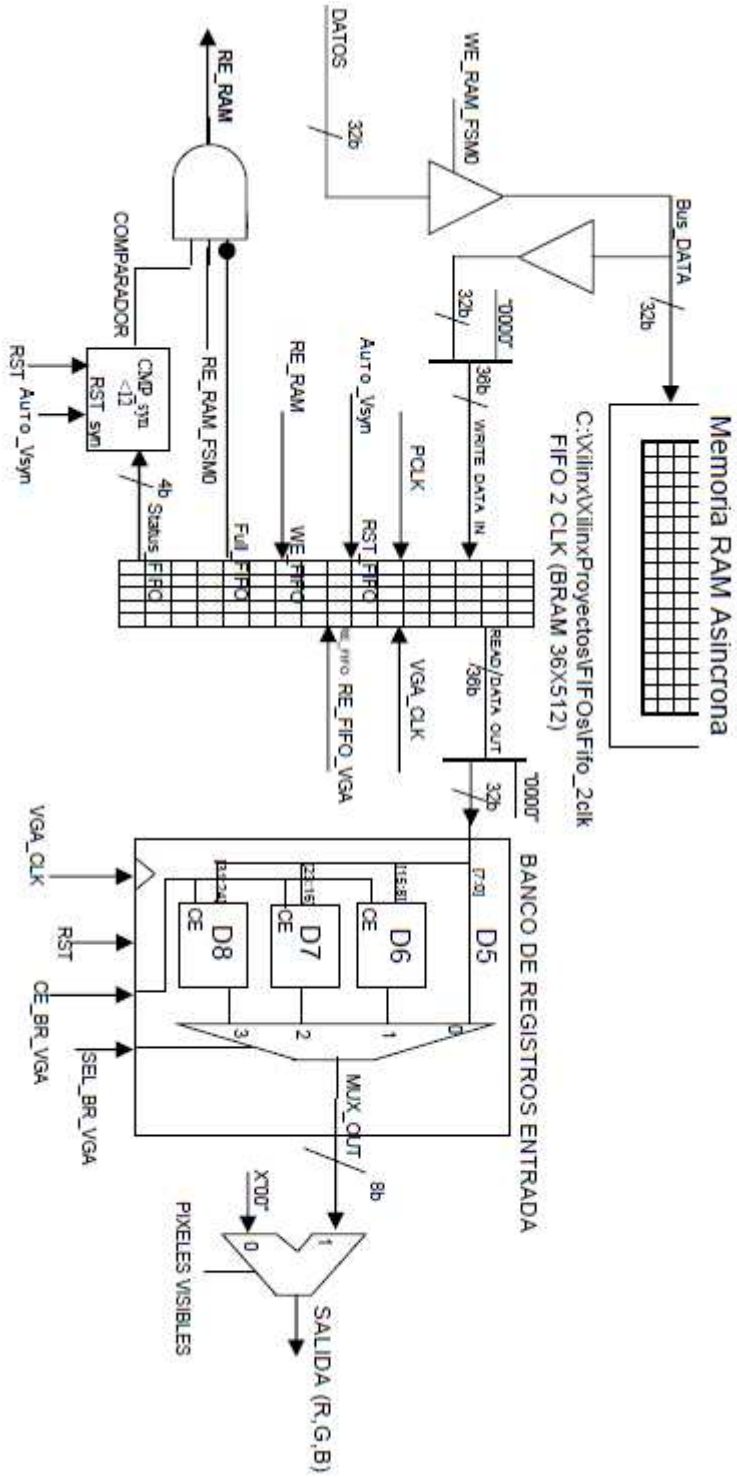


Figura 120. Arquitectura salida de datos

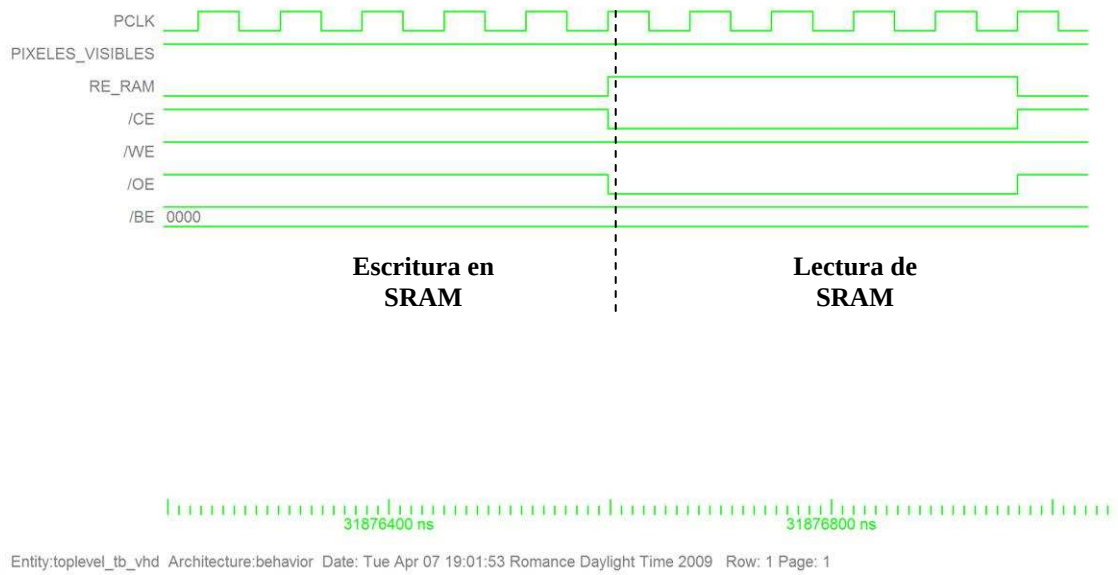


Figura 121. Cronograma de la salida de datos de la memoria SRAM hacia la cola FIFO.

La figura 121 representa la entrada de datos en la cola FIFO procedentes de la memoria SRAM. Cuando la señal *RE_RAM* se activa, la memoria configura sus señales en modo "Lee todos los bits de memoria"*[1] y la cola FIFO admitirá la entrada de datos.

1. Configuración para leer de la memoria SRAM Asíncrona:

/CE	/OE	/WE	/BLE	/BHE	$I/O_0 – I/O_7$	$I/O_8 – I/O_{15}$	Modo	Alimentación
L	L	H	L	L	Salida datos	Salida datos	Lee todos los bits	Activa (I_{CC})

La última figura presenta los valores que adquieren las señales de control que genera la máquina de estados FSM1. Repiten una secuencia, en condiciones de información visible (*PIXELES_VISIBLES* en alta), que comienza con la lectura de un dato de la FIFO (*RE_FIFO_VGA* en alta) y, en el siguiente ciclo el dato leído se registra en el Banco de Registros de Salida. En el mismo ciclo el multiplexor se configura para sacar la información correspondiente a los 8 LSB del dato de 32 bits leído de la FIFO (*SEL_BR_VGA* = "00"). Los otros dos ciclos de la secuencia continúan sacando la información por la salidas R,G, y B, ya que el multiplexor 2 a 1 de salida la deja salir (*PIXELES_VISIBLES* en alta). En el primer ciclo también se produce la salida de los 8 MSB del dato anteriormente cargado.

Observamos que las señales de información (*Bus_DATA*, *R*, *G* y *B*) toman un valor desconocido en la simulación. Esto se debe a que la RAM no está siendo simulada, sólo sus señales de control.

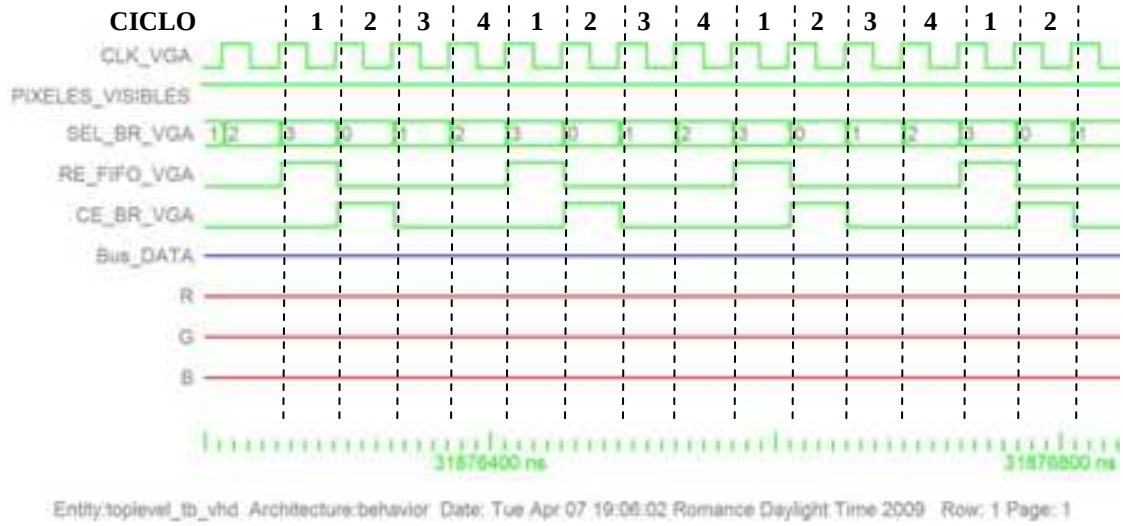


Figura 122. Cronograma de la salida de la información RGB procedente de la FIFO.

3.5 IMPLEMENTACIÓN

En esta sección, vamos a analizar las características de velocidad y consumo de área al volcar nuestra arquitectura en la FPGA

3.5.1 Características de velocidad

En nuestro caso las restricciones temporales nos las imponen las dos señales de reloj:

- PCLK: señal de reloj procedente de la cámara OV7620. Su frecuencia es de 13.5 Mhz. Es utilizado por la zona 1 de la arquitectura.
- VGA_CLK: señal de reloj que genera un oscilador interno de la FPGA. Su frecuencia es de 25.175 Mhz. Es utilizado por la zona 2 de la arquitectura.

3.5.2 Características de área

El consumo de recursos de nuestra arquitectura en la FPGA queda reflejado en el sumario de utilización de dispositivos del proyecto en XILINX. Su contenido se encuentra en la figura 123. Los aspectos más relevantes en cuanto al porcentaje de utilización de los recursos de la FPGA son el consumo de slices y de Block RAMs.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	256	26,624	1%	
Number of 4 input LUTs	339	26,624	1%	
Logic Distribution				
Number of occupied Slices	295	13,312	2%	
Number of Slices containing only related logic	295	295	100%	
Number of Slices containing unrelated logic	0	295	0%	
Total Number 4 input LUTs	433	26,624	1%	
Number used as logic	339			
Number used as a route-thru	94			
Number of bonded IOBs	144	333	43%	
IOB Flip Flops	8			
Number of Block RAMs	1	32	3%	
Number of GCLKs	2	8	25%	
Total equivalent gate count for design	70,775			
Additional JTAG gate count for IOBs	6,912			

Figura 123. Consumo de área de la FPGA

En la implementación de nuestro proyecto hemos utilizado 295 slices, lo que supone únicamente un 2% de los 13.312 disponibles. También se han empleado un 3% de los Block RAMs de la FPGA. En conclusión, la cantidad de recursos de la FPGA consumidos por nuestro proyecto es muy pequeña.

CAPÍTULO 4. Resultados

La configuración final del sistema está compuesta por la cámara digital C3188A con sensor de imagen CMOS OV7620, una FPGA Spartan 3 de Xilinx, en su modelo XC3S1500-FG456 y un monitor TFT de 15” de la marca DELL.

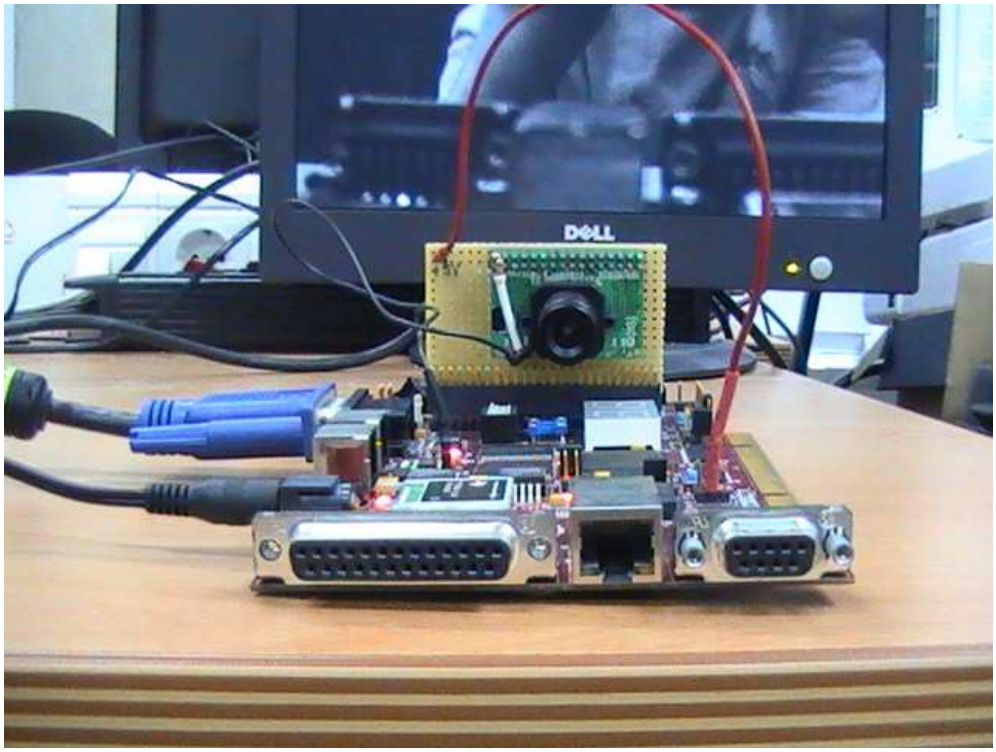


Figura 124. Vista de alzado del sistema

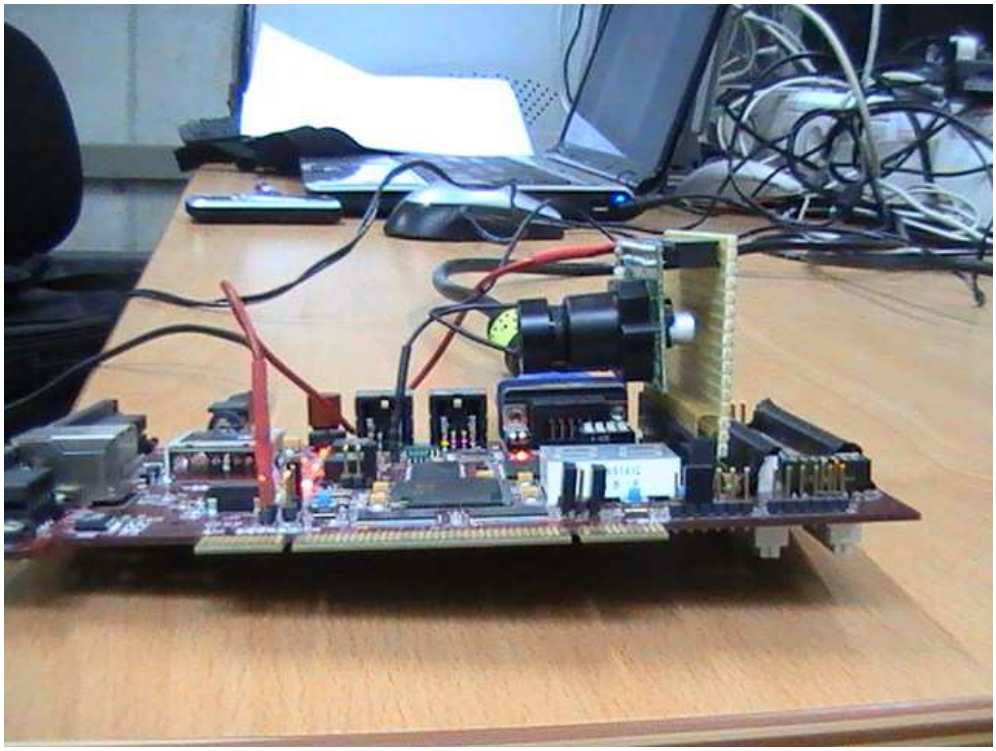


Figura 125. Vista de perfil del sistema

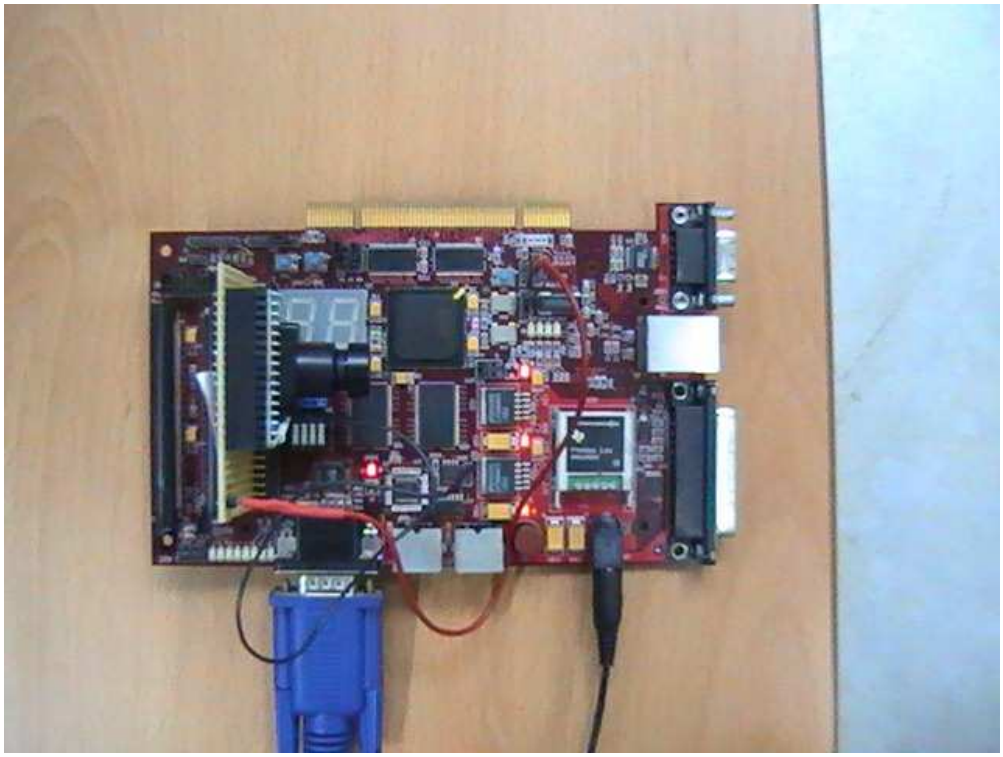


Figura 126. Vista de planta del sistema

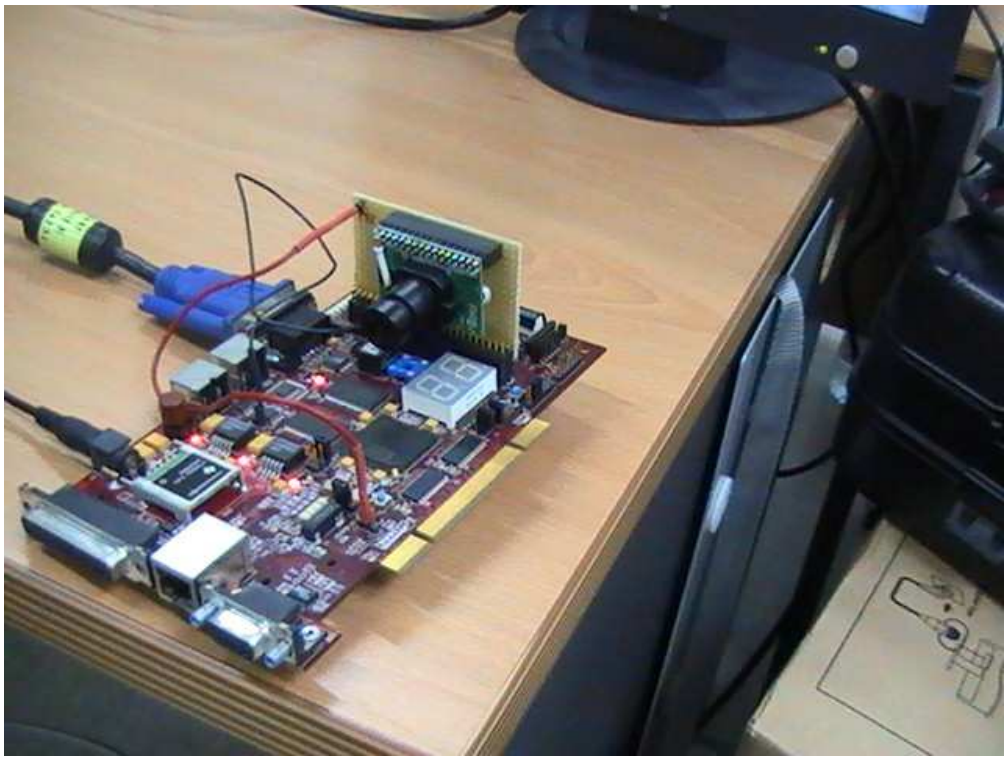


Figura 127. Vista del sistema

Las imágenes capturadas por la cámara son conducidas a la FPGA, que las adapta en resolución y frecuencia para poder mostrarlas a través del monitor VGA. A continuación, mostramos varios ejemplos de funcionamiento:

1) La cámara del sistema captura al individuo que sujeta una cámara de vídeo y lo muestra en tiempo real en el monitor. Se trata del efecto que haría un espejo.

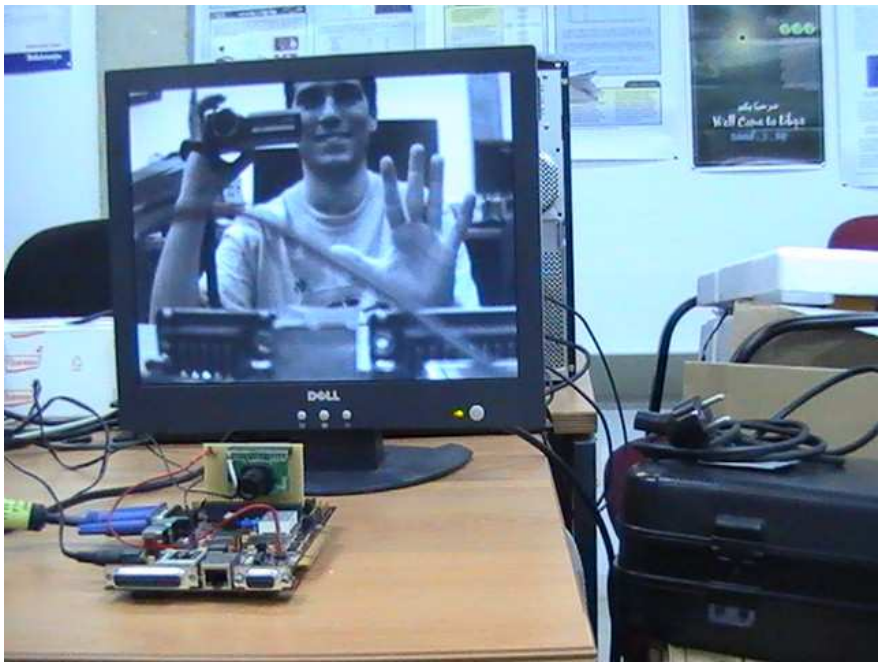


Figura 128. Ejemplo de funcionamiento 1

2) En este caso, la cámara del sistema enfoca al documento abierto en el portátil de la izquierda, mostrando su captura en el monitor de la derecha.

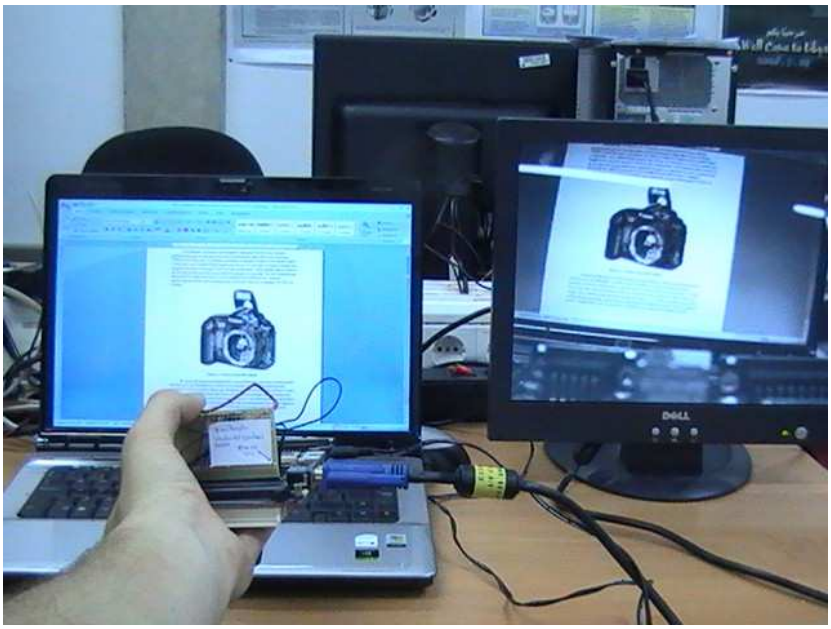


Figura 129. Ejemplo de funcionamiento 2

Por último, en el CD adjunto al proyecto se presenta un vídeo mostrando una imagen en movimiento para demostrar que se trata de un sistema de vídeo en tiempo real.

Nota: vídeo sólo disponible en la versión multimedia del proyecto.

CAPÍTULO 5. Conclusiones

5.1 Resumen del proyecto

A lo largo de este proyecto son muchos las etapas que hemos tenido que abordar. En un principio hemos realizado una labor investigadora importante, en el que se han tenido que estudiar con detalle el funcionamiento de la cámara, monitor y FPGA a utilizar. Hemos profundizado en aquellos aspectos que más nos ha interesado, por ejemplo, de la cámara hemos analizado tanto su funcionamiento interno como la forma y duración de sus señales de salida, mientras que, de la FPGA sólo hemos visto aquellos dispositivos que intervenían en la arquitectura (convertidor, memorias, etc).

Una vez superada la etapa de investigación hemos profundizado en el sistema a desarrollar. Se ha definido una arquitectura que cumple los objetivos de transformar la resolución y frecuencia de la cámara al formato VGA del monitor. Esta arquitectura ha sido programada en lenguaje VHDL y volcada en la FPGA.

La etapa de testeo de la arquitectura ha sido larga y con muchas dificultades. Puesto que la arquitectura implementada utiliza la memoria SRAM de la FPGA, no hemos podido simular su funcionamiento completo a través de la herramienta ModelSim, con lo que hemos tenido que agudizar el ingenio para realizar modificaciones en la arquitectura que nos permitiera realizar comprobaciones.

Finalmente, hemos superado todos los obstáculos que presenta un proyecto con implementación hardware y hemos logrado cumplir los objetivos. En el siguiente apartado profundizaremos más en este último aspecto.

5.2 Análisis de los resultados obtenidos

Tras observar los resultados obtenidos podemos concluir que el sistema implementado resuelve satisfactoriamente los objetivos planteados inicialmente de adaptación entre la cámara digital y el monitor. Las imágenes capturadas por la cámara se muestran perfectamente y sin vibración en el monitor aun cuando la cámara está en movimiento. Por tanto, se trata de un sistema de vídeo en tiempo real.

El sistema se caracteriza por utilizar una cámara digital que proporcione las imágenes de entrada al sistema con una frecuencia de 30 Hz. En cuanto a la salida, el único requisito que se exige es que el monitor sea compatible con el estándar VGA.

La principal ventaja de este sistema es que permite la realización de sistemas de procesamiento de vídeo en tiempo real en un espacio reducido y sin emplear un PC.

En cuanto a las limitaciones que presenta, son varias. Por una parte las que caracterizan al propio sistema, con una cámara y monitor con unas determinadas características. Por otra parte, las limitaciones de la FPGA: el tamaño de la memoria SRAM sólo nos va a permitir trabajar en blanco y negro, si quisiéramos mostrar las imágenes a color sería necesario aumentarla.

Por último, el proyecto implementado tiene infinidad de aplicaciones. Se trata de un sistema de soporte de procesamiento en tiempo real, por lo que cualquier procesamiento de las imágenes que se realice es una posible aplicación. Una posible aplicación sería realizar un procesamiento de la imagen para que obtuviese el contorno de los objetos.

5.3 Líneas futuras

A la vista de los resultados obtenidos en este proyecto, se presentan como posibles líneas de investigación:

- Se podría utilizar una FPGA con mayor memoria SRAM, lo que permitiría obtener las imágenes en color con unas pequeñas modificaciones en la arquitectura.
- También sería interesante adaptar la salida a un conector HDMI permitiendo ver los resultados en alta definición.

CAPÍTULO 6. Bibliografía

[1] Garrigós Guerrero, F. Javier, Toledo Moreo, F. Javier, Martínez Álvarez, J. Javier. "Síntesis de Sistemas Digitales con VHDL". Universidad Politécnica de Cartagena. 2003.

[2] David Galadí-Enríquez, Ignasi Ribas Canudas. "Manual Práctico de Astronomía con CCD". Ed. Omega. 1998.

[3] Steve B. Howell. "Handbook of CCD Astronomy". Cambridge University Press 2000.

[4] "VGA Timing Information.pdf"
http://209.85.229.132/search?q=cache:Vws-tqabNsJ:secmem.springnote.com/pages/2115780/attachments/955392+VGA+Timing+Information.pdf&cd=2&hl=es&ct=clnk&gl=es
Temporizaciones del estándar VGA

[5] "Proyectos 2008-2009: Controlador VGA"
http://arantxa.ii.uam.es/
Funcionamiento de un tubo de rayos catódicos

[6] "Xilinx Spartan-3 Evaluation Kit - Brief 022504F.pdf"
http://www.xilinx.com
Especificaciones de la placa Spartan 3

[7] "Xilinx Spartan-3 Evaluation Kit - User's Guide 022304F.pdf"
http://www.xilinx.com
Guía de usuario de la Spartan 3

[8] "Guía_Inicio_ISE.pdf"
http://webs.uvigo.es/mdgomez/SED/Guia_Inicio_ISE.pdf
Introducción al software ISE.

[9] "XilinxTutorial_ISE101_Parte01.pdf"
http://tourdigital.net/Tutoriales/VHDL/XilinxTutorial_ISE101_Parte01.pdf
Turorial ISE

[10] "Omnivision-ds_7620.pdf"
http://www.alldatasheet.com
Datasheet de la cámara digital Omnivision 7620.

[11] "c3188a.pdf"
http://www.alldatasheet.com
Datasheet de la placa C3188A de la cámara digital Omnivision 7620.

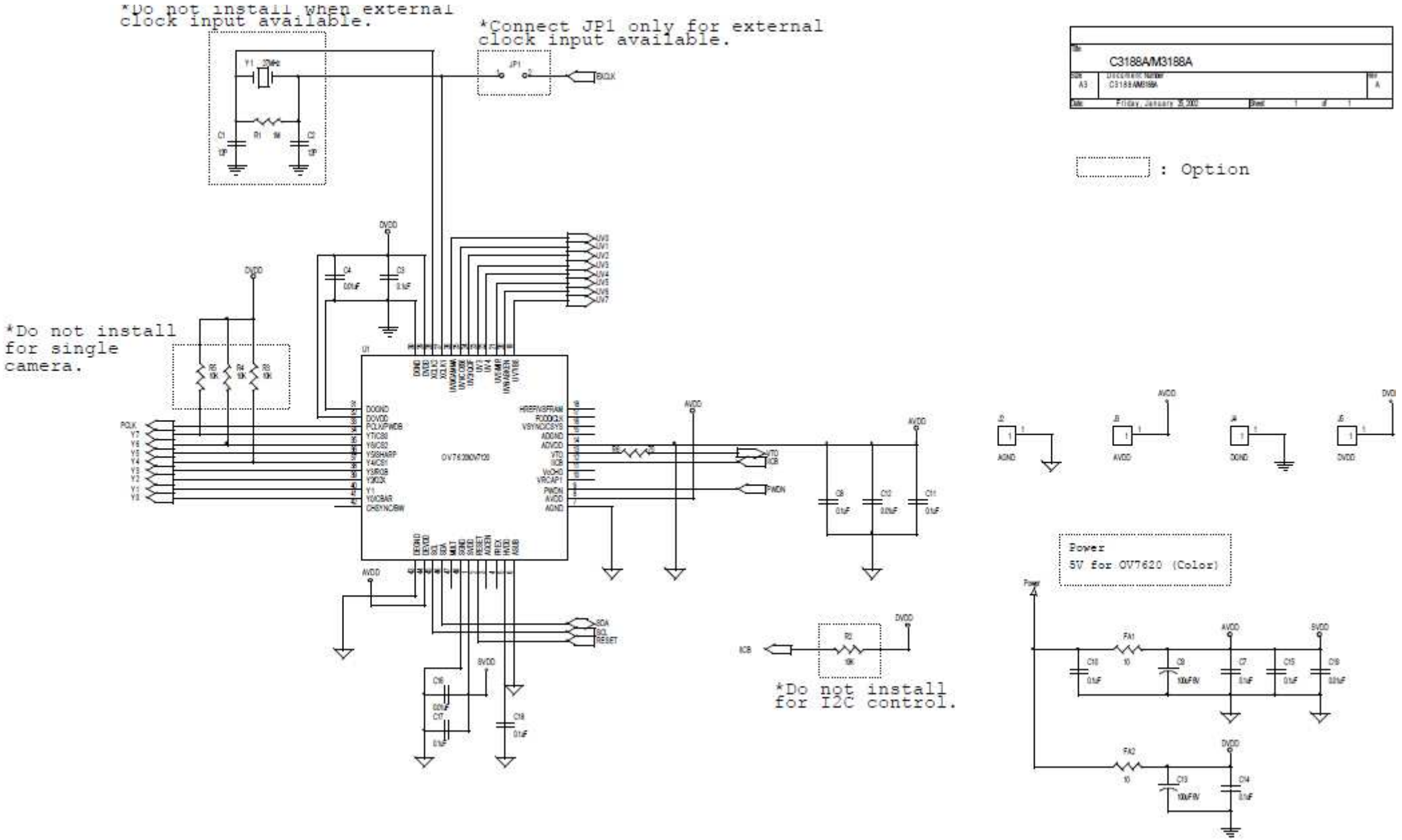
[12] "c3188a_sch.pdf"
http://www.alldatasheet.com
Esquemático de la placa C3188A de la cámara digital Omnivision 7620.

[13] "ADV7125.pdf"
http://www.datasheetsite.com/datasheet/ADV7125
Datasheet del convertidor analógico digital

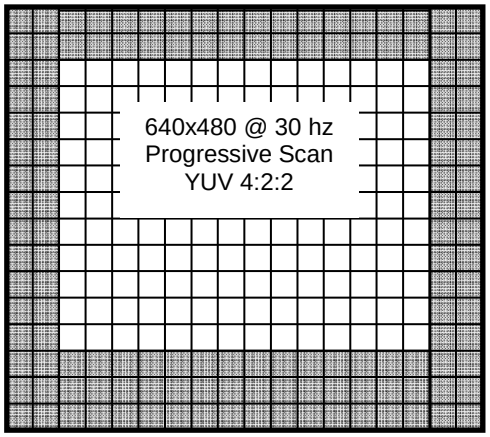
[14] "CY7C1041V33.pdf"
http://www.alldatasheet.com/datasheet-pdf/ pdf/114366/CYPRESS/ CY7C1041V33.html
Datasheet memoria SRAM

[15] "xapp131FIFO.pdf"
http://www.xilinx.com
Módulo FIFO asíncrona en lenguaje VHDL

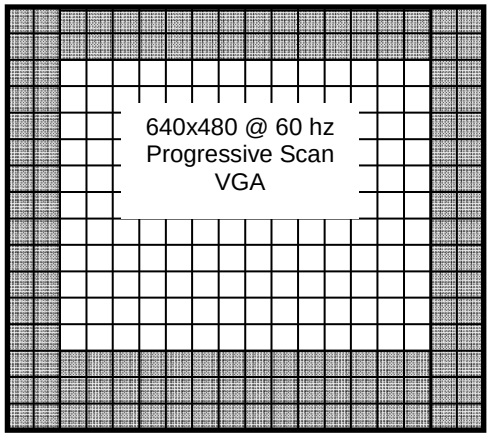
ANEXO A: Esquema de conexiones de la cámara C3188A



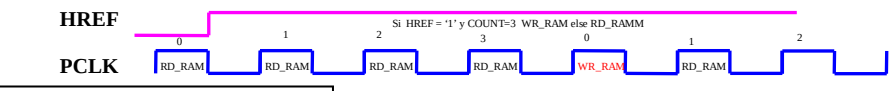
ANEXO B: Arquitectura para transformar la resolución y frecuencia de la cámara OV7620 a VGA (frames relativos a Vsyn)



I =858x525 (OV7620)

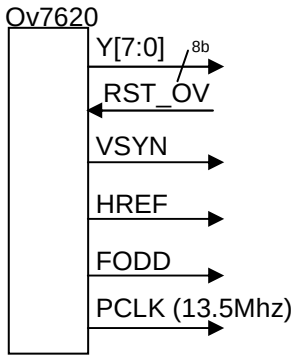


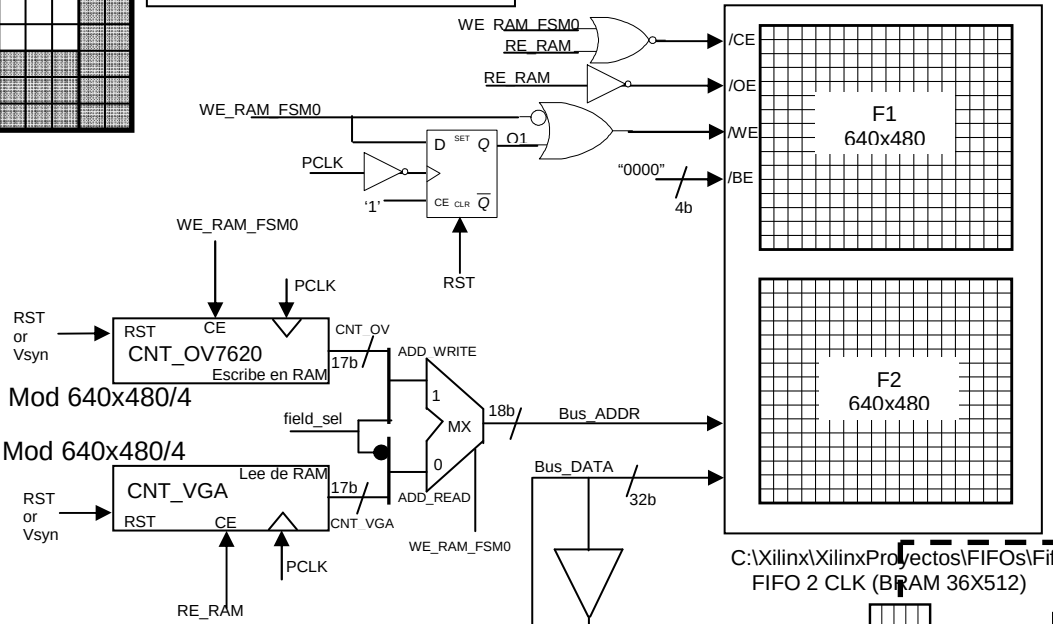
I =800x525 (VGA)

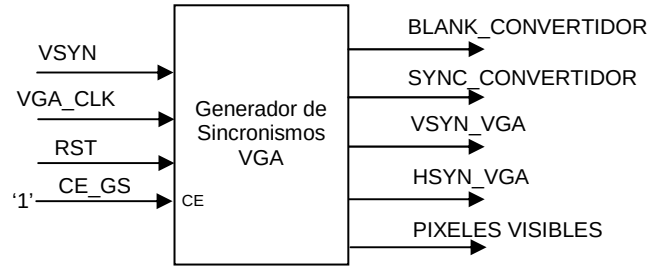


RE_RAM='1' (habilita la lectura)
WE_RAM='1' (habilita la escritura)

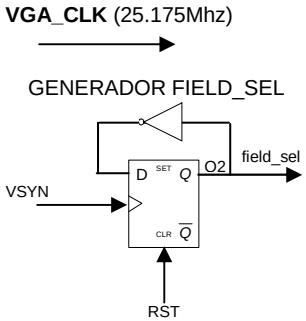
Memoria RAM Asíncrona (2xCY7C1041C33)

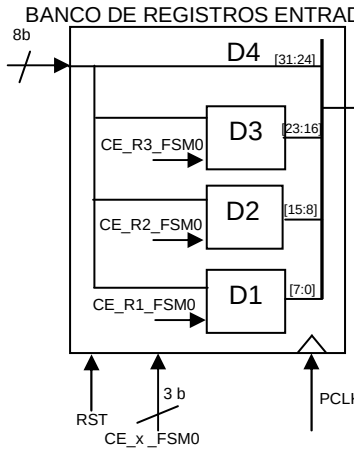


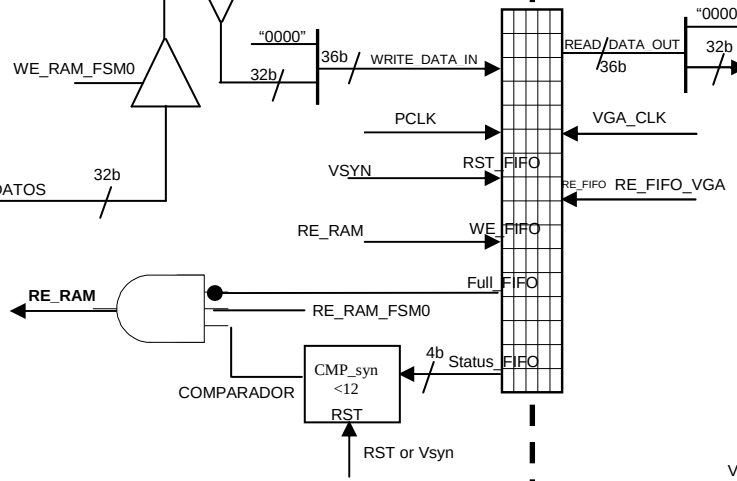


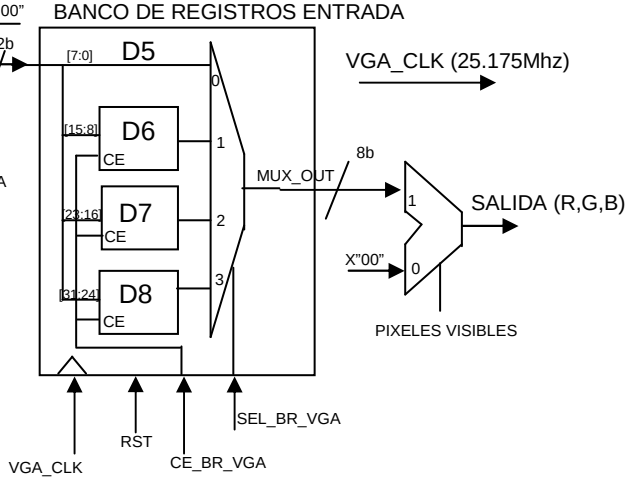


C:\Xilinx\XilinxProyectos\FIFOs\Fifo_2clk
FIFO 2 CLK (BRAM 36X512)





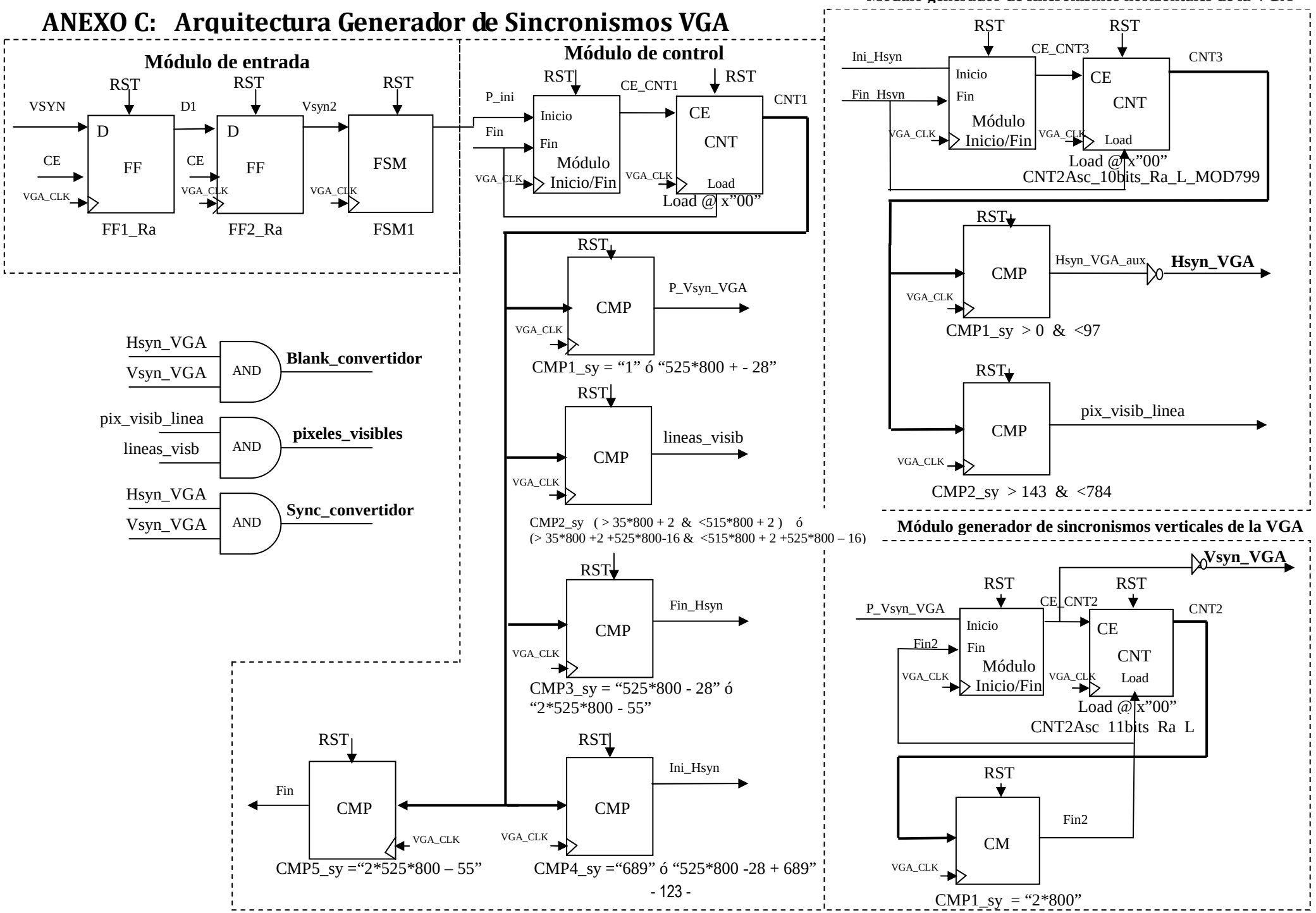




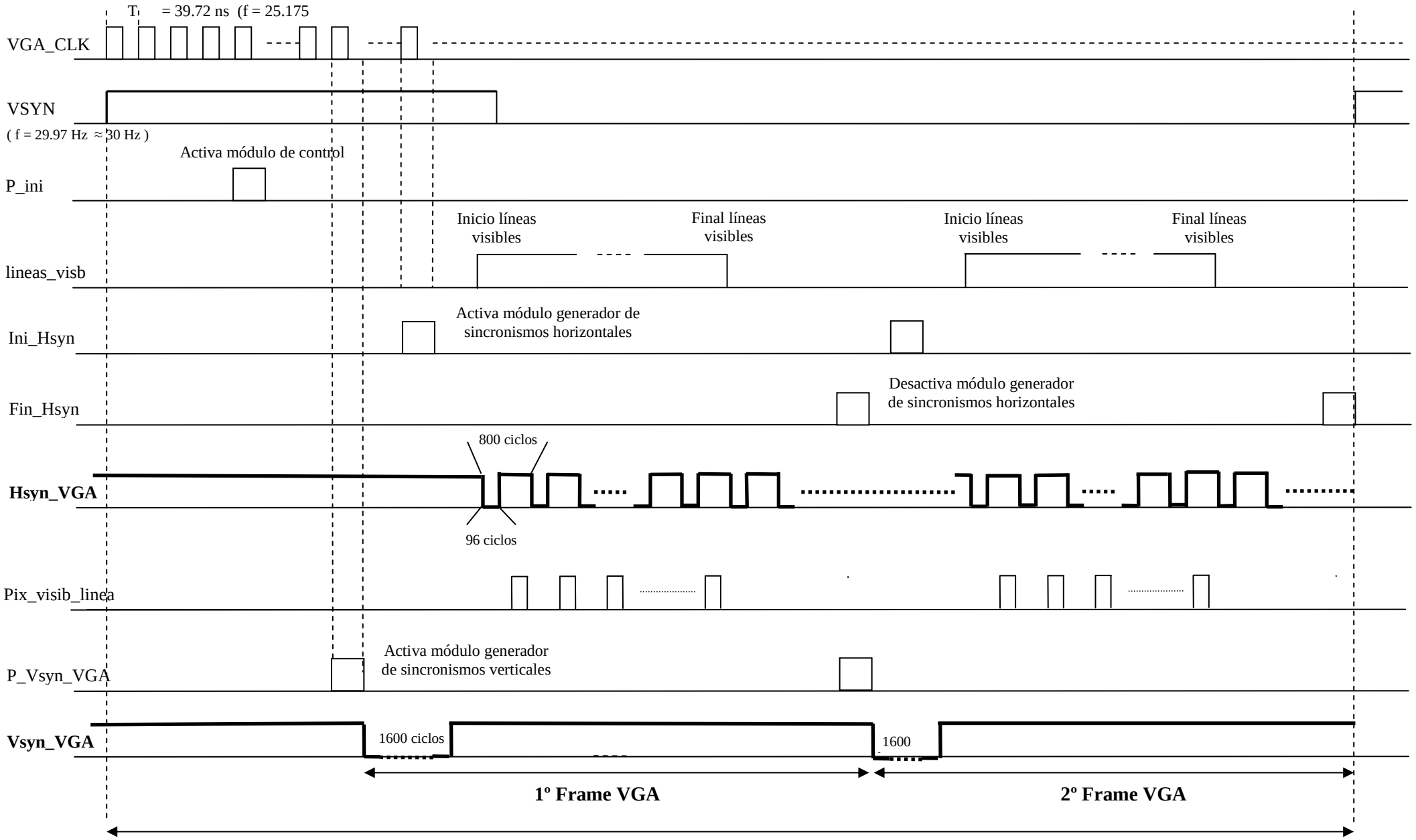
Dominio de reloj 1

Dominio de reloj 2

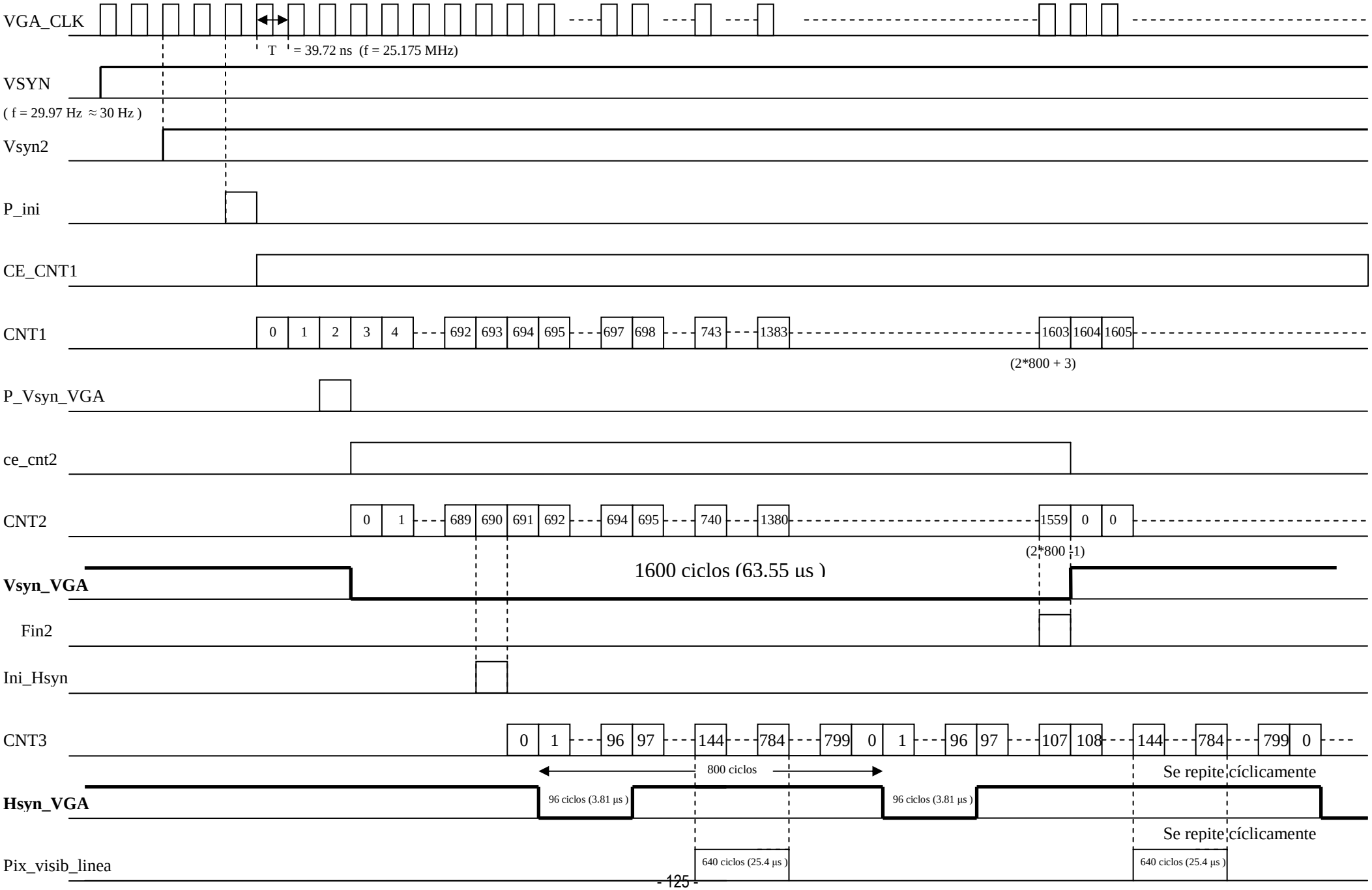
ANEXO C: Arquitectura Generador de Sincronismos VGA

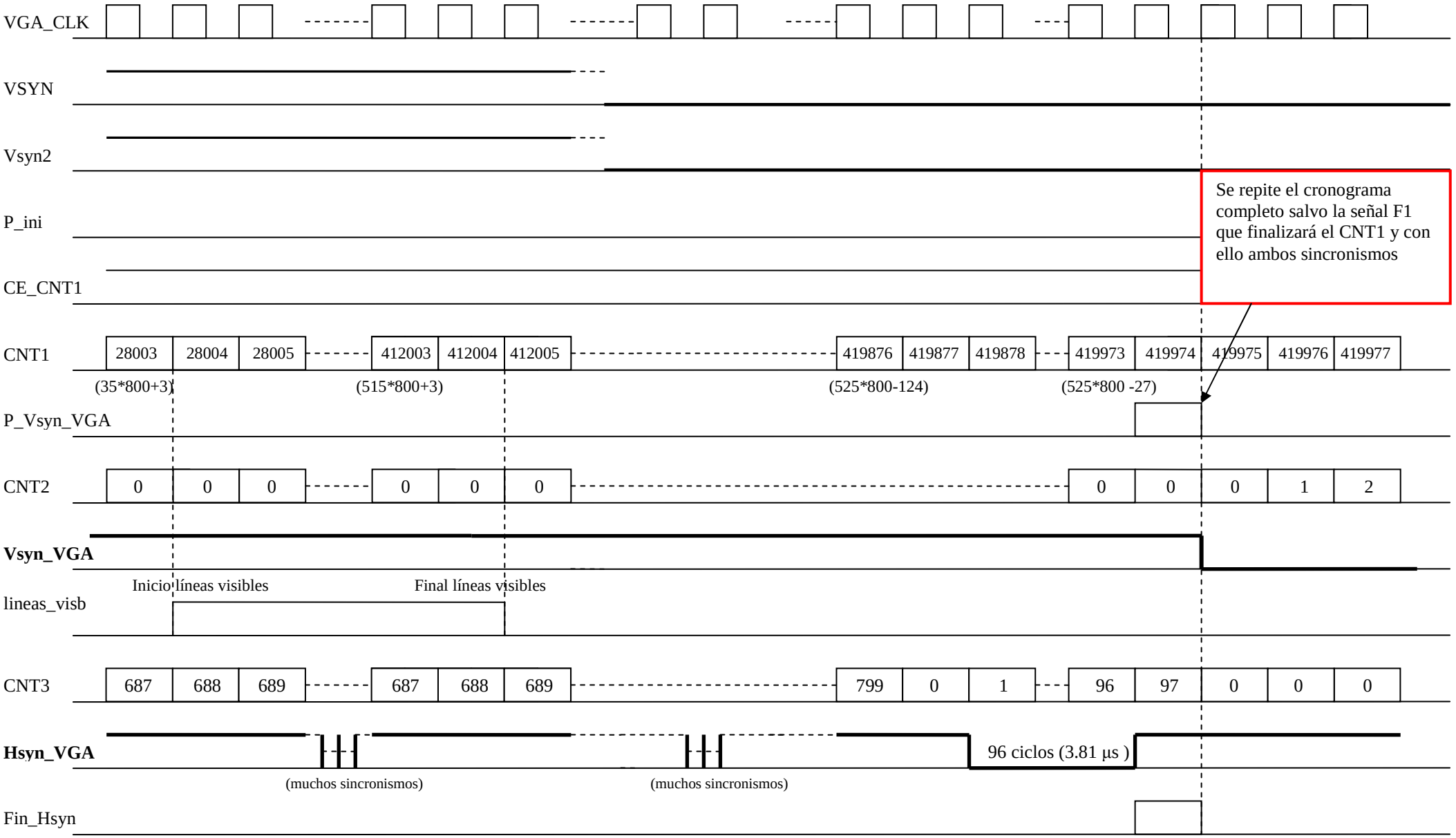


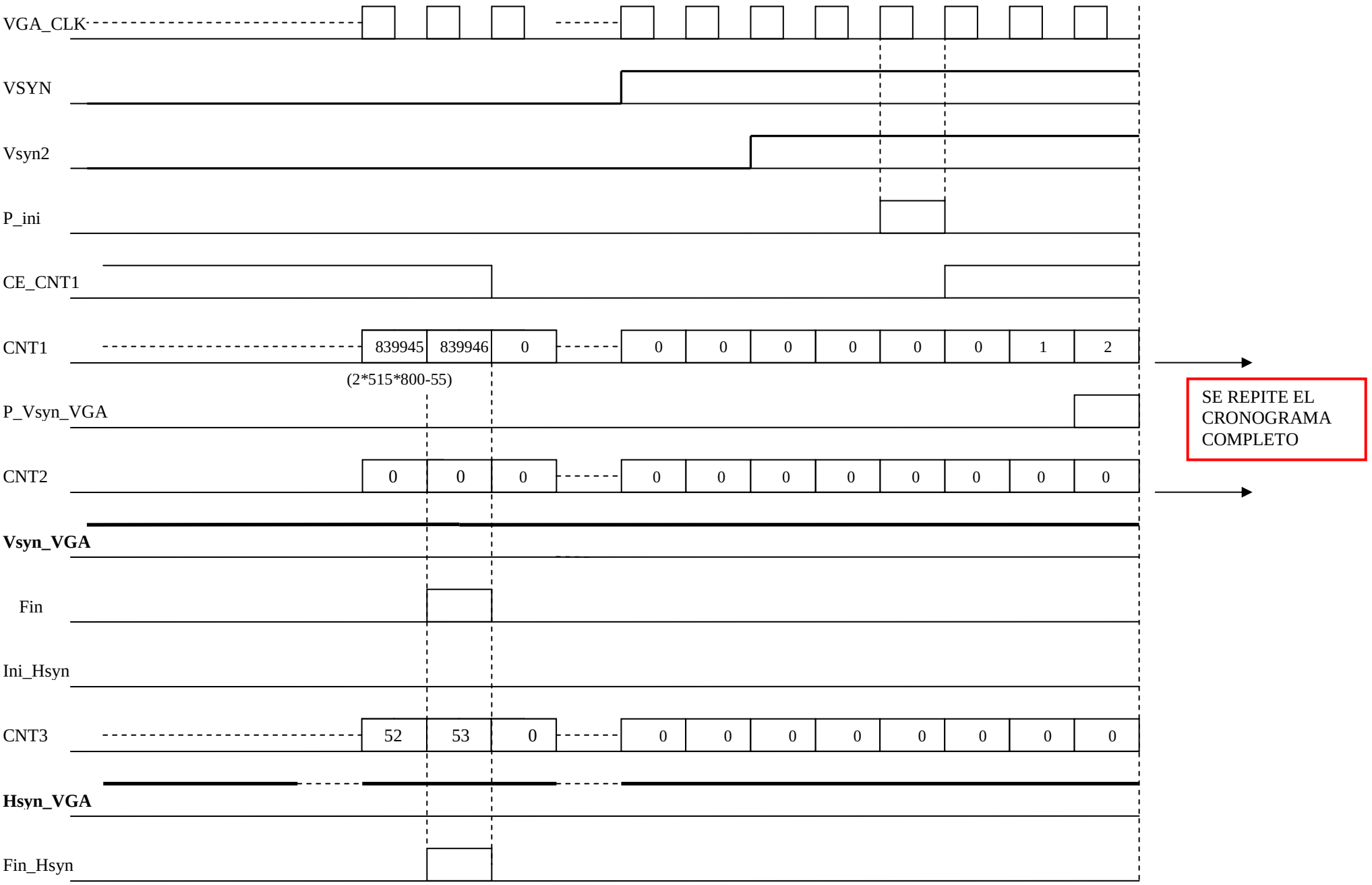
ANEXO D: Cronograma general Generador de sincronismos



ANEXO E: Cronograma completo Generador de sincronismos




VGA_CLK
VSYN
Vsyn2
P_ini
CE_CNT1
CNT1
28003
28004
28005
412003
412004
412005
419876
419877
419878
419973
419974
419975
419976
419977
(35*800+3)
(515*800+3)
(525*800-124)
(525*800 -27)
Se repite el cronograma completo salvo la señal F1 que finalizará el CNT1 y con ello ambos sincronismos
P_Vsyn_VGA
CNT2
0
0
0
0
0
0
0
0
0
1
2
Vsyn_VGA
Inicio líneas visibles
Final líneas visibles
lineas_visb
CNT3
687
688
689
687
688
689
799
0
1
96
97
0
0
0
Hsyn_VGA
96 ciclos (3.81 µs)
(muchos sincronismos)
(muchos sincronismos)
Fin_Hsyn


VGA_CLK
VSYN
Vsyn2
P_ini
CE_CNT1
CNT1
839945
839946
0
0
0
0
0
0
0
0
1
2
(2*515*800-55)
P_Vsyn_VGA
CNT2
0
0
0
0
0
0
0
0
0
0
0
0
Vsyn_VGA
Fin
Ini_Hsyn
CNT3
52
53
0
0
0
0
0
0
0
0
0
0
Hsyn_VGA
Fin_Hsyn
SE REPITE EL CRONOGRAMA COMPLETO