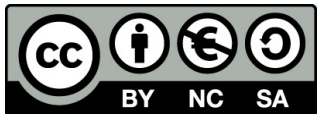
Laboratorio 11:

Diseño con IP cores

configuración de una cámara por bus SSBC y captura de video

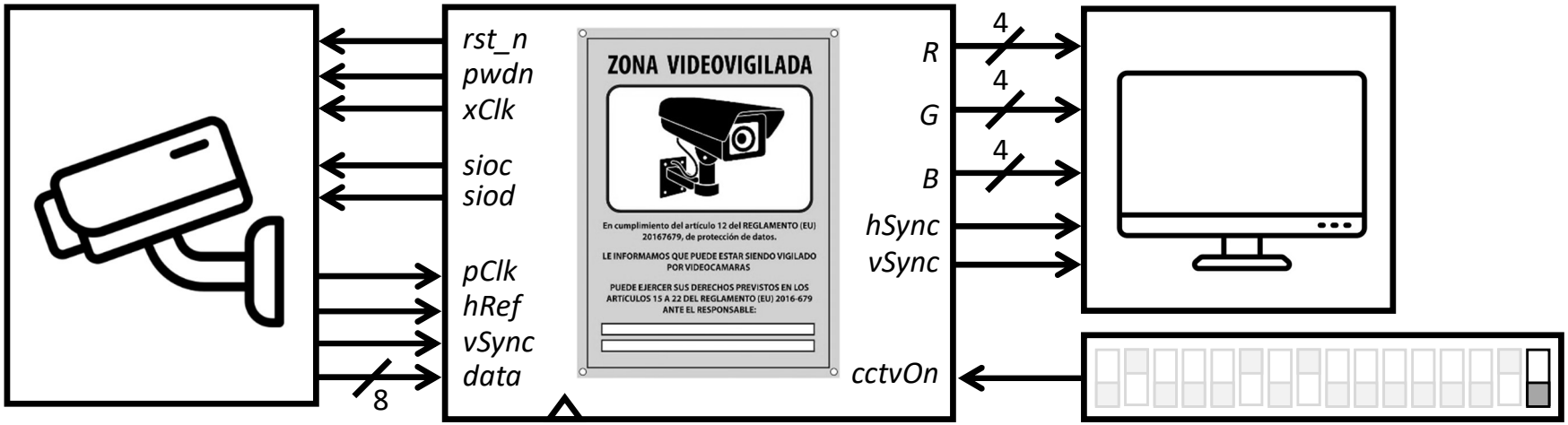
Diseño automático de sistemas

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid



Presentación

- Diseñar un circuito cerrado de televisión (CCTV)
  - Visualizará en tiempo real sobre el monitor las imágenes capturadas por la cámara.
  - La imagen se congelará activando el switch 0.

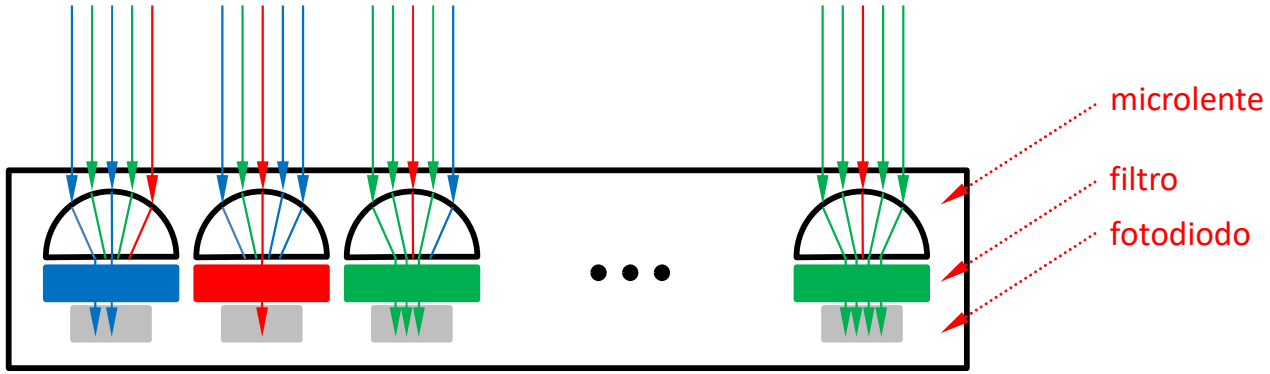


J.M. Mendías
2024

Captura digital de imágenes

- La luz visible es un haz de ondas electromagnéticas (o fotones) formada por longitudes de onda comprendidas entre 380 y 750 nm.
  - El brillo es la percepción humana de la intensidad absoluta de la luz.
  - El color es la percepción humana de la intensidad relativa de cada una de las longitudes de onda que forman la luz.

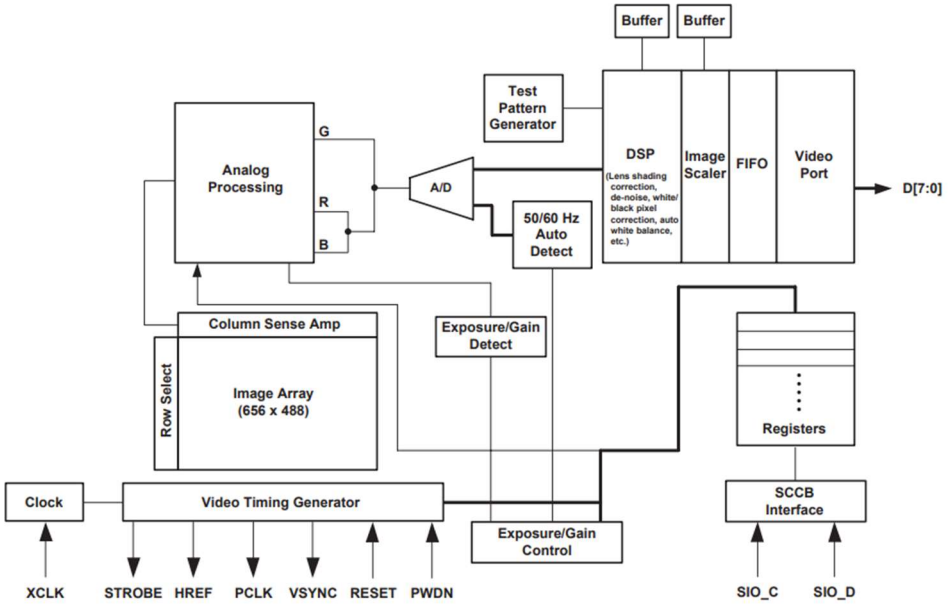
- Un sensor de imagen consta de una matriz de celdas formadas por:
  - Un fotodiodo CMOS capaz de detectar únicamente la intensidad de la luz incidente.
  - Una microlente que concentra la luz sobre el fotodiodo.
  - Un filtro R, G o B que evita que ciertas longitudes de onda alcancen el fotodiodo.
  - Celdas con distintos filtros se distribuyen alternadamente en la matriz.



J.M. Mendías 2024

Cámara digital OV7670

- La cámara digital OV7670 dispone de:
  - Un sensor de imagen de 0,3 Mpix.
  - Un ASP y un DSP que leen los sensores para calcular el color real de cada pixel.
  - Un puerto de vídeo síncrono de 8 bits que envía el color de cada pixel en serie:
    - Comenzando por la esquina superior izquierda de la imagen.
  - Un módulo programable por bus SCCB que fija las características vídeo transmitido:
    - Lo programaremos: Imágenes VGA (640×480 px) en formato RGB 4:4:4 (12b/px) y a 30 fps.

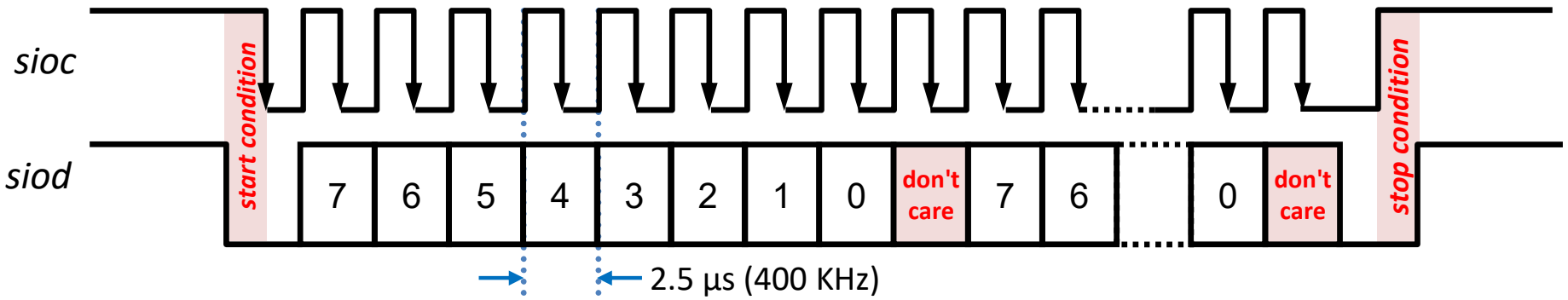


J.M. Mendías 2024

Cámara digital OV7670

programación (i)

- El protocolo SCCB (*Serial Camera Control Bus*) es bus bidireccional serie síncrono compatible con IIC (*Inter Integrated Circuits*).
  - En su variante de 2 hilos dispone de dos líneas:
    - SIOC: para transmitir el reloj de sincronización, por defecto en alta.
    - SIOD: para transmitir los datos serie, por defecto en alta.
  - La señal de reloj siempre debe ser generada por el host.
    - Puede tener una frecuencia de hasta 400 KHz.
  - Todos los datos transmitidos son de 8 bits (*MSB first*) seguidos de 1 *don't care* bit.
    - Se muestrean a flancos de bajada de CLK.
    - Si se transmiten varios datos, lo hacen en tramas.
    - El comienzo de la trama se señaliza con una *start condition* (transición 1-0 en SIOD).
    - El final de la trama se señaliza con una *stop condition* (transición 0-1 en SIOD).

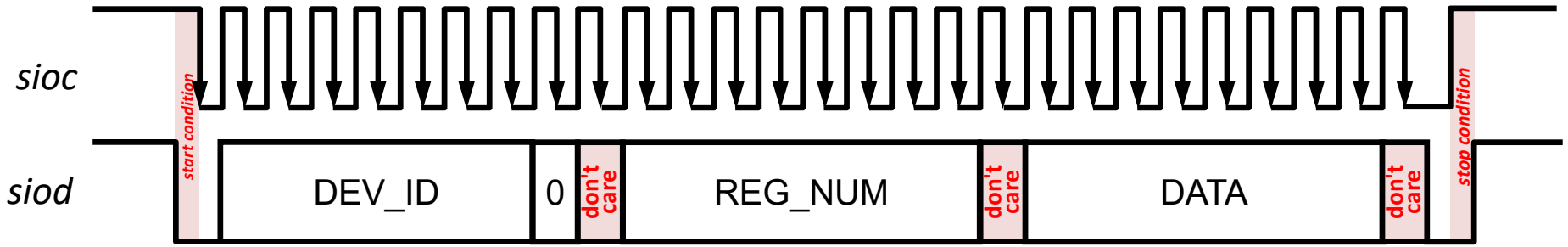


J.M. Mendías
2024

Cámara digital OV7670

programación (ii)

- El host configura la cámara enviando tramas de 3 datos por el bus SCCB:
  - Inicia la transmisión generando la *start condition* (transición 1-0 en SIOD).
  - Envía la dirección del dispositivo (7b) y el tipo de operación (1b)
    - La cámara OV7670 tiene como identificador: "0100001"
    - Las operaciones siempre serán de escritura, que se codifica con 0.
  - Envía el número de registro de la cámara (8b) y el valor a escribir en él (8b).
    - Existen un total de 43 registros de 8 bits cada uno.
  - Finaliza la transmisión generando la *stop condition* (transición 0-1 en SIOD).



- En este laboratorio, para configurar la cámara, se enviarán 35 tramas.

J.M. Mendías
2024

Cámara digital OV7670

transmisión de vídeo (i)

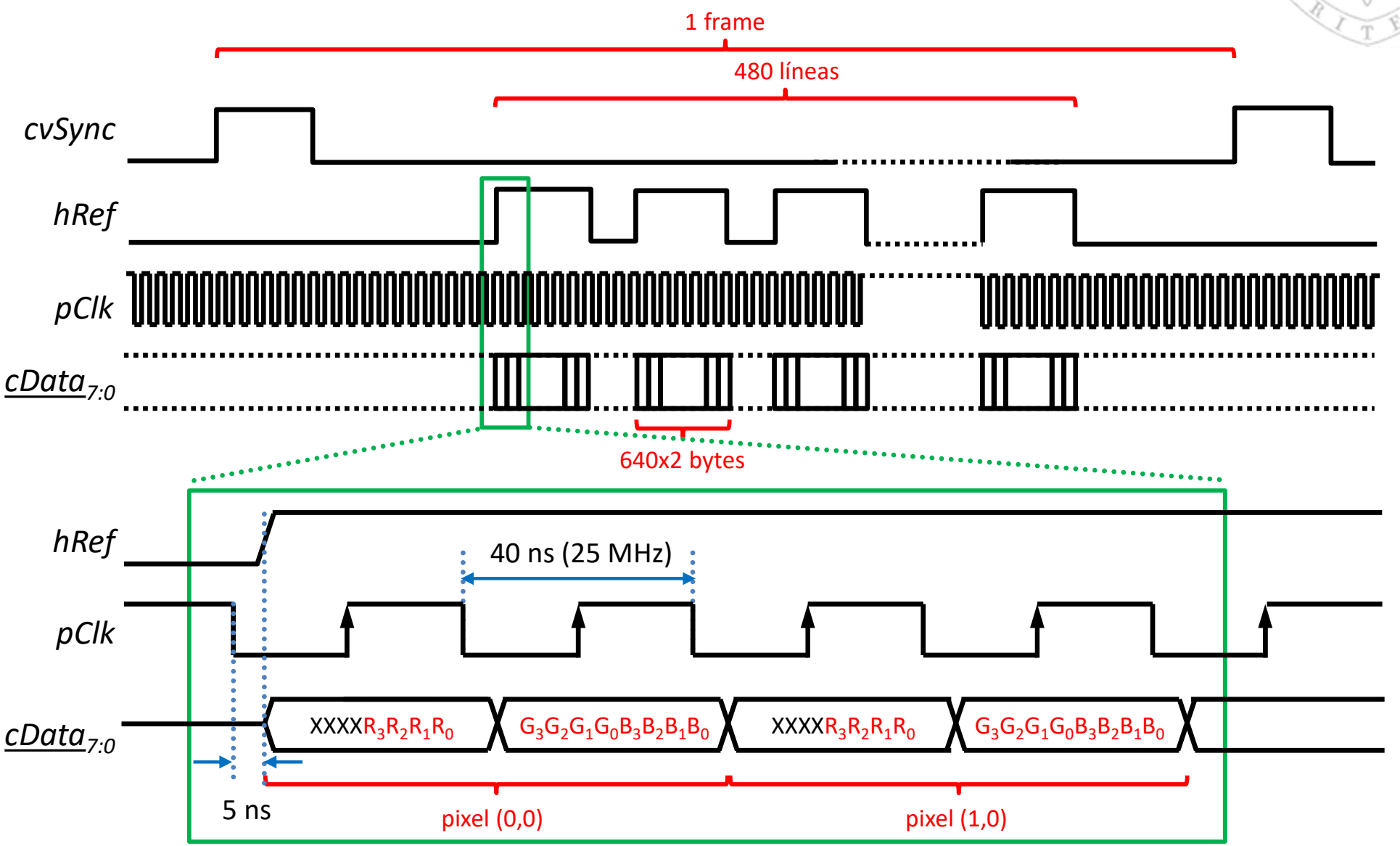
- La transmisión de vídeo se realiza indefinidamente a través de un bus de 8 bits síncrono de 4 líneas:
  - cvSync: señal que señaliza con un pulso el comienzo de cada imagen (frame).
  - hRef: señal de *strobe* que señaliza en ALTA cuando hay datos válidos.
  - cData: señal de 8 bits por donde transmiten las componentes RGB de cada píxel.
  - pClk: señal de reloj a cuyo flanco de bajada el host debe muestrear cData.

- Adicionalmente la cámara digital tiene 3 entradas:
  - rst_n: reset hardware, que fijaremos a 1.
  - pwdn: apaga la cámara, que fijaremos a 0.
  - xClk: señal base de reloj usada por la cámara para generar pClK, será de 25 MHz.

J.M. Mendías
2024

Cámara digital OV7670

transmisión de vídeo (ii)



J.M. Mendías
2024

Diseño principal
captura de vídeo (i)

- Los sistemas de procesamiento de vídeo almacenan temporalmente *frames* completos en *buffers*.
  - Así, las distintas fases del procesamiento pueden desacoplarse.
  - Un *buffer* de vídeo es una memoria en la se almacenan secuencialmente los píxeles en orden de captura:
    - Su profundidad depende del tamaño de la imagen.
    - Su anchura depende de la profundidad de color de la imagen.

- La cámara envía vídeo VGA en color, sin embargo, es posible:
  - Reducir el tamaño de la imagen mediante submuestreo (*undersampling*):
    - QVGA: se toman únicamente pixeles en posiciones (x,y) múltiplos de 2 (pares).
    - QQVGA: se toman únicamente pixeles en posiciones (x,y) múltiplos de 4.
  - Reducir la profundidad de color transformando el color a escala de grises.

Cámara digital OV7670

transmisión de vídeo (iii)

- Cada alternativa requiere un buffer de vídeo de distinto tamaño:
  - La FPGA de la placa dispone de 1800 Kb de memoria Block RAM.
  - Organizadas en 50 slices RAMB36E1 (36 Kb) cada uno divisible en 2 RAMB18E1 (18 Kb).

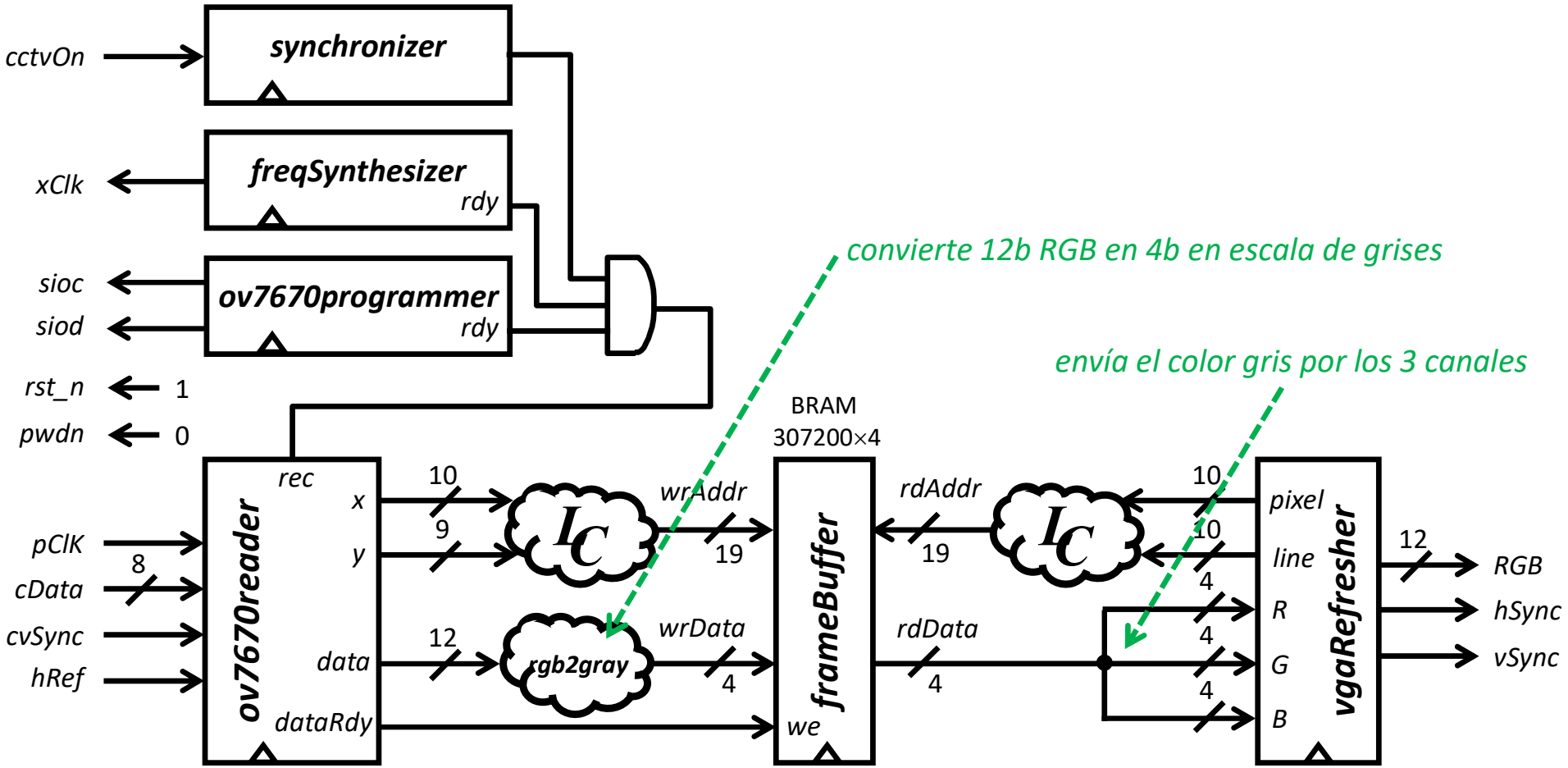
tamaño	# pix	color 12b/px	# RAMB18E1 requeridas	# RAMB36E1 requeridas	gris 4b/px	# RAMB18E1 requeridas	# RAMB36E1 requeridas
VGA 640×480	307200	3600 Kb	1	103	1200 Kb	3	36
QVGA 320×240	76800	900 Kb	1	26	300 Kb	3	8
QQVA 160×120	19200	225 Kb	5	5	75 Kb	5	0

- La dirección del buffer que ocupa un pixel en posición (x,y) se calcula:

tamaño	# pix	# bits x	# bits y	# bits addr	addr
VGA 640×480	307200	10	9	19	640·y + x
QVGA 320×240	76800	9	8	17	(640/2)·(y/2) + (x/2)
QQVA 160×120	19200	8	7	15	(640/4)·(y/4) + (x/4)

Diseño principal

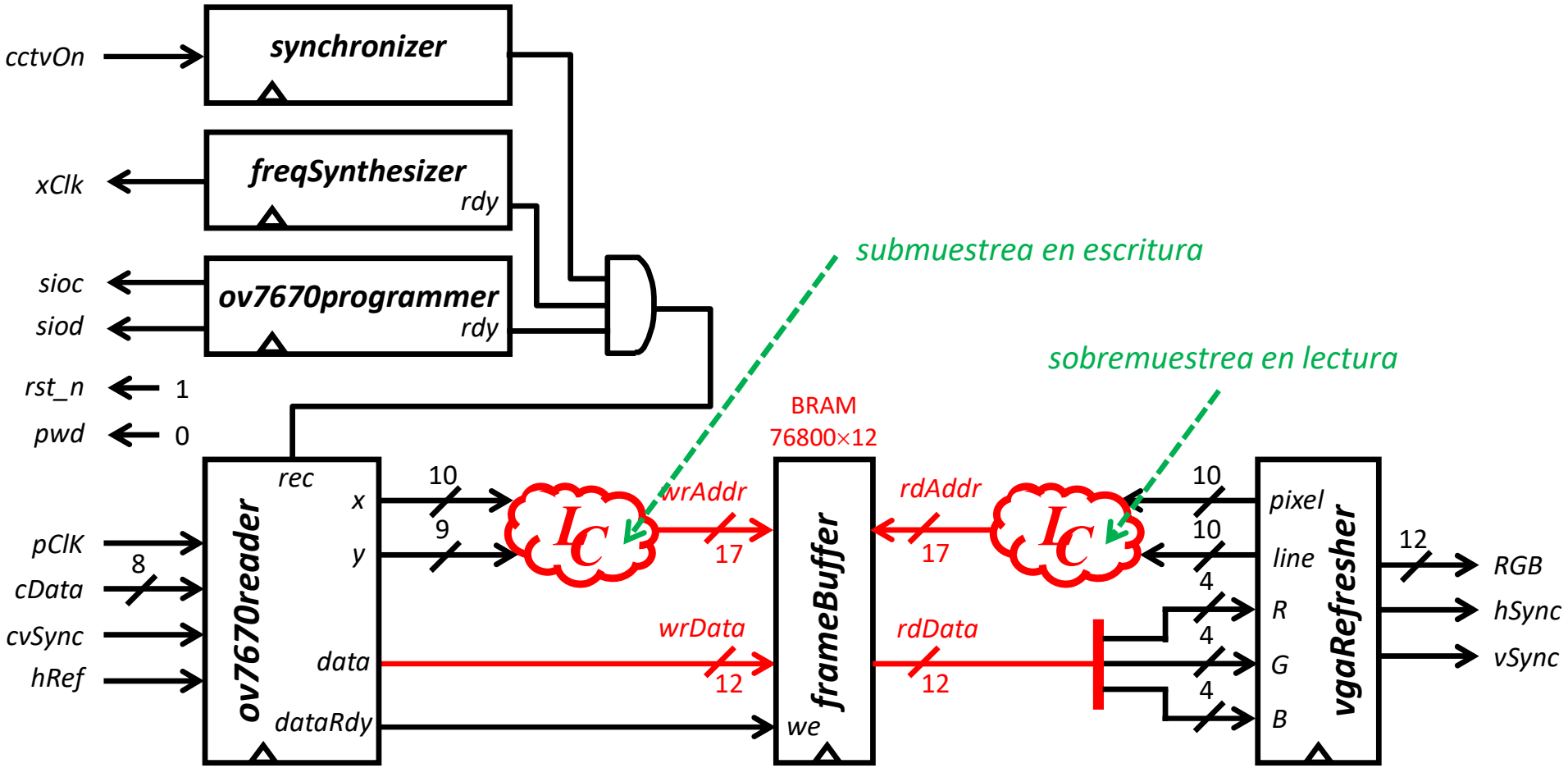
VGAgrey: esquema RTL



J.M. Mendías
2024

Diseño principal

QVGAcolor: esquema RTL



J.M. Mendías
2024

Programador de cámara

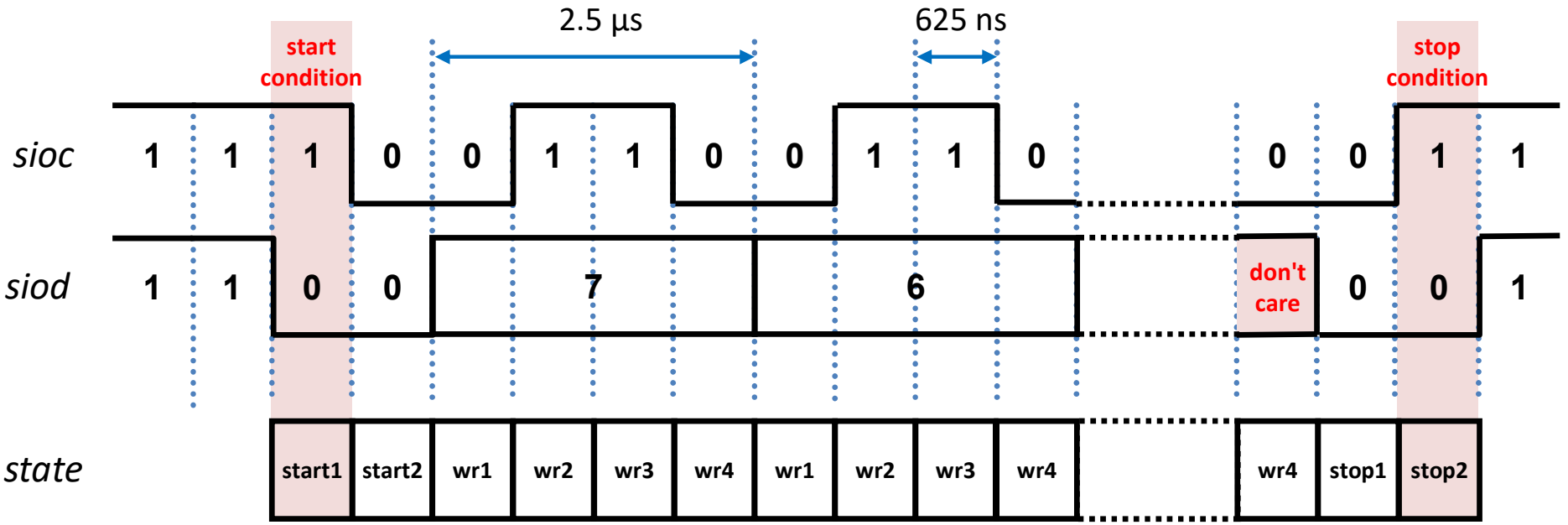
presentación

- Diseñar un programador de cámara elemental:
  - Configura la cámara una única vez 2 ms después del inicio del sistema.
  - Enviando 35 comandos con una separación de 1 ms.
  - Cada comando es una trama de 27 bits (3×9 bits) que consta de:
    - Un identificador de dispositivo.
    - Un identificador de registro.
    - Un valor a cargar en el registro.
  - Cuando finaliza el envío activará permanentemente la salida rdy.

Programador de cámara

FSMD temporizada

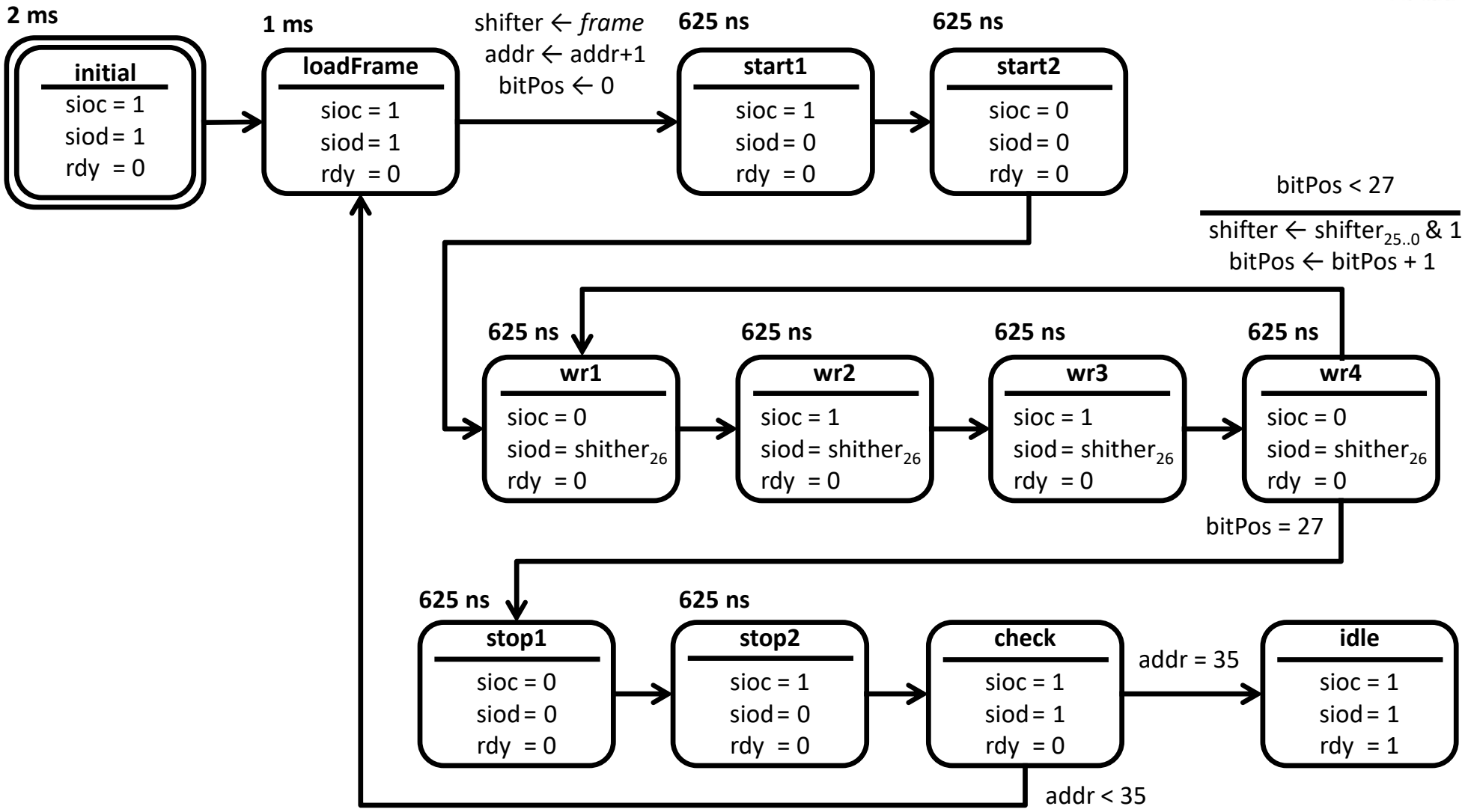
- El protocolo SCCB (y su homólogo IIC) es extremadamente sensible a la correcta secuenciación de los flancos en sioc y siod.
  - En especial durante la generación de la start/stop conditions.
- Para asegurar una temporización adecuada en el envío de tramas:
  - Dividiremos el tiempo de transmisión de cada bit en 4 fases.
  - Una FSMDT generará explícitamente las formas de onda.



J.M. Mendías
2024

Programador de cámara

FSMD temporizada



J.M. Mendías 2024

Programador de cámara

ov7670programmer.vhd

```
architecture syn of ov7670programmer is

  type t_conf is record
    reg  : std_logic_vector(7 downto 0);
    data : std_logic_vector(7 downto 0);
  end record t_conf;

  constant PROG_LEN : natural := 35;

  type romType is array (0 to PROG_LEN-1) of t_conf;

  signal confROM : romType := (
    ( X"12", X"80" ),  -- COM7   Reset
    ( X"12", X"04" ),  -- COM7   Salida RGB, VGA
    ( X"40", X"F0" ),  -- COM15  habilita rango completo de salida, RGB444
    ( X"8C", X"02" ),  -- RGB444 habilita RGB444, formato xRGB
    ...
  );

  -- Tiempos de operación (en ciclos de reloj) de la camara
  constant POWERUP_CYCLES : natural := ms2cycles(FREQ_KHZ, 2); -- tiempo de arranque
  constant SWRST_CYCLES   : natural := ms2cycles(FREQ_KHZ, 1); -- tiempo de espera entre tramas

  constant SCK_CYCLES     : natural := (FREQ_KHZ*1000)/BAUDRATE;  -- tiempo de transmisión de 1b
  constant QSCK_CYCLES    : natural := SCK_CYCLES/4;              -- dividido entre 4

  constant FRAME_LEN : natural := 3*9;

begin
  ...
end syn;
```

el uso de una estructura mejora la legibilidad del programa

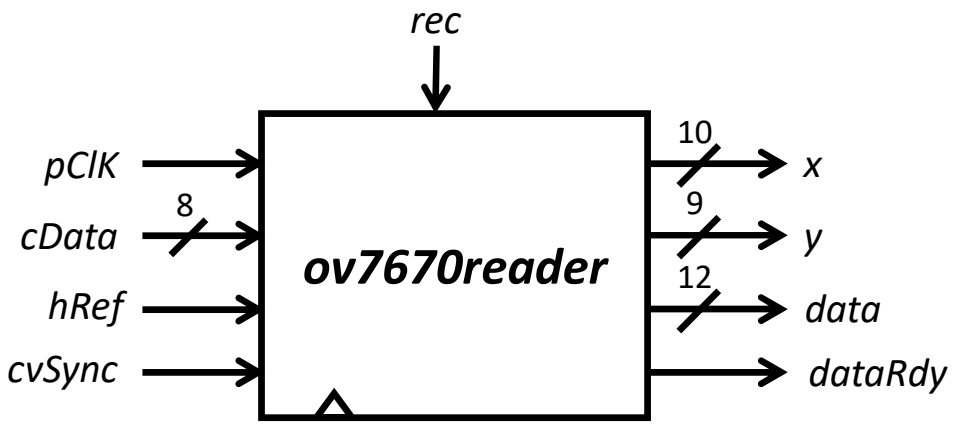
todos los tiempos parametrizados

J.M. Mendías 2024

Lector de vídeo

presentación

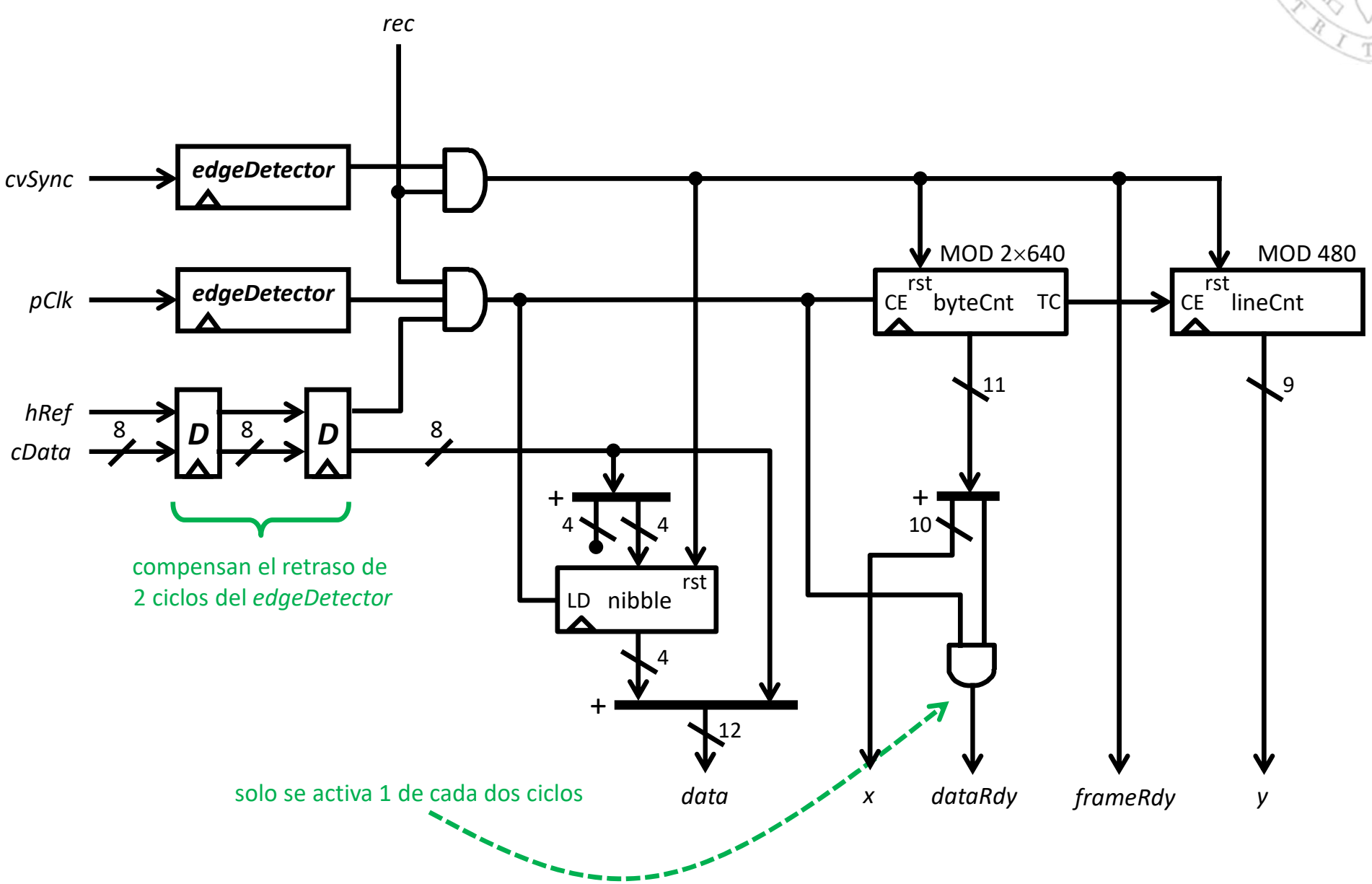
- Diseñar un lector de vídeo elemental:
  - Captura imágenes en color (12b/px) de tamaño VGA (640×480)
  - Lee la imagen enviada pixel a pixel solo si rec está activada.
    - La cámara envía cada píxel en 2 bytes consecutivos.
  - Para cada pixel recibido indica la posición (x,y) que ocupa y su color RGB.
  - Indica la recepción de un nuevo pixel activando durante un ciclo la señal dataRdy.



J.M. Mendías
2024

Lector de vídeo

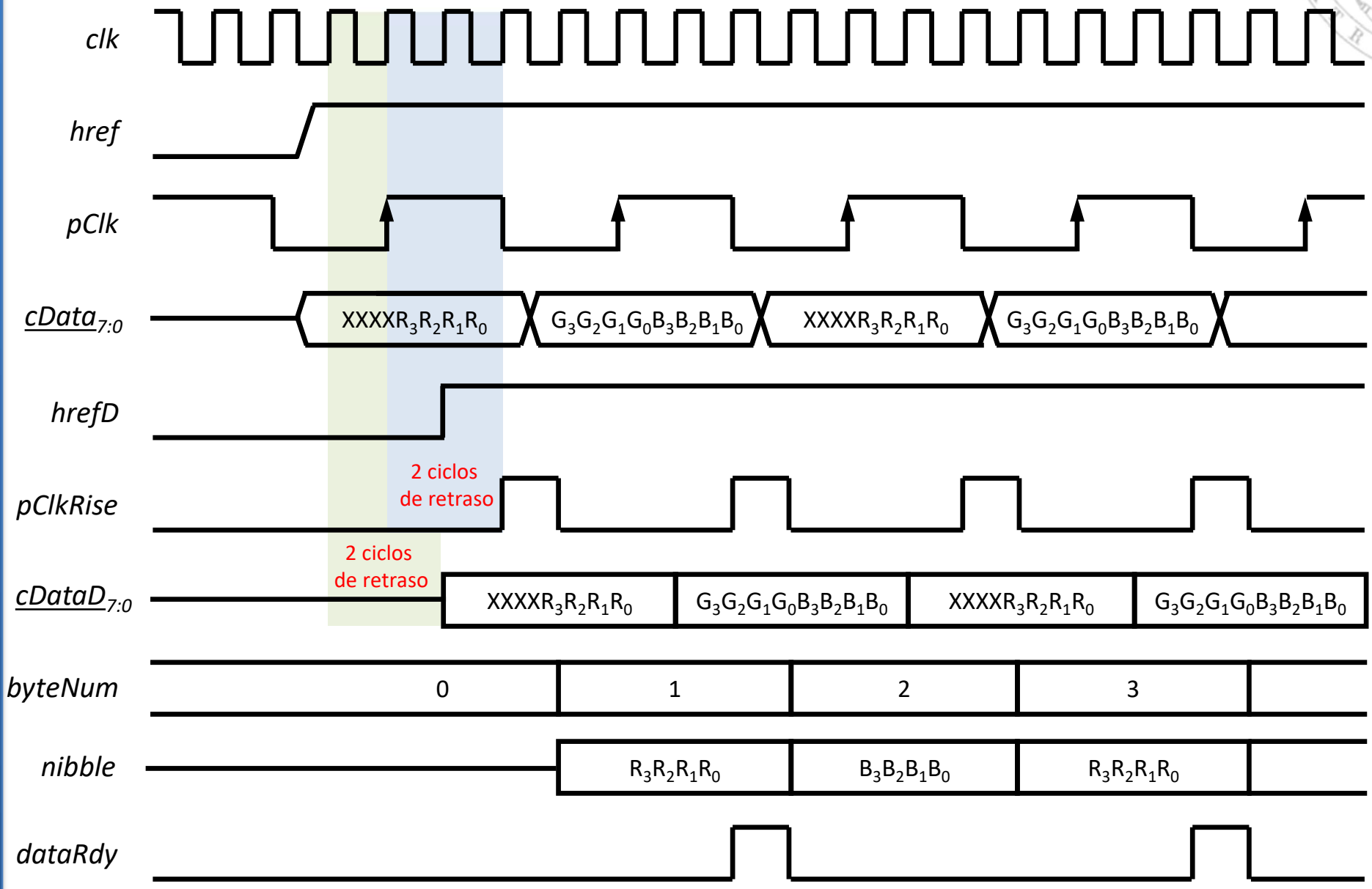
esquema RTL



J.M. Mendías
2024

Lector de vídeo

esquema RTL: análisis del comportamiento

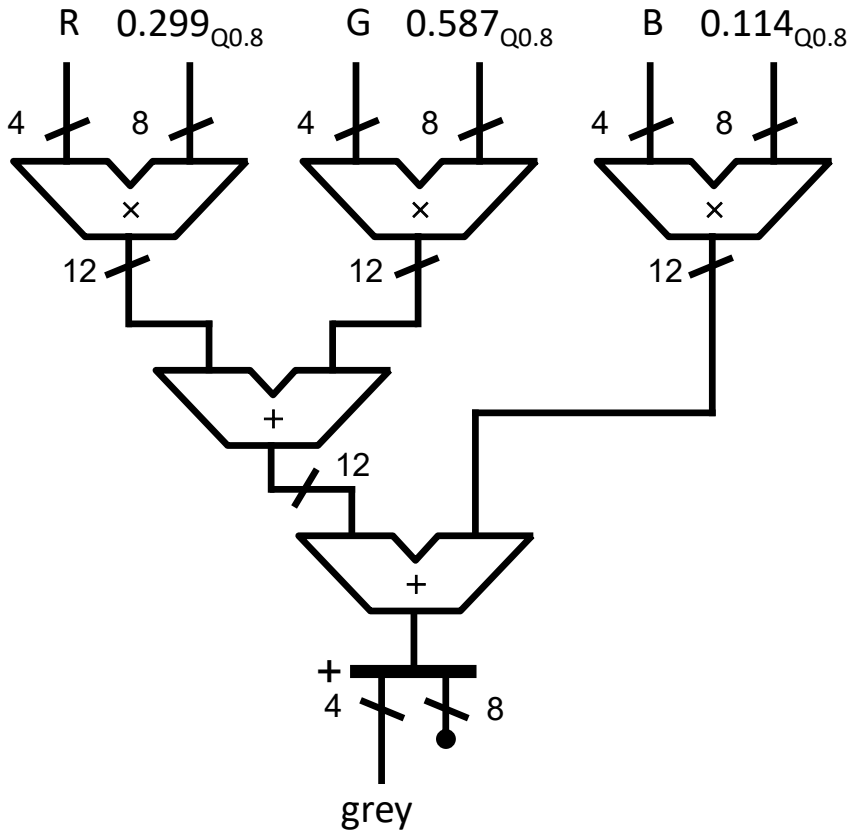


J.M. Mendías 2024

Conversor a escala de grises

esquema RTL

- Para convertir el color de un pixel a escala de grises se calcula la suma ponderada de cada canal RGB:
  - Existen distintas ponderaciones, las más común es según el estándar NTSC.
    - Tratan de reflejar la sensibilidad relativa del ojo humano a cada color.



NTCS / Rec. ITU-R BT.601:

$$grey = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Rec. ITU-R BT.709:

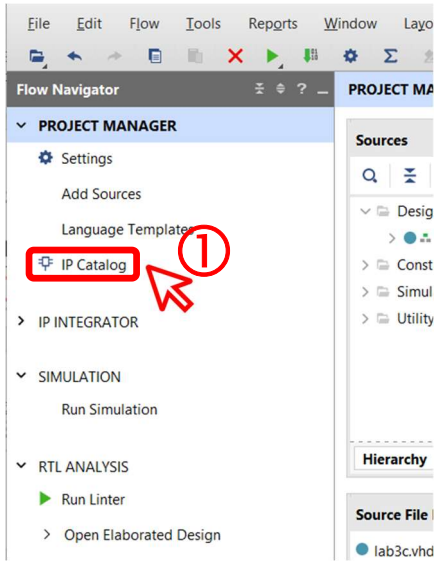
$$grey = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$$

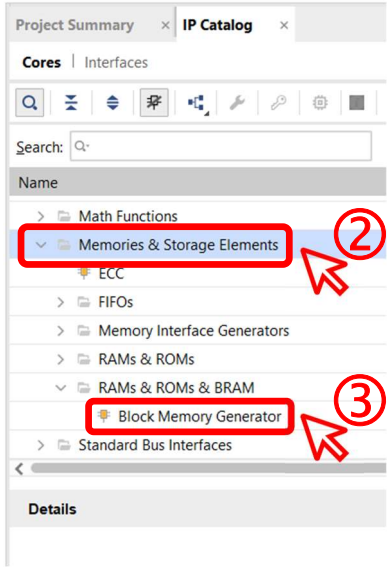
Promedio:

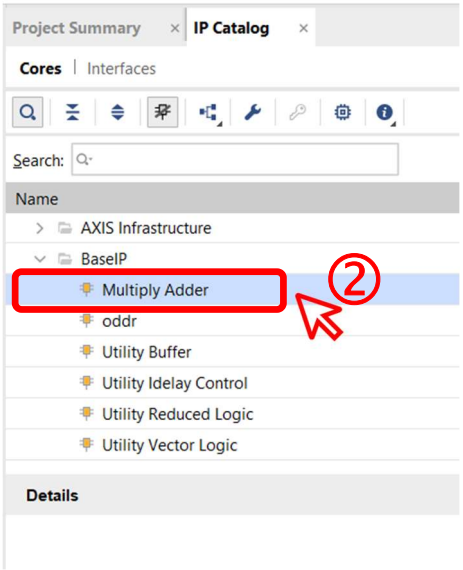
$$grey = 0.333 \cdot R + 0.333 \cdot G + 0.333 \cdot B$$

Diseño con IP-cores

- Para un uso óptimo de la BRAM usaremos un IP generado por el Block Memory Generator
  - Por ejemplo, en VGAgray, una RAM de 2^{19}x4 = 2048 Kb, pero solo necesito 1200 Kb.
- Para calcular addr en función de (x,y) usaremos IPs Multiply Adder







- Estos IPs se:
  - Configuran e incorporan al diseño usando un Wizard.
  - Instancian en VHDL como compontes.

J.M. Mendías
2024

Diseño con IP-cores

VGAgrey: Block Memory Generator (i)

Nombre: frameBuffer

Tipo: Simple dual port RAM

Interfaz: Nativo

Reloj: Común

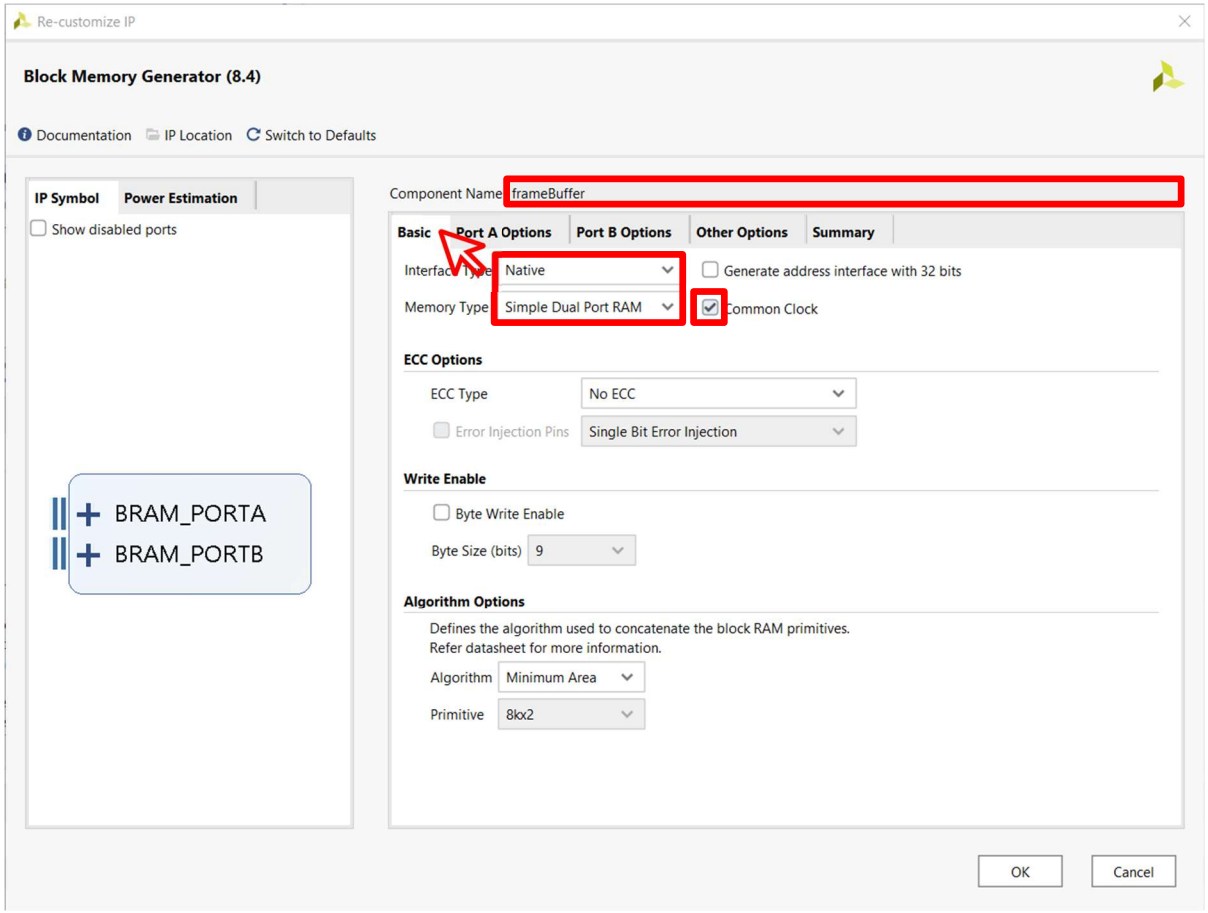
Puerto A: Escritura (4 bits)

Puerto B: Lectura (4 bits)

Profundidad: 307200 palabras

Sin puertos de capacitación

Sin registro de salida



J.M. Mendías
2024

Diseño con IP-cores

VGAgrey: Block Memory Generator (ii)

Nombre: frameBuffer

Tipo: Simple dual port RAM

Interfaz: Nativo

Reloj: Común

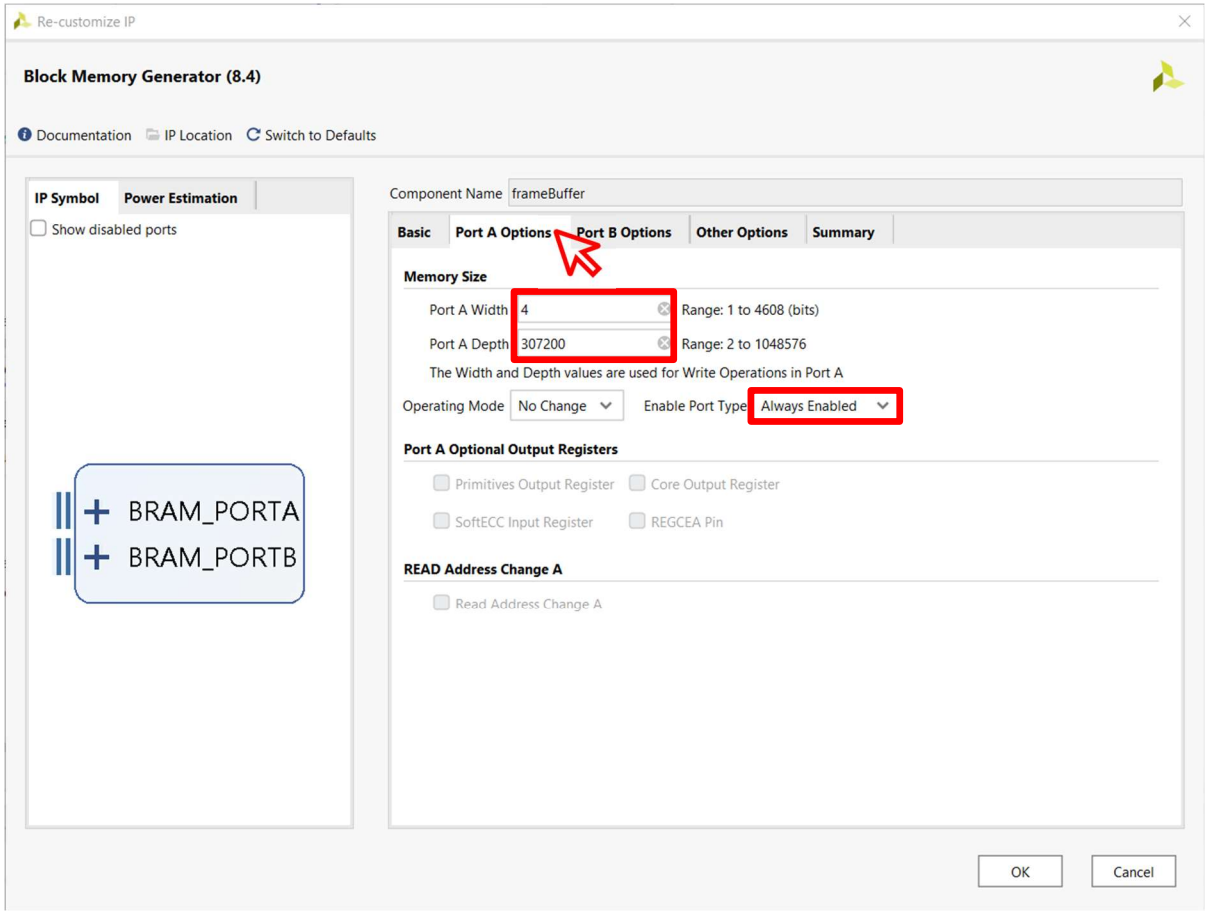
Puerto A: Escritura (4 bits)

Puerto B: Lectura (4 bits)

Profundidad: 307200 palabras

Sin puertos de capacitación

Sin registro de salida



J.M. Mendías
2024

Diseño con IP-cores

VGAgrey: Block Memory Generator (iii)

Nombre: frameBuffer

Tipo: Simple dual port RAM

Interfaz: Nativo

Reloj: Común

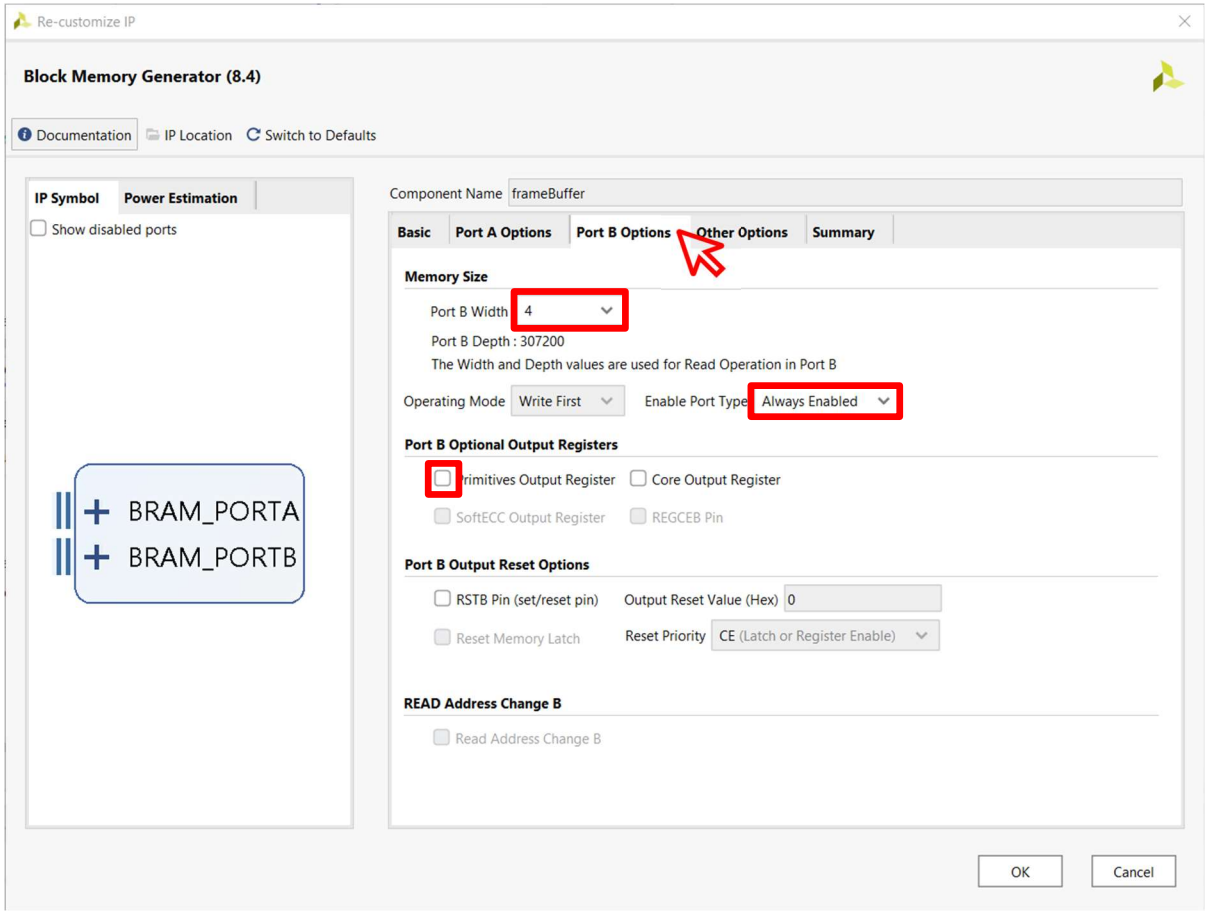
Puerto A: Escritura (4 bits)

Puerto B: Lectura (4 bits)

Profundidad: 307200 palabras

Sin puertos de capacitación

Sin registro de salida



J.M. Mendías
2024

Diseño con IP-cores

VGAgrey: Block Memory Generator (iv)

Nombre: frameBuffer

Tipo: Simple dual port RAM

Interfaz: Nativo

Reloj: Común

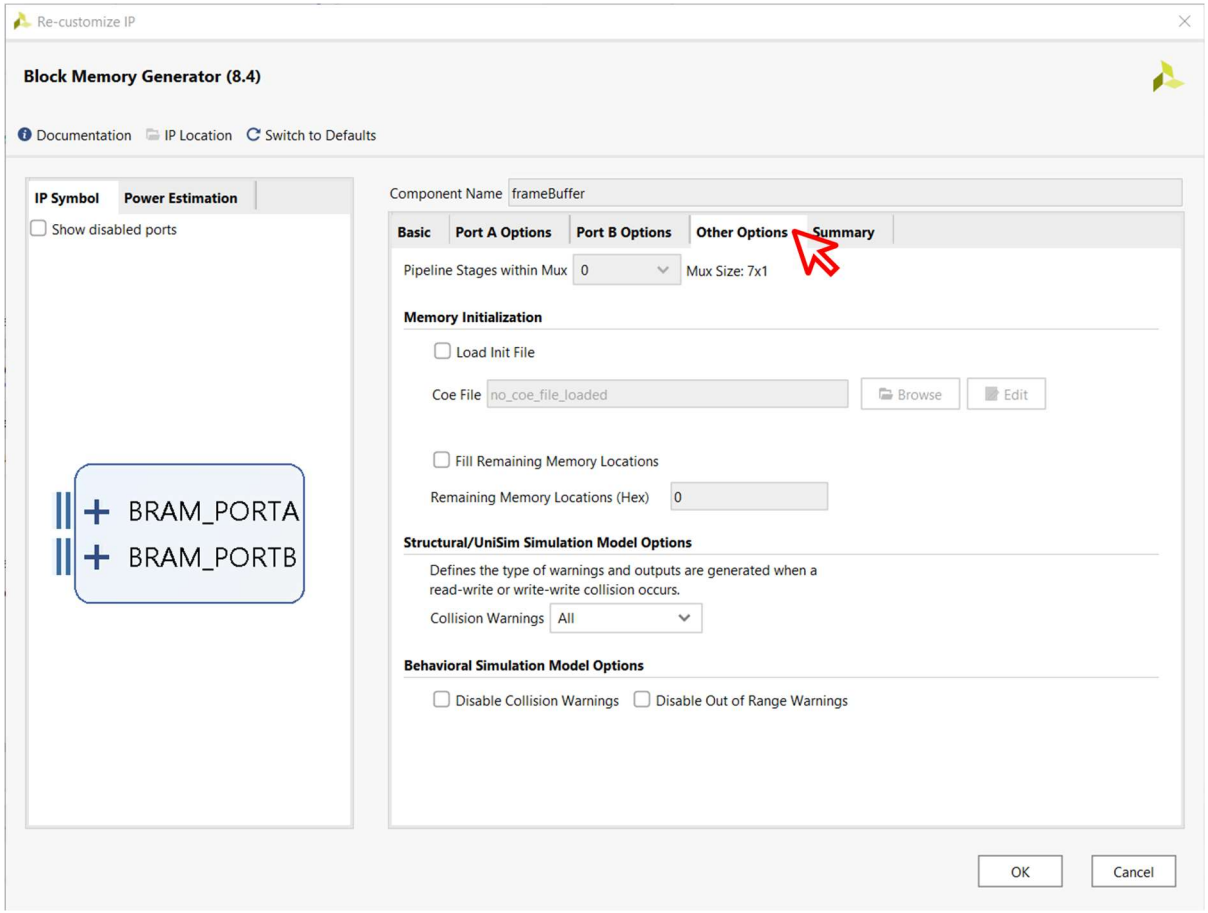
Puerto A: Escritura (4 bits)

Puerto B: Lectura (4 bits)

Profundidad: 307200 palabras

Sin puertos de capacitación

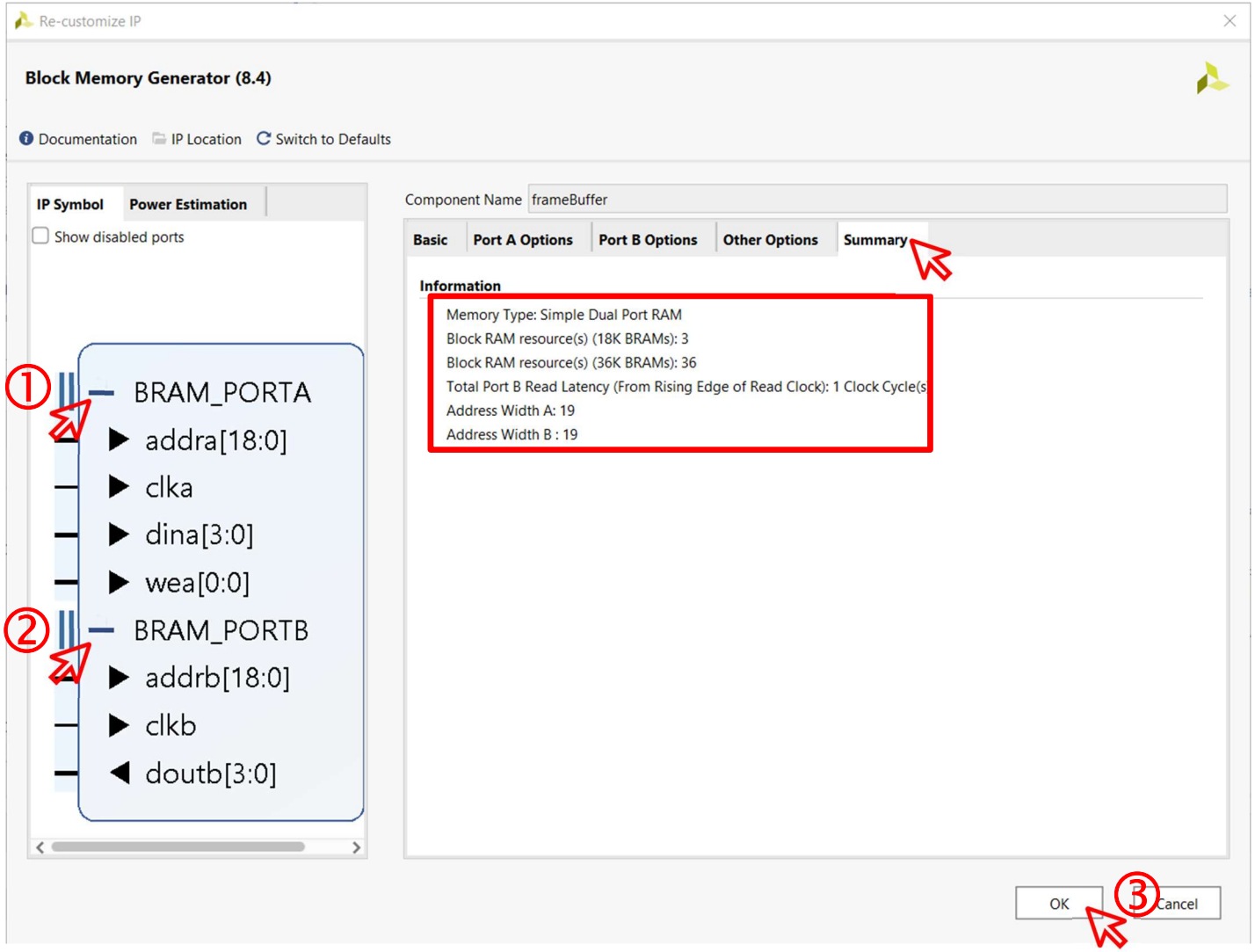
Sin registro de salida



J.M. Mendías
2024

Diseño con IP-cores

VGAgrey: Block Memory Generator (v)



J.M. Mendías 2024

Diseño con IP-cores

VGAgrey: Multiply Adder

$$addr = y \cdot 640 + x$$

tamaño	# bits x	# bits y	# pix	# bits addr
VGA 640×480	10	9	307200	19

Nombre: multAdd

Función: A×B+C

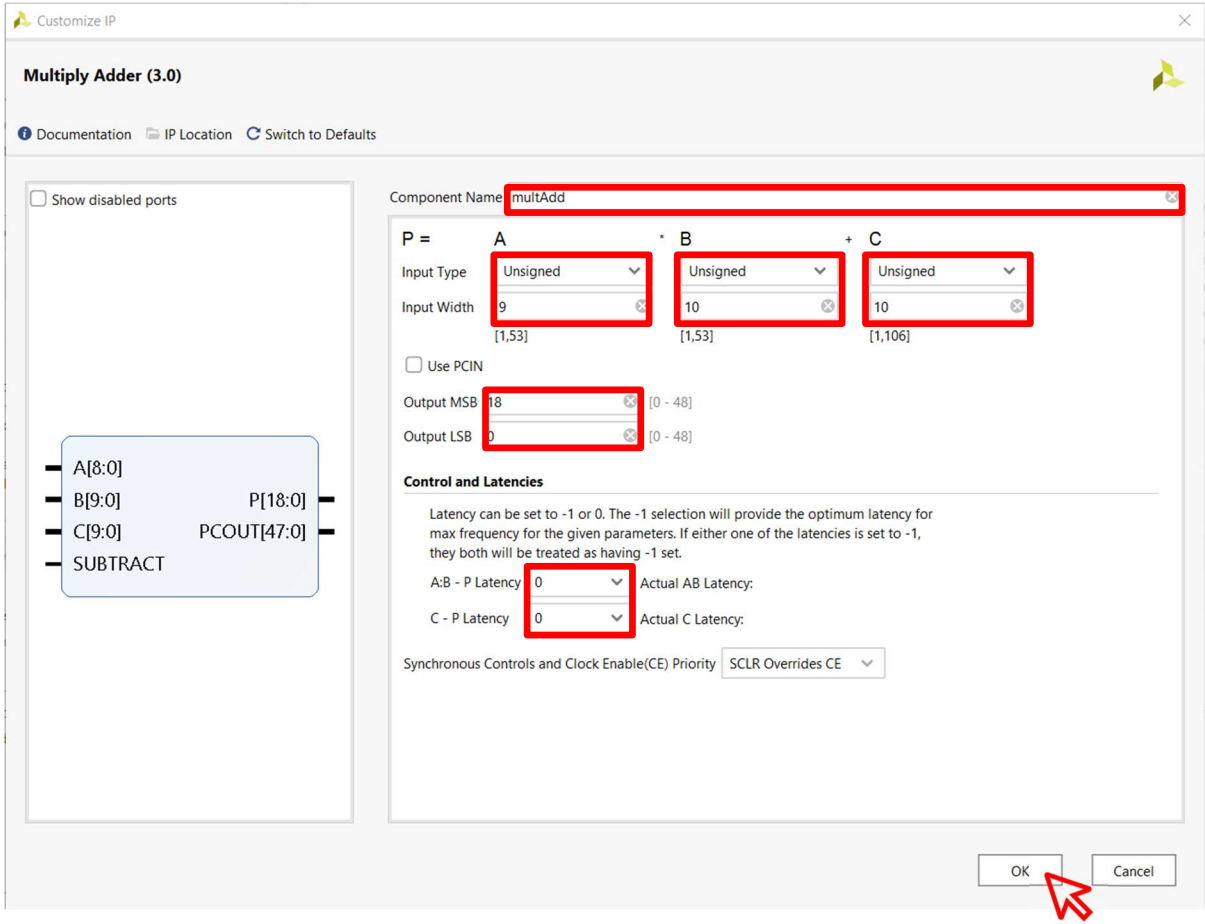
Tipo: Combinacional

Input A: y (9b sin signo)

Input B: 640 (10b sin signo)

Input C: x (10b sin signo)

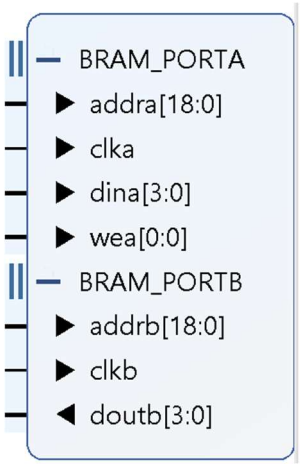
Output P: addr (19 b)

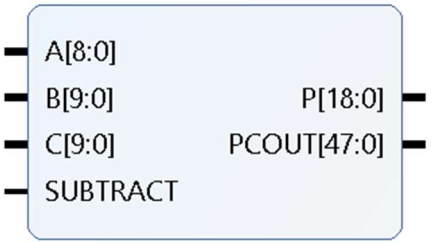


Diseño con IP-cores

VGAgrey: instanciación

```
architecture syn of lab11VGAgray is
  component frameBuffer
    port (
      clka  : in  std_logic;
      wea   : in  std_logic_vector(0 downto 0);
      addra : in  std_logic_vector(18 downto 0);
      dina  : in  std_logic_vector(3 downto 0);
      clkb  : in  std_logic;
      addrb : in  std_logic_vector(18 downto 0);
      doutb : out std_logic_vector(3 downto 0)
    );
  end component;
  component multAdd
    port (
      a         : in  std_logic_vector(8 downto 0);
      b         : in  std_logic_vector(9 downto 0);
      c         : in  std_logic_vector(9 downto 0);
      subtract  : in  std_logic;
      p         : out std_logic_vector(18 downto 0);
      pcout     : out std_logic_vector(47 downto 0)
    );
  end component;
  ...
begin
  wrAddrCalculator: multAdd
    port map ( a => wrY, b => std_logic_vector(to_unsigned(640,10)),
               c => wrX, subtract => '0', p => wrAddr, pcout => open );
  ...
end syn;
```





J.M. Mendías 2024

Diseño con IP-cores

VGAcolor

tamaño	# bits x	# bits y	# pix	# bits addr
QVGA 320×240	9	8	76800	17

$$addr = (y/2) \cdot (640/2) + (x/2)$$

Nombre: frameBuffer
Tipo: Simple dual port RAM
Interfaz: Nativo
Reloj: Común
Puerto A: Escritura (12 bits)
Puerto B: Lectura (12 bits)
Profundidad: 76800 palabras
Sin puertos de capacitación
Sin registro de salida

Nombre: multAdd
Función: A×B+C
Tipo: Combinacional
Input A: y (8b sin signo)
Input B: 320 (9b sin signo)
Input C: x (9b sin signo)
Output P: addr (17 b)

Tareas

1. Crear el proyecto **lab11VGAgrey** en el directorio **DAS**
2. Descargar de la Web los ficheros en:
   - **common:** **ov7670programmer.vhd**, **ov7670reader.vhd** y **rgb2grey.vhd**
   - **lab11VGAgrey:** **lab11VGAgrey.vhd** y **lab11VGAgrey.xdc**
3. Completar **common.vhd** con la declaración de los nuevos componentes.
4. Completar el código omitido en los ficheros:
   - **ov7670programmer.vhd** y **ov7670reader.vhd**
5. Añadir al proyecto los ficheros:
   - **common.vhd**, **synchronizer.vhd**, **edgeDetector.vhd**, **ov7670programmer.vhd**, **ov7670reader.vhd**, **rgb2grey.vhd**, **freqSynthesizer.vhd**, **vgaRefresher.vhd**, **lab11VGAgrey.vhd** y **lab11VGAgrey.xdc**
6. Configurar y añadir al proyecto los IPs:
   - **multAdd** y **frameBuffer**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Conectar la cámara y el monitor a la placa y encenderla.
9. Descargar el fichero **lab11VGAgrey.bit**

J.M. Mendías
2024

Tareas

1. Crear el proyecto **lab11QVGAcolor** en el directorio **DAS**
   - **lab11QVGAcolor:** **lab11QVGAcolor.vhd** y **lab11QVGAcolor.xdc**
2. Completar **common.vhd** con la declaración del nuevo componente.
3. Completar el código omitido en el fichero:
   - **lab11QVGAcolor.vhd**
4. Añadir al proyecto los ficheros:
   - **common.vhd**, **synchronizer.vhd**, **edgeDetector.vhd**, **ov7670programmer.vhd**, **ov7670reader.vhd**, **rgb2grey.vhd**, **freqSynthesizer.vhd**, **vgaRefresher.vhd**, **lab11QVGAcolor.vhd** y **lab11QVGAcolor.xdc**
5. Configurar y añadir al proyecto los IPs:
   - **multAdd** y **frameBuffer**
6. Sintetizar, implementar y generar el fichero de configuración.
7. Conectar la cámara y el monitor a la placa y encenderla.
8. Descargar el fichero **lab11QVGAcolor.bit**

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/