Arquitecturas Reconfigurables

Desarrollo de una librería de soporte (PSL)

Profesores: Sergio Cuenca y Antonio Martínez

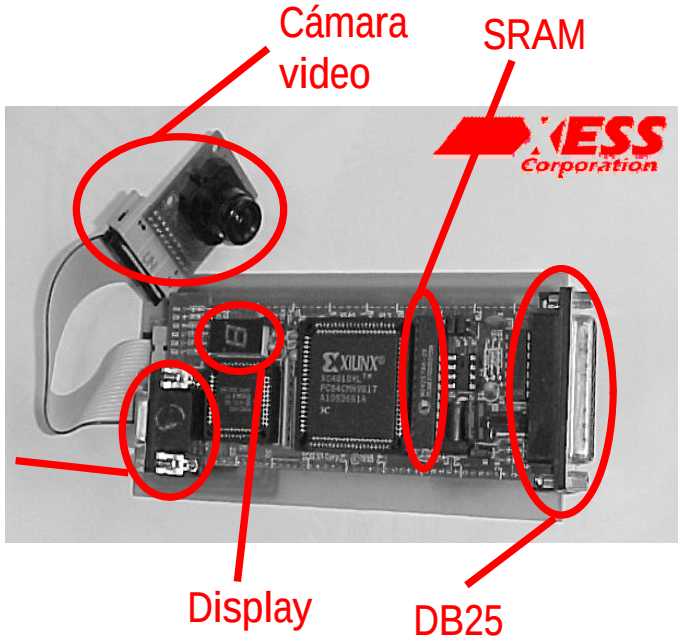
sergio@dtic.ua.es

Dept. Tecnología Informática y Computación

Universidad de Alicante

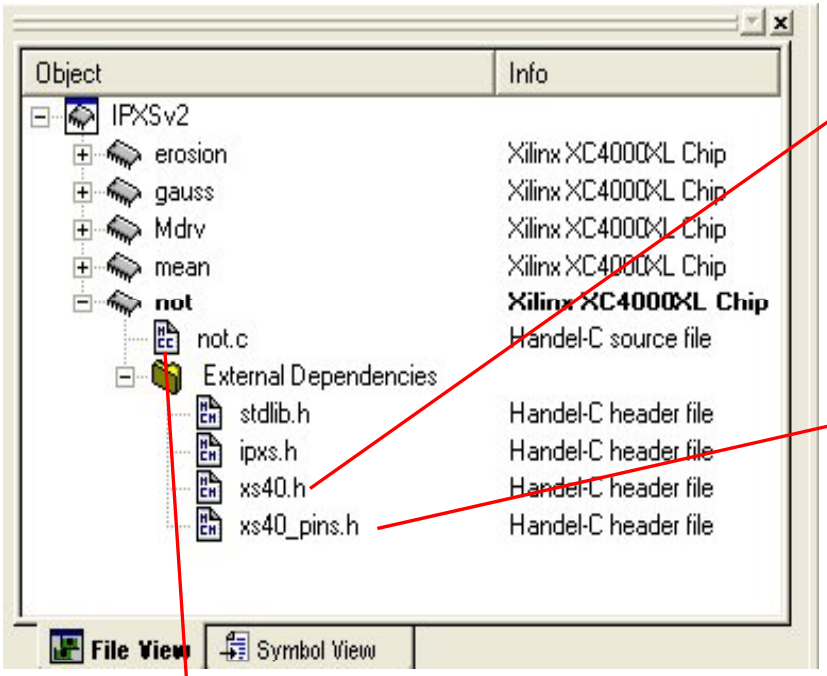
Librería de soporte para XS40

Permite escribir código HandelC independiente de la plataforma => portable a distintas tarjetas



Organización de la librería

- 2 ficheros .hch (similar a .h de ANSI-C)



- tipo de FPGA
- reloj del sistema
- drivers

- conexión a los pines de la FPGA

#include “xs40.hch”

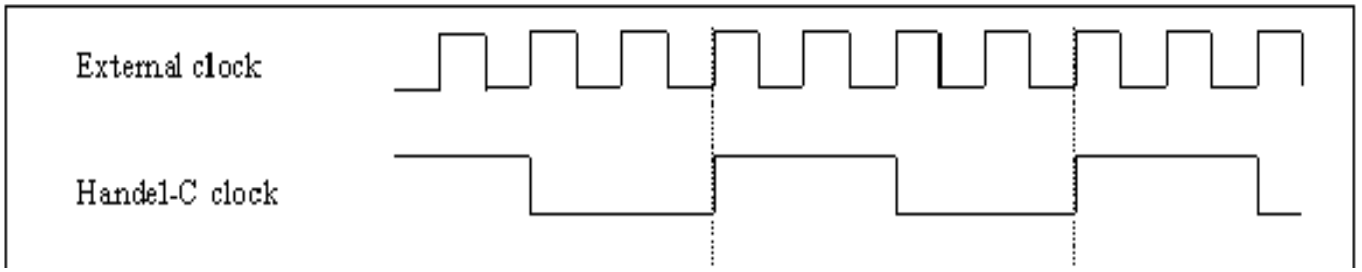
XS40_pins.hch

- Definición de las conexiones entre señal/es y pin/es de la FPGA

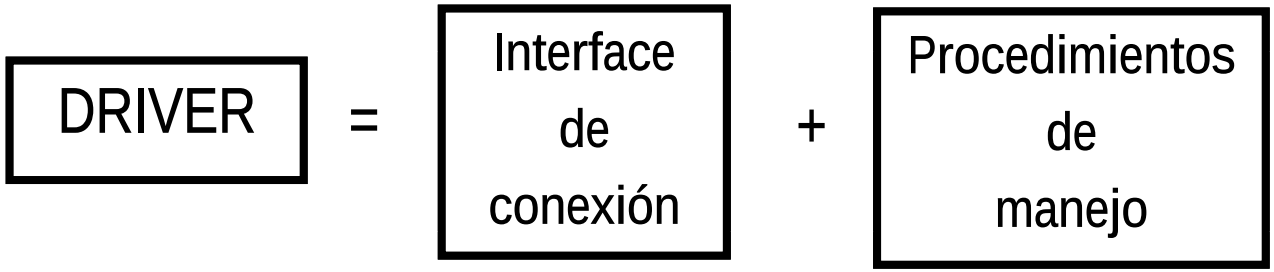
```
// SRAM Pins
// Control Bus Pins
static macro expr XS40_SRAM_OE = {"P61"};
static macro expr XS40_SRAM_WE = {"P62"};
static macro expr XS40_SRAM_CE = {"P65"};
// Data Bus Pins
static macro expr XS40_SRAM_DATA =
{"P10","P80","P81","P35","P38","P39",
"P40","P41"};
```

XS40.hch

```
// part and family definitions for XS40board
set part = "XC4010XLPC84-3";
set family = Xilinx4000XL;
// Clock definition
#ifdef XS40_DIVIDE1
set clock = external XS40_ClkPin;
#elif defined (XS40_DIVIDE3)
set clock = external_divide XS40_ClkPin 3;
#elif defined (XS40_DIVIDE4)
set clock = external_divide XS40_ClkPin 4;
#endif
```

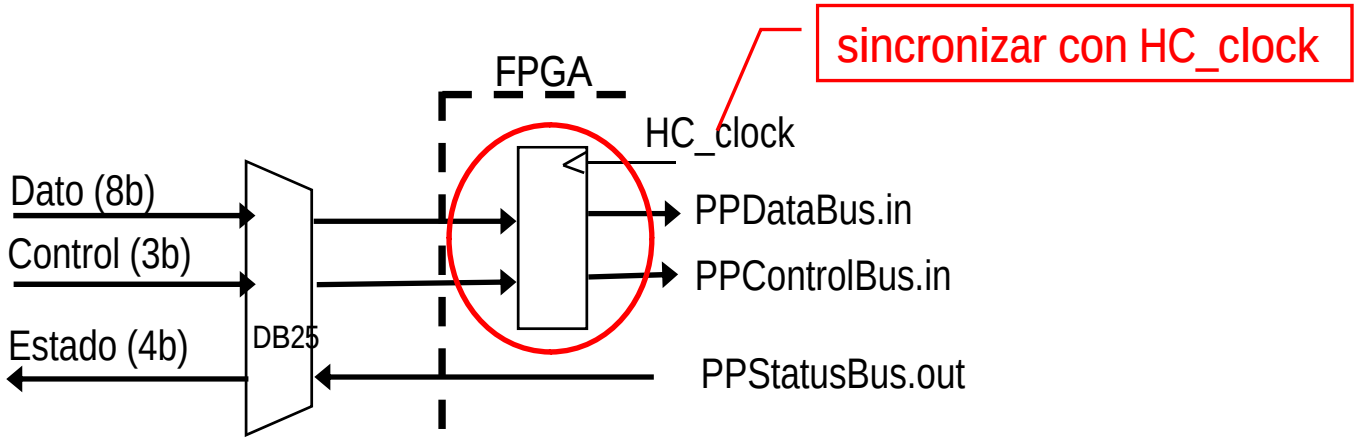


Drivers



- *Interfaces* permiten a los diseños Handel-C conectarse a hardware y lógica externa.
- `bus_in/bus_out` - conexión directa a los pines de un bus
- `bus_latch_in` bus de entrada registrado bajo ciertas condiciones
- `bus_clock_in` bus de entrada registrado con Handel-C clock
- `bus_ts` bus bidireccional triestado.
- `bus_ts_latch_in` Bus bidireccional triestado con bus de entrada registrado.
- `bus_ts_clock_in` Bus bidireccional triestado con bus de entrada registrado.
- Procedimientos:
  - Funciones
  - Macros

Conector DB25




```
interface bus_clock_in (unsigned 3) XS40_PPControlBus()with
{data={XS40_PPCPin3,XS40_PPCPin2,XS40_PPCPin1}};


interface bus_out() XS40_PPStatusBus(unsigned 4
out=XS40_PPS6@XS40_PPS5@XS40_PPS4@XS40_PPS3)
with { data={XS40_PPSPin6,XS40_PPSPin5,XS40_PPSPin4,
XS40_PPSPin3}};
```

Ejemplo

simple cambio de nombre

```
macro expr XS40_PPC1 = XS40_PPControlBus.in[0];
macro expr XS40_PPC2 = XS40_PPControlBus.in[1];
macro expr XS40_PPC3 = XS40_PPControlBus.in[2];

void main (void){
  unsigned 2 CMD;
  CMD=XS40_PPC1@XS40_PPC2; //lectura de los pines de control
  if (CMD==0)
      XS40_PPS6=0; //escritura en pin de estado
  else if (CMD==1)
      XS40_PP6=1; //escritura en pin de estado
  else
      delay;
...
}
```

Display 7 seg

Utilzación de macro procedures.

- no devuelven valores.
- el paso de parámetros se realiza por referencia
- cada llamada a una macro proc genera su correspondiente hardware asociado y por tanto no compartido.

versión_a:

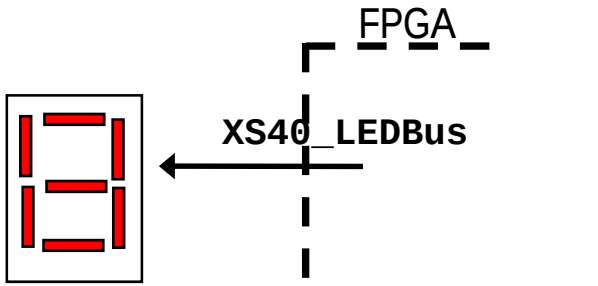
- 1 macro proc para definir el interfaz (`interface bus_out()`)
- 3 macro proc para describir los procedimientos

versión_b:

- 1 macro proc para definir conjuntamente el interfaz y los procedimientos

version_a

```
macro proc XS40LEDDriver(){
interface bus_out() XS40_LEDBus(unsigned 7
out=XS40_LEDState[6]@XS40_LEDState[5]...)
with { data = {XS40_Seg6Pin,XS40_Seg5Pin,XS40_Seg4Pin...}};
}


macro proc XS40LEDOn(LEDNumber)
{
    switch(LEDNumber<-3)
    {
    case 0:
        XS40_LEDState[0] = 1;
        break;
    case 1:
        XS40_LEDState[1] = 1;
        break;
...
}
```

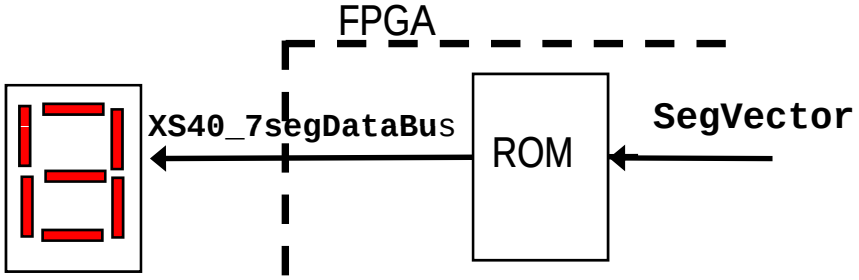


Incluir llamada al driver en código secuencial o paralelo

version_b

```
macro proc XS407SegDriver(SegVector)
{
  static rom unsigned 7 SegPatterns[16]=
  {0b1110111,0b0010010, … };

  interface bus_out() XS40_7segDataBus (unsigned 7
  XS40_7SegData) with{data= XS40_7SegPins};

  while(1)
  {
     XS40_7SegData = SegPatterns[SegVector];}
  }
}
```

Incluir llamada al driver en paralelo al resto del código



Memorias

Handel-C incluye estructuras para acceso transparente a memorias externas

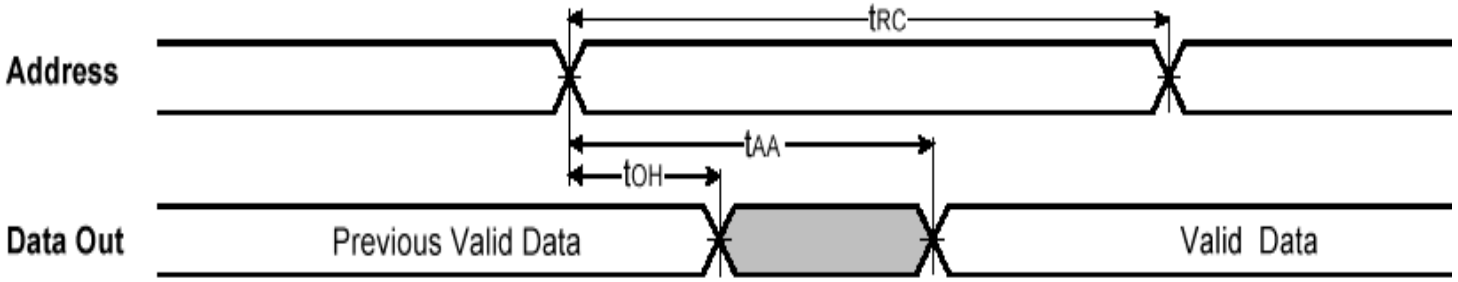
tipo de memoria SRAM, tamaño, ancho palabra

```
ram unsigned 8 XS40SRAM[32768] //32Kx8
with XS40_RAMSpec( XS40_SRAM32K_ADDR, XS40_SRAM_DATA,
XS40_SRAM_CE, XS40_SRAM_WE, XS40_SRAM_OE);
```

macro expr:
-declaración interfaz
-timing de las señales de control

Ciclo de lectura SRAM

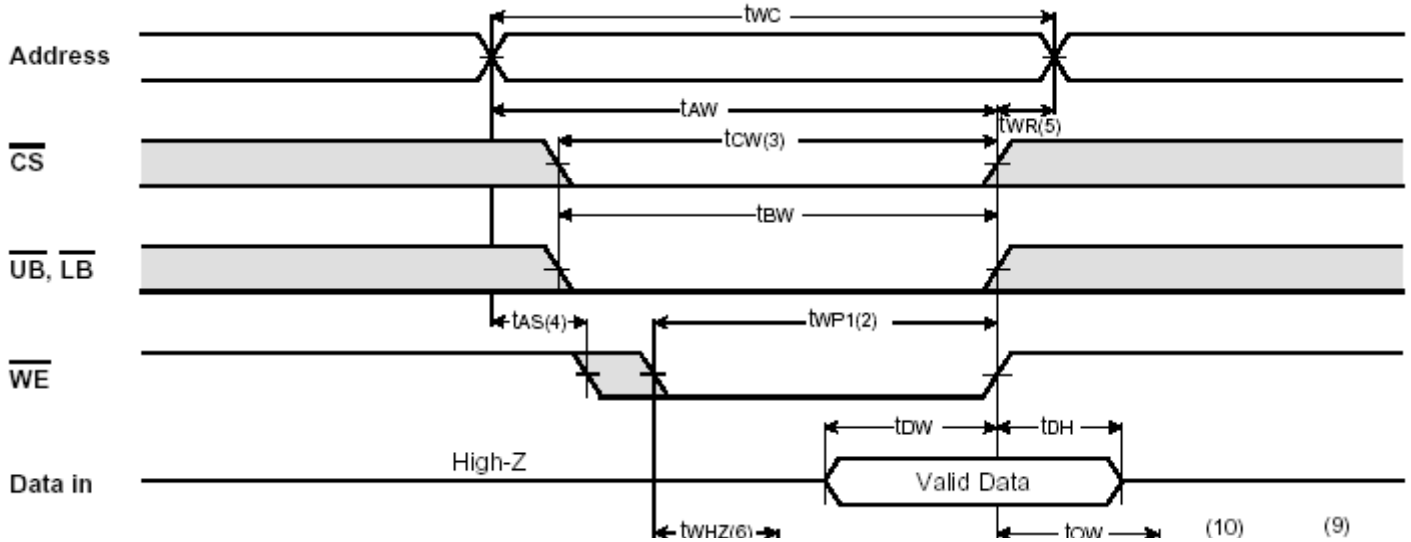
TIMING WAVEFORM OF READ CYCLE(1) (Address Controlled, $\overline{CS}$=$\overline{OE}$=V_{IL}, $\overline{WE}$=V_{IH}, $\overline{UB}$, $\overline{LB}$=V_{IL})



Parameter	Symbol	KM616V4002B/BL-10		KM616V4002B/BL-12	
		Min	Max	Min	Max
Read Cycle Time	t_{RC}	10	-	12	-
Address Access Time	t_{AA}	-	10	-	12
Chip Select to Output	t_{CO}	-	10	-	12
Output Enable to Valid Output	t_{OE}	-	5	-	6
$\overline{UB}$, $\overline{LB}$ Access Time	t_{BA}	-	5	-	6
Chip Enable to Low-Z Output	t_{LZ}	3	-	3	-
Output Enable to Low-Z Output	t_{OLZ}	0	-	0	-
$\overline{UB}$, $\overline{LB}$ Enable to Low-Z Output	t_{BLZ}	0	-	0	-
Chip Disable to High-Z Output	t_{HZ}	0	5	0	6
Output Disable to High-Z Output	t_{OHZ}	0	5	0	6
$\overline{UB}$, $\overline{LB}$ Disable to High-Z Output	t_{BHZ}	0	5	0	6
Output Hold from Address Change	t_{OH}	3	-	3	-

Ciclo de escritura SRAM

TIMING WAVEFORM OF WRITE CYCLE(2) ($\overline{OE}$=Low fixed)



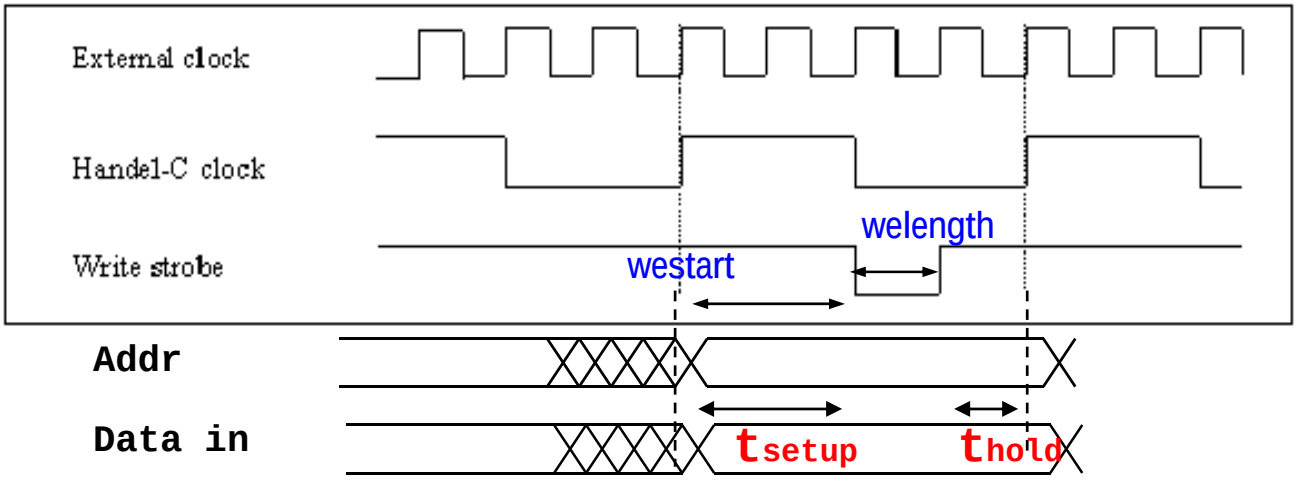
Parameter	Symbol	KM616V4002B/BL-10		KM616V4002B/BL-12	
		Min	Max	Min	Max
Write Cycle Time	tWC	10	-	12	-
Chip Select to End of Write	tCW	7	-	8	-
Address Set-up Time	tAS	0	-	0	-
Address Valid to End of Write	tAW	7	-	8	-
Write Pulse Width($\overline{OE}$ High)	tWP	7	-	8	-
Write Pulse Width($\overline{OE}$ Low)	tWP1	10	-	12	-
$\overline{UB}$, $\overline{LB}$ Valid to End of Write	tBW	7	-	8	-
Write Recovery Time	tWR	0	-	0	-
Write to Output High-Z	tWHZ	0	5	0	6
Data to Write Time Overlap	tDW	5	-	6	-
Data Hold from Write Time	tDH	0	-	0	-
End Write to Output Low-Z	tOW	3	-	3	-

Timing de los accesos

wegate:
-1: 1er 1/2 ciclo
1: 2º 1/2 ciclo
0: ciclo

memoria externa

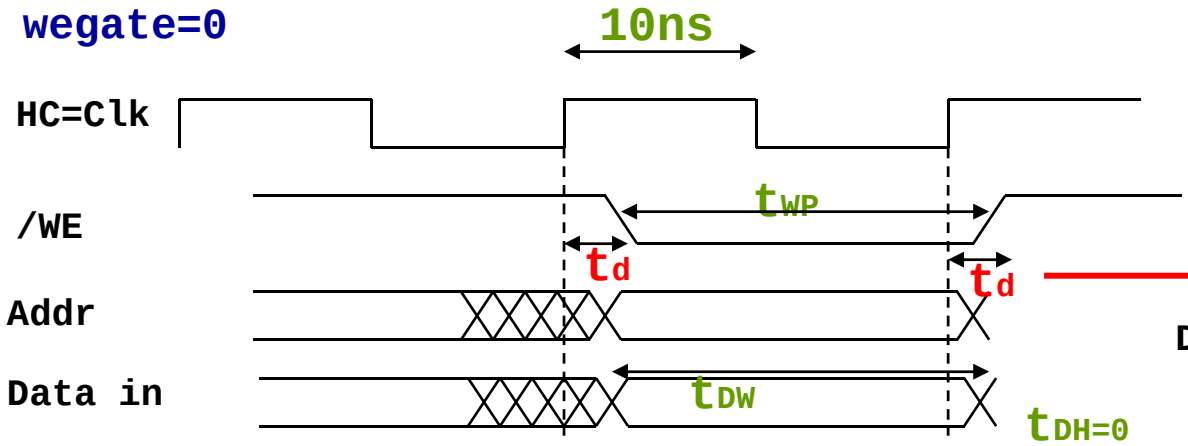
```
macro expr XS40_RAMSpec(a, d, c, w, o) ={  offchip = 1,
            addr = a, data = d, cs = c, we = w, oe = o,

                                                  interface

#ifdef XS40_DIVIDE1 wegate = 1,
#elif defined(XS40_DIVIDE3) westart = 1, welength = 1,
#elif defined(XS40_DIVIDE4) westart = 2, welength = 1,
#endif warn = 0
}
```



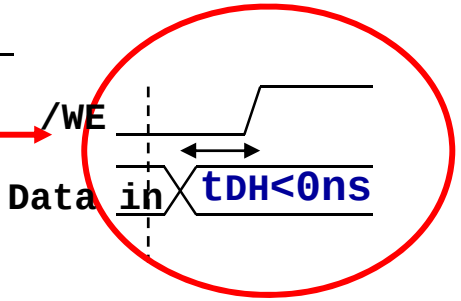
Ejemplo

Reloj externo: 50MHz
Memoria SRAM -10:tWC=10ns tWP=7ns tDW=5ns tDH=0ns
Memoria SRAM –15:tWC=15ns tWP=10ns tDW=7ns tDH=0ns

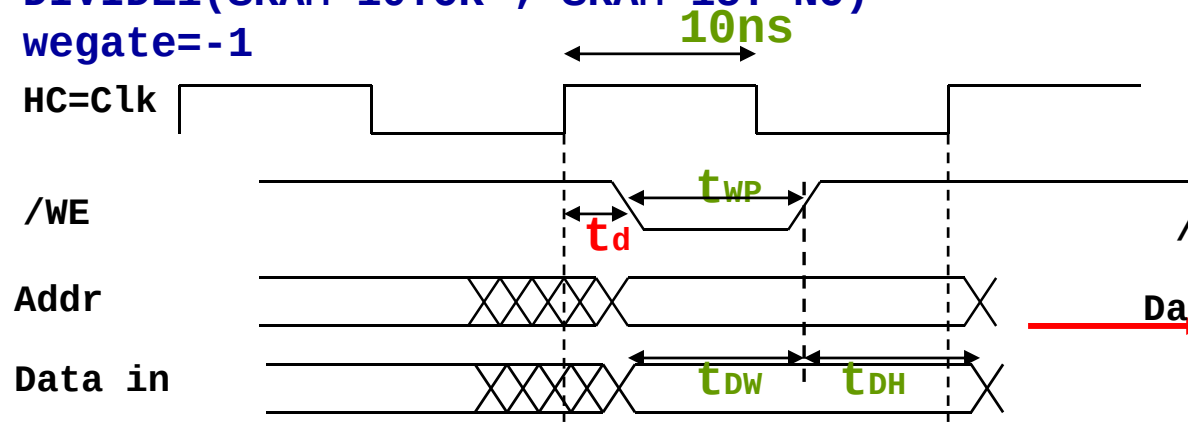
DIVIDE1 (SRAM-10:OK ; SRAM-15: OK)
wegate=0



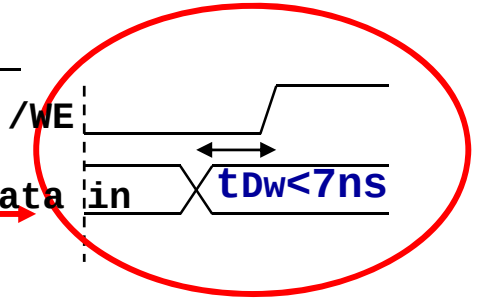
POSIBLE PELIGRO:
retardo íneas(td) distinto (Data_in cambia antes de WE)



DIVIDE1(SRAM-10:OK ; SRAM-15: NO)
wegate=-1



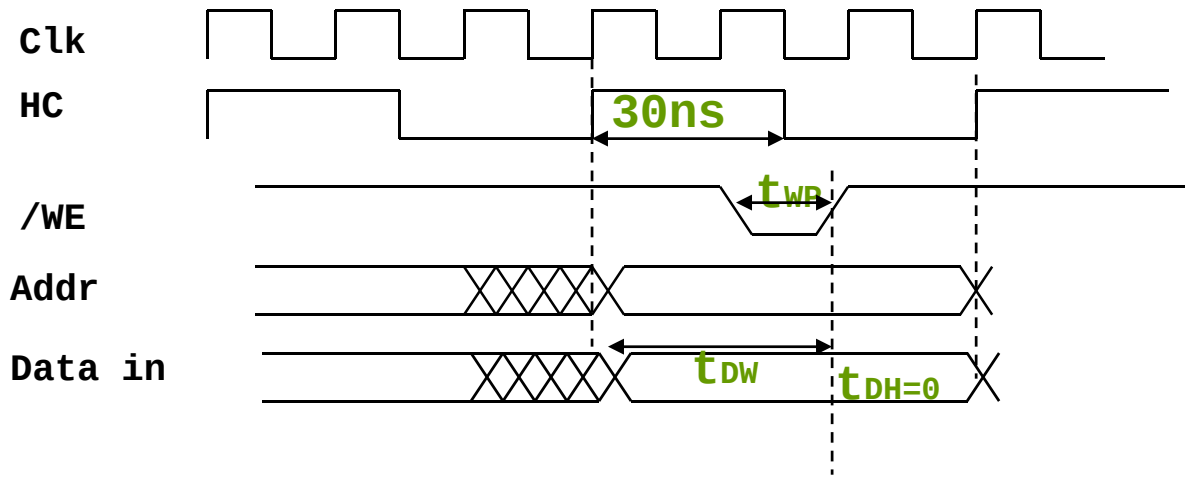
POSIBLE PELIGRO:
retardo líneas datos



Suponiendo que WE cambia al final del ciclo => td+tDW≤ 20ns

Ejemplo cont.

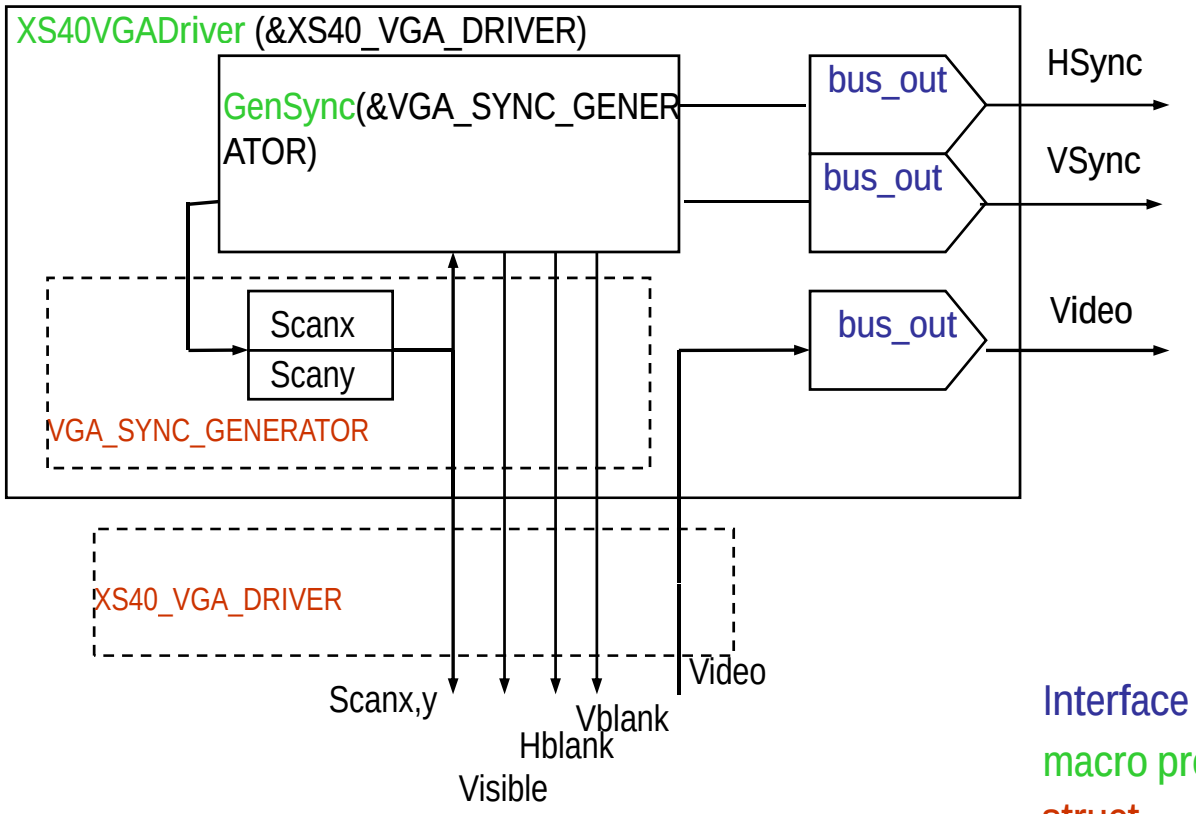
DIVIDE3(SRAM-10:OK ; SRAM-15: OK)
westart=1; welength=1



Procedimientos de R/W

```
// Mediante acceso a un array
Reg = XS40SRAM[Addr];  // lectura
XS40SRAM[Addr] = Expr; // escritura

// Mediante macros
macro proc XS40ReadSRAM(Reg, Addr){Reg =
XS40SRAM[Addr];}
macro proc XS40WriteSRAM(Addr,
Expr){XS40SRAM[Addr]= Expr;}
```

Salida VGA



Interface

macro proc

struct

Parámetros

```
// macro expressions to set up VGA timing
macro expr LineTime       = 31770;    // ns
macro expr HSyncLength    = 3770;     // ns
macro expr HVideoTime     = 25170;    // ns
macro expr HVideoToSync   = 940;      // ns
macro expr FrameTime      = 16680000; // ns
macro expr VSyncLength    = 60000;    // ns
macro expr VVideoTime     = 15250000; // ns
macro expr VVideoToSync   = 350000;   // ns

macro expr ClockRate      = XS40_ClkRate; // Hz

macro expr Time2Cycles(t) = ((ClockRate*t+500000000)/1000000000);
macro expr XS40TotalCols    = Time2Cycles(LineTime);
macro expr XS40VisibleCols  = Time2Cycles(HVideoTime);
macro expr HSyncStartCol  = Time2Cycles(HVideoTime+HVideoToSync);
macro expr HSyncEndCol    = (HSyncStartCol+Time2Cycles(HSyncLength));

macro expr XS40TotalLines   = ((FrameTime+LineTime/2)/LineTime);
macro expr XS40VisibleLines = ((VVideoTime+LineTime/2)/LineTime);
macro expr VSyncStartLine = ((VVideoTime+VVideoToSync+LineTime/2)/LineTime);
macro expr VSyncEndLine   = (VSyncStartLine+((VSyncLength+LineTime/2)/LineTime));
```

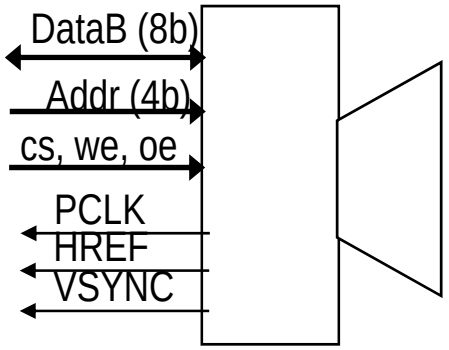
Parámetros

```
macro proc XS40VGADriver(VideoRecPtr)
{VGA_SYNC_GENERATOR SyncGen;     // structure to pass to SyncGen
 unsigned 1 Hsync, Vsync;
 unsigned 6 Video;
 // interfaces for H-sync, V-sync and video data.
   interface bus_out() HSyncOut (Hsync) with {data = XS40_hsync_pin};
   interface bus_out() VSyncOut (Vsync) with {data = XS40_vsync_pin};
   interface bus_out() VideoOut (Video) with {data = XS40_video_pins};
par
    { //Update the externally visible signals every cycle
          do
          {
            par
            {
             VideoRecPtr->ScanX = SyncGen.ScanX;
             VideoRecPtr->ScanY = SyncGen.ScanY;
             VideoRecPtr->Visible = SyncGen.Visible;
             VideoRecPtr->HBlank = SyncGen.HBlank;
             VideoRecPtr->VBlank = SyncGen.VBlank;

             Hsync = SyncGen.HSync;
             Vsync = SyncGen.VSync;
             Video = VideoRecPtr->Output;
            }
          }while(1);
          // Call the SyncGen process to generate the signals.
          VGASyncGen(&SyncGen);
   }
}
```

Conexión a cámara CMOS

M4088:

- Scan progresivo (no entrelazado) y salida digital
- Parámetros ajustables escribiendo en sus registros internos. Se acceden de forma similar a una SRAM
- Señal de video (8bits) se obtiene también a través del bus de datos leyendo el registro VPORT, que se actualiza con el flanco de la señal de reloj de la cámara, PCLK, mientras las sincronías HREF y VSYNC están activas



Opción_A

Driver:

- Estructura de memoria SRAM
- Interfaz con las señales de sincronía

```
ram unsigned 8 M4088Reg[16] with M4088_RAMSpec( M4088_Reg_ADDR,
M4088_Reg_DATA, M4088_Reg_WE, M4088_Reg_OE);

interface bus_clock_in (unsigned 3) M4088_SyncBus() with
{data = {M4088_VSYNCPin, M4088_HREFPin, M4088_PCLKPin}};
```

- Configurar cámara => escribir en SRAM
- Capturar videio => comprobar las señales de sincronía y leer SRAM

Ejemplo: frame-grabber (i)

```
#define XS40_SRAM32K //carga el driver SRAM
#define XS40_DIVIDE1 //utiliza el external clock
#define M4088Camera  //carga driver cámara
void main (void){
   unsigned 2 CMD;   // Stop/Capturar/Configurar
   while (1)  {
      CMD=XS40_PPC1@XS40_PPC2; // lectura pines cntrl
      if (CMD==CAPTURA)       {GetFrm();}
      else if (CMD==CONFIGURA) {M4088Config();}
      else {delay;}
   }
}
```

Ejemplo: frame-grabber (ii)

Función para configurar cámara

```
void M4088Config(){
  unsigned 4 regaddr;
  static rom unsigned 8 CamConf[8] =
        {0b1110111,0b0010010,0b1011101,, ……};
  regaddr=1; // inicializa dirección primer registro
    do {
       M4088WriteReg(regaddr,CamConf(regaddr));
       regaddr++;
    }while(regaddr!=9);
}
```

Ejemplo: frame-grabber (iii)

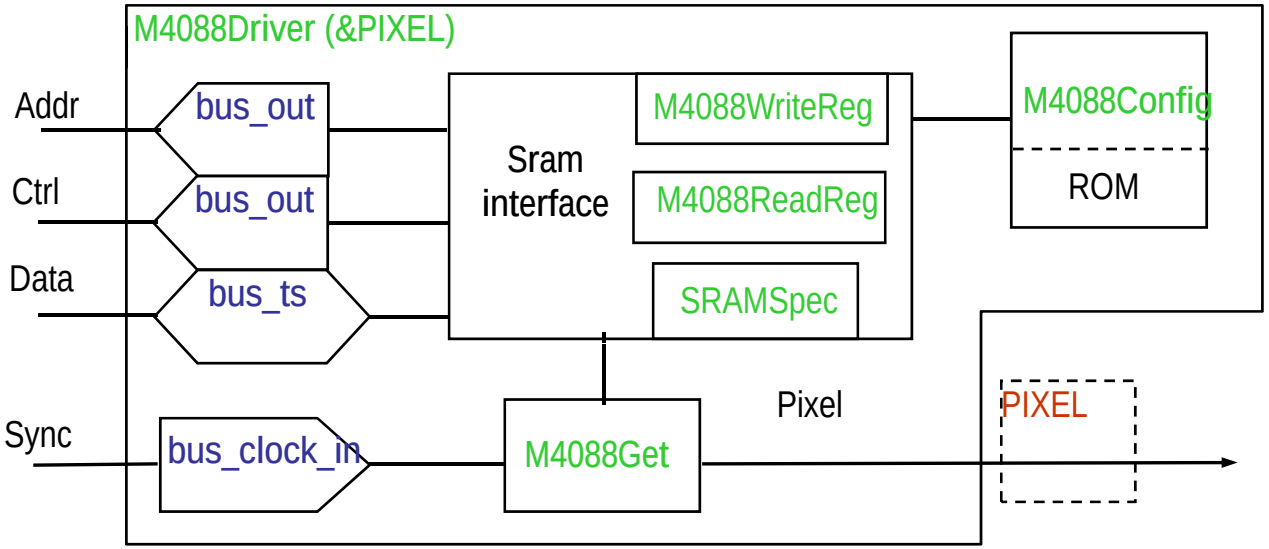
Función para capturar imagen

```
void GetFrm(){
  macro expr RisingEdgePCLK=(Old_PCLK==0)&&(MPCLK==1);
  par {Addr = 0;Old_PCLK=1;XS=0;} //inic. variables
  while (!MVSYNC) {delay;} //espera VSYNC=1
  while (MVSYNC) {delay;}  //espera VSYNC=0
  do { // captura imagen
       par{
           M4088ReadReg(pixel,VPORT);
           if (RisingEdgePCLK&&(MHSYNC==1)){
              par{
                 XS40WriteSRAM(AddrWrite,pixel);
                 AddrWrite++;}
              }
        Old_PCLK=MPCLK;
       } //par
  }while (!MVSYNC); //fin de imagen
  XS40_PPS7=1;   //notifica imagen capturada
}
```

Opción_B

Driver:

- Incluye las rutinas para realizar la captura de los pixels
- Configuración contenida en una ROM interna



Interface

macro proc/functions

chan

Opción_C

2 dominio de reloj (PCLK, HC) y comunicación mediante RAM de doble puerto y canales