Universidad de Alcalá

Práctica libre: Generador de funciones con Matlab y FPGA

ASIGNATURA
Modelado y Síntesis de Sistemas Electrónicos Digitales

Grado en Ingeniería de Computadores

Universidad de Alcalá

Curso Académico 2014/2015

Curso 3º – Cuatrimestre 2º

Dpto. Electrónica

1. Introducción

La siguiente práctica tiene como objetivo el desarrollo en VHDL de un sistema digital que permite cubrir todas las fases del flujograma de diseño de sistemas electrónicos para configurar FPGAs u otro tipo de ASICs: modelado, simulación, síntesis, implementación y descarga en placa.

Para su implementación se utilizarán la tarjeta de pruebas del laboratorio Atlys de Digilent basada en una FPGA Spartan 6 (XC6SLX45-CSG324) de la firma comercial Xilinx que incorpora distintos periféricos de entrada/salida.

Concretamente, se desea diseñar un generador de funciones que proporciona dos señales analógicas cuyas formas serán diferentes y controladas desde una aplicación ejecutada en Matlab. Ambas señales tendrán la misma frecuencia, que será seleccionada desde esta aplicación. La comunicación entre el PC y la tarjeta Atlys se va a realizar utilizando un cable USB.

Además de alcanzar este objetivo, la presente práctica busca que el alumno pueda realizar un primer diseño de forma libre siguiendo unas pautas mínimas. Para ello el alumno deberá resolver problemas habituales en el diseño con FPGAs como son la búsqueda del tamaño óptimo de datos, sincronización entre etapas, diseño de máquinas de estado (FSMs) así como la instanciación de componentes diseñados previamente.

Los problemas a los que el alumno se enfrentará en esta práctica son los habituales que se encontrará en diseños más complejos.

2. Estructura

El diseño a realizar consiste en el modelado en VHDL de un sistema digital que permite, una vez programado en una FPGA, generar dos señales analógicas Vo1 y Vo2 (figura 1) cuyas formas y frecuencia se van a poder seleccionar desde un archivo creado y ejecutado en Matlab, estos valores serán enviados a través de una conexión USB.

Desde la aplicación de Matlab se enviará a través de un cable USB un periodo, formado por 256 muestras, correspondientes a la forma de onda de las señales Vo1 y Vo2. Cada una de las muestras tendrá un tamaño de 8 bits. A su vez, la frecuencia de dichas señales también se podrá seleccionar desde la aplicación de Matlab utilizando un dato de 8 bits. Cada uno de los datos generados por la aplicación de Matlab está constituido por dos campos: dirección y valor. Con la dirección se selecciona el destino del campo valor pudiéndose trabajar con 3 distintas:

- $A1_{HEX}$. Dato correspondiente a la forma de onda de la señal Vo1.
- $A2_{HEX}$. Dato correspondiente a la forma de onda de la señal Vo2.
- $F0_{HEX}$. Dato correspondiente a la selección de la frecuencia de Vo1 y Vo2.

Los datos proporcionados por la aplicación de Matlab se envían a la placa de desarrollo Atlys a través de un cable USB. En esta última se encuentra el driver CY7C68013A que convierte los datos recibidos a un formato paralelo (EPP- Enhanced Parallel Port) y se envían a la FPGA. Cada uno de los datos correspondientes a las formas de onda de Vo1 y Vo2 tiene un tamaño de 8 bits, si bien se envían, desde el PC, en un formato de 8 bits. Por cada dato se realizan dos transferencias desde el PC. Para la frecuencia se utilizará solamente un dato de 8 bits.

Los datos almacenados se enviarán a dos conversores digitales analógicos DAC121S101 de National Semiconductor presentes en la placa PmodDA1 de Digilent. Para introducir los datos en el DAC se utiliza un protocolo serie de tres líneas: reloj (***SCLK***), dato de entrada (***DIN***) y un pin de selección (***SYNC***).

Las conexiones entre los diferentes elementos de la práctica se muestran en la Figura 1.

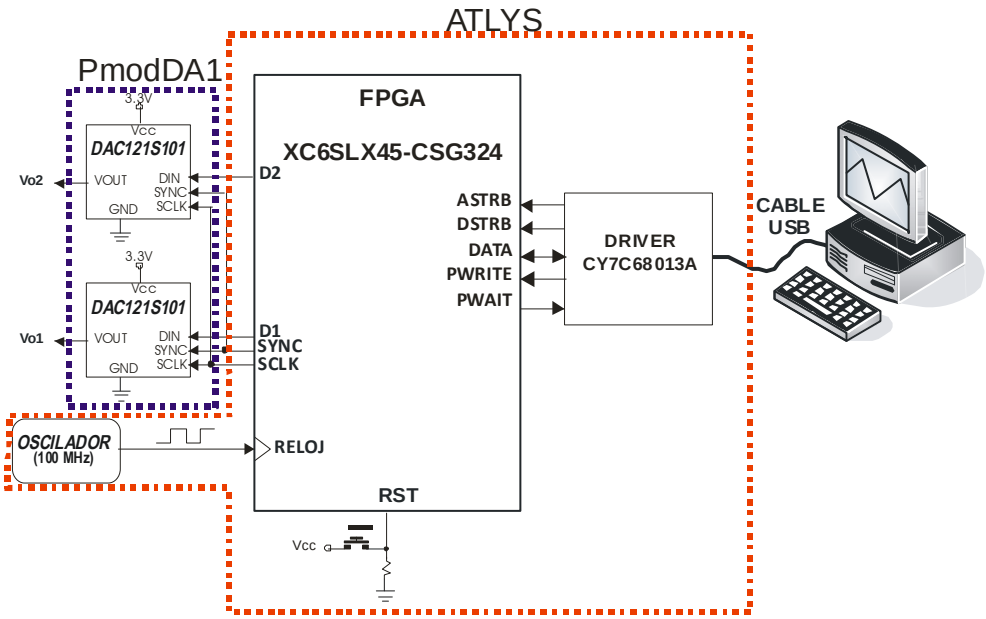

Figura 1 Conexión del hardware a utilizar.

El sistema a diseñar hace uso de un RELOJ de 100 MHz como señal de sincronismo de los diferentes módulos secuenciales. Esta señal es generada por un oscilador presente en la placa Atlys. Por su parte, la entrada ***RST*** se va a utilizar como señal asíncrona de inicialización para llevar todos los módulos secuenciales a su estado inicial.

3. Entidad gen_funciones.

Con el fin de facilitar el trabajo, para el modelado del sistema de medida se realizará un diseño jerárquico en el que la entidad de mayor nivel se llamará ***gen_funciones***, y cuya definición se realizará en el archivo **gen_funciones.vhd** siendo su declaración la mostrada en la Figura 2.

```
entity gen_funciones is
  port (
    RELOJ  : in  std_logic;
    RST    : in  std_logic;
    -- PUERTO EPP
    ASTRB   : in  std_logic;
    DSTRB   : in  std_logic;
    DATA    : inout  std_logic_vector(7 downto 0);
    PWRITE : in  std_logic;
    PWAIT  : out std_logic;
    -- DAC
    SYNC    : out std_logic;
    SCLK    : out std_logic;
    D1      : out std_logic;
    D2      : out std_logic);
end gen_funciones;
```

Figura 2 Declaración de la entidad *gen_funciones*.

Para modelar la entidad ***gen_funciones*** se va a hacer uso de seis componentes (Figura 3). En la documentación de la práctica se proporcionan "los esqueletos" de archivos VHDL para modelar estos elementos, con excepción del ***DCM***, los cuales se deberán completar con el código necesario para que realicen la función que posteriormente se detallará. Si bien, **no se podrán modificar los puertos de cada una de las entidades**. Además, el diseño de los componentes anteriores no podrá ser jerárquico y los nombres de las señales a utilizar para conectar los diferentes componentes de la entidad ***gen_funciones*** deberán ser los utilizados en la Figura 3. El incumplimiento de las restricciones anteriores conlleva a una calificación de **No Apto**.

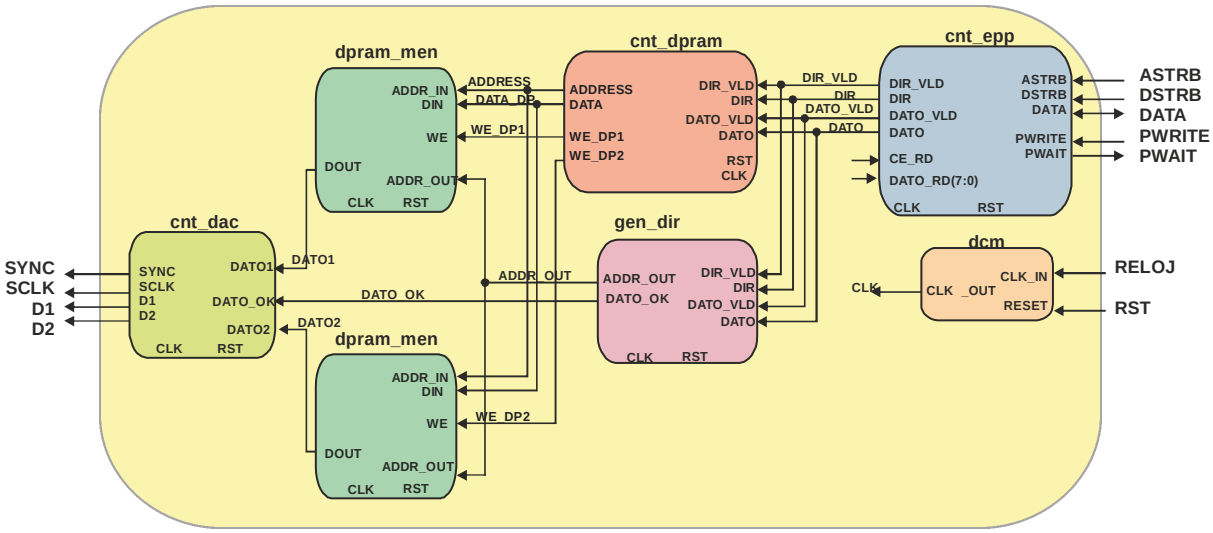

Figura 3 Estructura de la entidad *gen_funciones.*

Brevemente, la funcionalidad de cada una de las entidades de la Figura 3 es:

dcm. Es el gestor de reloj digital usado para la gestión de las señales de reloj de los módulos secuenciales. Se utiliza para reducir la asimetría de los relojes compensando el posible desfasaje entre ellas.

cnt_epp. Entidad que recibe los datos con la temporización establecida en el protocolo EPP y proporciona dos salidas ***DIR*** y ***DATO***, ambas de 8 bits con el valor de la dirección y dato incluidos. Se proporcionan sendas señales de validación: ***DIR_VLD*** para indicar que hay una nueva dirección y ***DATO_VLD*** para indicar que hay un nuevo dato. Ambas señales permanecen a nivel alto un periodo de la señal de reloj ***CLK*** (10 ns).

dpram_mem. Entidad que modela una memoria de doble puerto en la que se guarda cada una de las 256 muestras correspondientes a un periodo de las señales Vo1 y Vo2. Cada muestra tiene un tamaño de 8 bits. De los dos puertos uno es de escritura y el otro de lectura.

cnt_dac. Entidad que realiza la conversión al protocolo serie del dato de 8 bits a introducir en cada uno de los DACS. La entrada ***DATO_OK*** se utiliza como sincronismo, de forma que pasa a nivel alto, durante un periodo de la señal de reloj, para indicar que hay un nuevo dato a enviar a los DACs.

gen_dir. Entidad que se encarga de proporcionar la dirección correspondiente a cada una de las muestras de un periodo de las señales Vo1 y Vo2. El ritmo al que se cambian esta

dirección variará en función del valor de frecuencia seleccionado desde la aplicación de Matlab.

cnt_dpram. Entidad que gestiona la carga de los datos proporcionados por el PC, a través del protocolo EPP, en cada una de las memorias dual-port.

4. Entidad *cnt_epp*.

La entidad ***cnt_epp*** es la encargada de actuar como interface entre el puerto USB del ordenador y el diseño. Los diferentes parámetros de las señales de salida son enviados desde un PC conectado a la tarjeta de FPGAs mediante dicho puerto.

La tarjeta Atlys, tal y como se puede ver en la documentación del fabricante (http:/www.digilentinc.com), posee un puerto USB (Figura 4) que comunica el PC con la FPGA a través de un driver de Cypress (CY7C68013A). Este driver se encarga de transformar los datos enviados con un protocolo USB a un protocolo EPP (Enhanced Parallel Port). Se trata de un protocolo paralelo que opera casi a la velocidad del bus ISA y ofrece un incremento de hasta 10 veces la velocidad de transmisión sobre un puerto paralelo convencional.

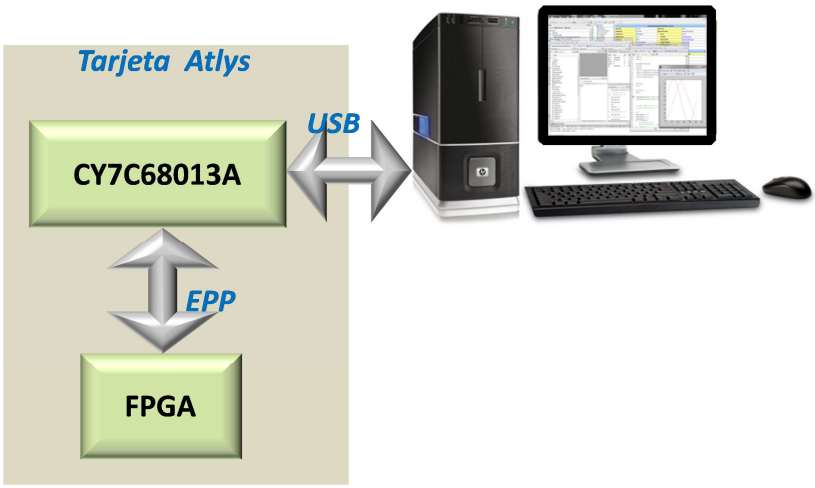

Figura 4 Comunicación PC- CY7C68013A-FPGA.

El protocolo EPP permite realizar comunicaciones paralelo con velocidades hasta 2 Mbyte/s. Para ello emplea un único bus ***DATA*** (Figura 5) para realizar transferencias de datos y de direcciones. Utilizando dos líneas adicionales, ***ASTRB*** (address strobe) y ***DSTRB*** (data strobe), para discernir entre ellos: ***ASTRB*** activa a nivel bajo indica que está teniendo lugar una transferencia de direcciones y ***DSTRB*** se utiliza para indicar que la transferencia en curso es de datos. Además, se dispone de la línea ***PWRITE*** (Write enable) para seleccionar el sentido (lectura o escritura) de la transferencia de datos: un nivel alto para la lectura y un nivel bajo para la escritura. Por otro lado, el puerto ***PWAIT***, generado por el periférico, permite prolongar los ciclos de acceso insertando estados de espera. Para esta práctica no es necesario prolongar los ciclos de acceso.

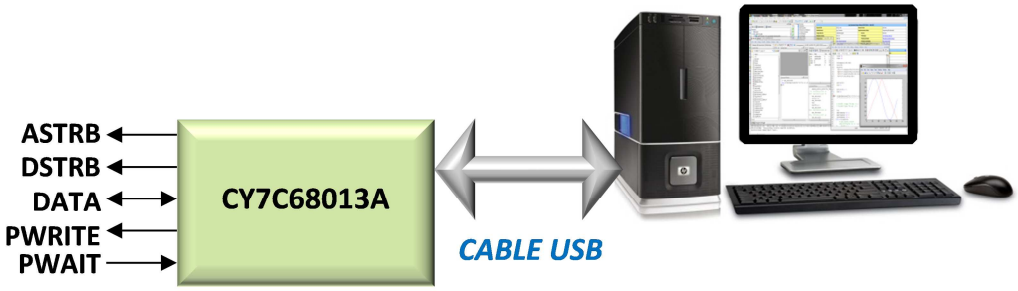

Figura 5 Señales empleadas en el protocolo EPP.

Un ciclo de lectura (Figura 6) se realiza cuando el PC quiere acceder al contenido de una posición del periférico, en este caso la FPGA, especificada por el valor indicado por el bus ***DATA*** (Dirección *en la* Figura 6). El ciclo se divide en dos partes: escritura de la dirección y lectura del dato. El ciclo de escritura de la dirección se inicia cuando ***PWRITE*** pasa a nivel bajo, continuando con un nivel bajo en ***ASTBR***, a lo que el periférico responde con un nivel alto en ***PWAIT***. Durante el nivel bajo de ***ASTBR*** se proporciona la dirección del periférico a la que se quiere acceder. El ciclo de escritura de la dirección finaliza cuando ***ASTBR*** pasa a nivel alto, instante que puede utilizar el periférico para leer el valor de la dirección. Si el periférico desea introducir estados de espera mantendrá ***PWAIT*** a nivel alto, en caso contrario la lleva a nivel bajo. A continuación se realiza el ciclo de lectura del dato, en el que es el periférico el que proporciona el dato, el cual debe estar estable antes de que se produzca el flanco de subida de ***DSTRB***. El periférico puede utilizar el flanco de bajada de esta señal para poner el dato en el puerto ***DATA***.

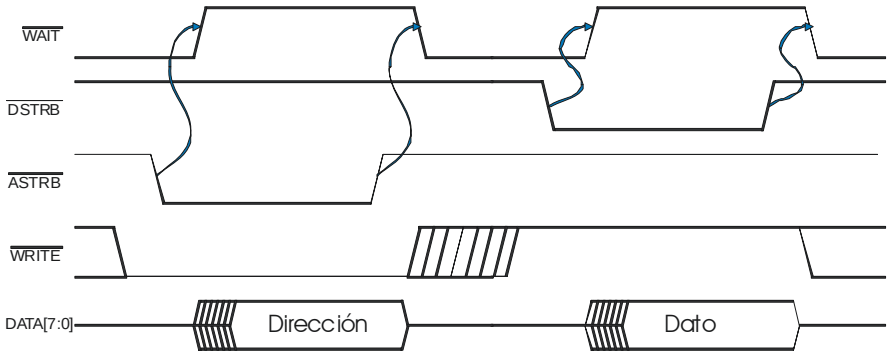

Figura 6 Ciclo de lectura del protocolo EPP.

Un ciclo de escritura (Figura 7) es aquel que realiza el PC para escribir un dato en una posición del periférico. En este caso, se realizan dos ciclos de escritura: de dirección y de dato. En la primera parte del ciclo se procede a escribir la dirección, de forma similar al de la Figura 6 y en la segunda parte del ciclo se procede a escribir un dato. En ella, la señal ***PWRITE*** se mantiene a nivel bajo, a la vez que el PC pone el dato a escribir en el periférico en ***DATA***. El periférico puede utilizar el flanco de subida de ***DSTRB*** para leer el dato.

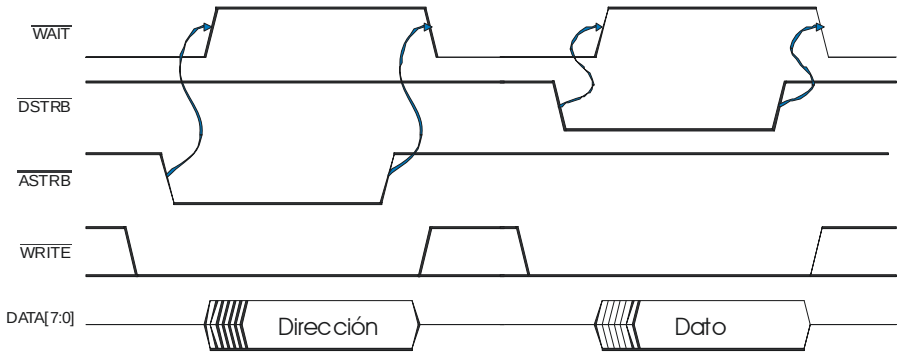

Figura 7 Ciclo de escritura del protocolo EPP.

De acuerdo con el diagrama de bloques de la Figura 3, la declaración de la entidad ***cnt_epp***. Se muestra en la Figura 8.

```
entity cnt_epp is
  port (
    CLK      : in    std_logic;
    RST      : in    std_logic;
    ASTRB    : in    std_logic;
    DSTRB    : in    std_logic;
    DATA     : inout std_logic_vector(7 downto 0);
    PWRITE   : in    std_logic;
    PWAIT    : out   std_logic;
    DATO_RD  : in    std_logic_vector(7 downto 0);
    CE_RD    : out   std_logic;
    DIR      : out   std_logic_vector (7 downto 0);
    DIR_VLD  : out   std_logic;
    DATO     : out   std_logic_vector (7 downto 0);
    DATO_VLD : out   std_logic);
end ;
```

Figura 8 Declaración de la entidad *cnt_epp*.

El funcionamiento de los puertos, a parte de los implicados en el protocolo EPP, de la Figura 8 es:

- ***DIR***. Valor de la dirección de un ciclo de acceso.
- ***DIR_VLD***. Indica con un nivel alto y durante un periodo de la señal ***CLK*** que en ***DIR*** se encuentra la dirección de un ciclo de lectura o escritura.
- ***DATO***. Valor del dato correspondiente a un ciclo de escritura.
- ***DATO_VLD***. Indica con un nivel alto y durante un periodo de la señal ***CLK*** que en ***DATO*** hay un dato válido de un ciclo de escritura.
- ***DATO_RD***. Dato de entrada a ser leído en un ciclo de lectura.
- ***CE_RD***. Señal de habilitación que permanece a nivel bajo el mismo tiempo que la señal ***DSTRB*** durante un ciclo de lectura.

En la Figura 9 se puede ver un cronograma detallado de la activación de los diferentes puertos de la entidad ***cnt_epp***, tanto para un ciclo de lectura como de escritura.

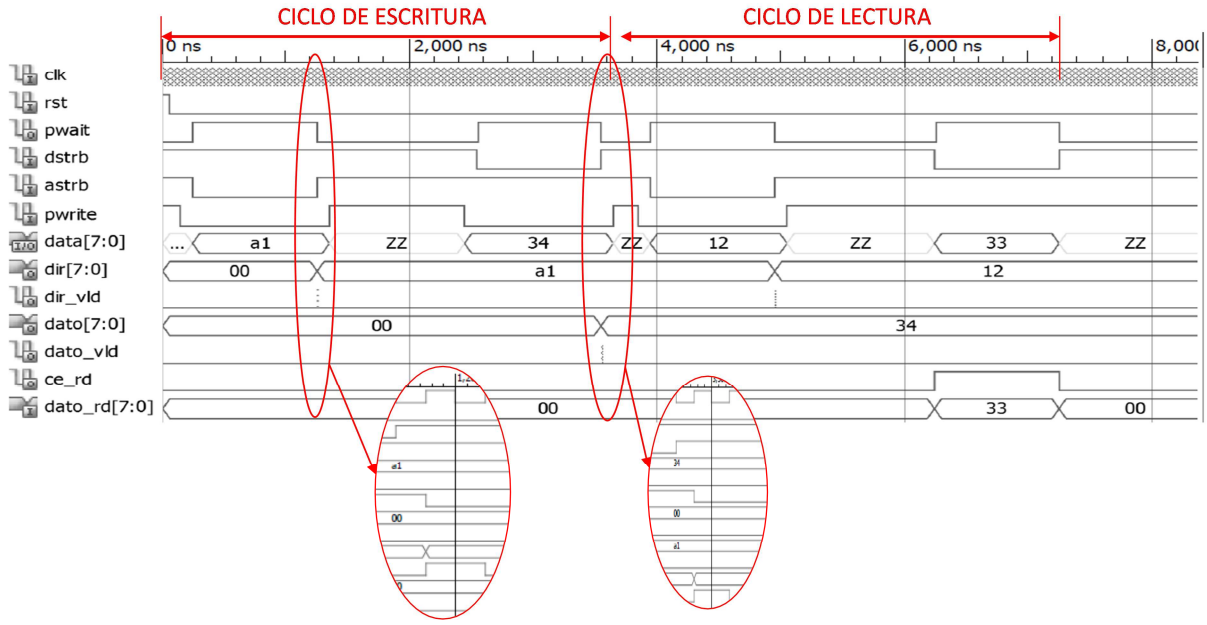

Figura 9. Cronogramas de funcionamiento del protocolo EPP.

En esta práctica, todos los ciclos que se realizarán serán de escritura. Sin embargo, la entidad ***cnt_epp*** debe realizar la decodificación tanto de los ciclos de escritura como los de lectura.

5. Entidad *cnt_DAC*.

La entidad ***cnt_dac*** **¡Error! No se encuentra el origen de la referencia.** es la encargada de proporcionar las señales de control de los dos DAC121S101 de la placa PmodDA1, los cuales van a proporcionan las tensiones de salida Vo1 y Vo2 a partir de los datos almacenados en las memorias dual port.

El DAC121S101 es un convertidor digital analógico de 12 bits con una interface serie que acepta diferentes protocolos (SPI™, QSPI, MICROWIRE). Para ello utiliza tres líneas de control: ***Din*** para introducir los bits del dato a convertir y ***SCLK*** y ***SYNC*** que sincronizan dicha transferencia. Para una correcta operación, la transferencia de datos debe cumplir el cronograma de la Figura 10, nótese como en la entrada D_{IN} se proporciona un nuevo valor coincidiendo con el flanco de subida de ***SCLK***. Las señales mostradas en el cronograma de la Figura 10 deben cumplir los tiempos mostrados en la Tabla 1.

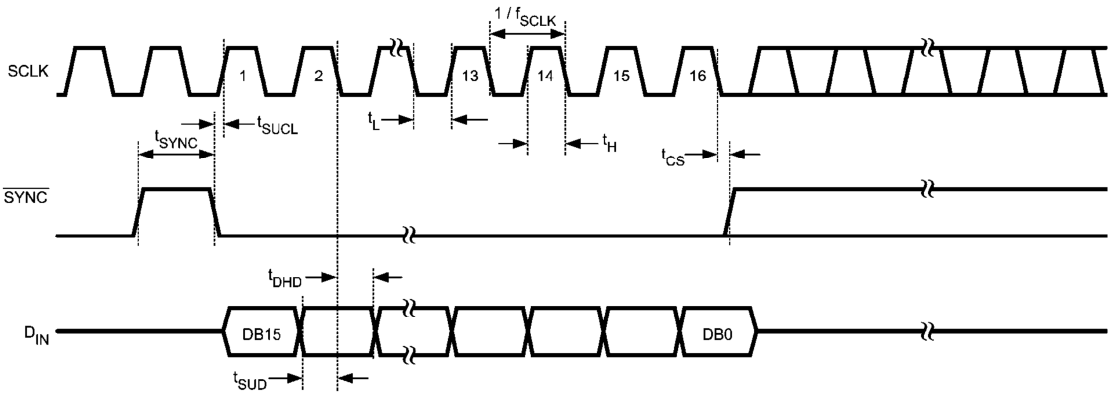

Figura 10 Cronograma de funcionamiento del DAC121S101.

Tabla 1. Valores límites de los tiempos de la Figura 10.

Symbol	Parameter	Conductions		Typical	Limits	Units (Limits)
f_{SCLK}	SCLK Frequency				30	MHz (max)
t_s	Output Voltage Settling Time (*Note 10*)	400h to C00h code change, R_L = 2kΩ	$C_L \le 200$ pF	8	10	µs (max)
			C_L = 500 pF	12		µs
		00Fh to FF0h code change, R_L = 2kΩ	$C_L \le 200$ pF	8		µs
			C_L = 500 pF	12		µs
SR	Output Slew Rate			1		V/µs
	Glitch Impulse	Code change from 800h to 7FFh		12		nV-sec
	Digital Feedthrough			0.5		nV-sec
t_{WU}	Wake-Up Time	V_A = 5V		6		µs
		V_A = 3V		39		µs
$1/f_{SCLK}$	SCLK Cycle Time				33	ns (min)
t_H	SCLK High time			5	13	ns (min)
t_L	SCLK Low Time			5	13	ns (min)
t_{SUCL}	Set-up Time $\overline{SYNC}$ to SCLK Rising Edge			−15	0	ns (min)
t_{SUD}	Data Set-Up Time			2.5	5	ns (min)
t_{DHD}	Data Hold Time			2.5	4.5	ns (min)
t_{CS}	SCLK fall to rise of SYNC	V_A = 5V		0	3	ns (min)
		V_A = 3V		−2	1	ns (min)
t_{SYNC}	$\overline{SYNC}$ High Time	2.7 ≤ V_A ≤ 3.6		9	20	ns (min)
		3.6 ≤ V_A ≤ 5.5		5	10	ns (min)

Como se puede extraer de la Figura 10 el DAC recibe 16 bits: 12 del dato, más 4 (los de mayor peso) de control (Figura 11). Para esta práctica todos los bits de control deben ser 0.

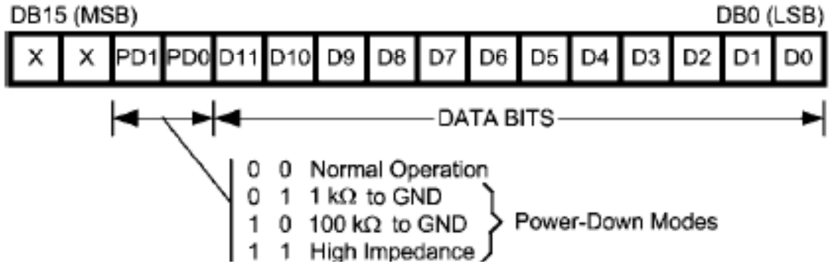

Figura 11 Trama de datos a enviar al DAC.

El valor de la tensión *V* que proporciona el DAC121S101 viene dado por:

$$V = \frac{DATO * 3.3}{4096}$$

donde *DATO* se corresponde con el dato binario de 12 bits introducido en el DAC. Junto con este enunciado se proporciona el archivo ***DAC121S101.pdf*** donde se puede encontrar una descripción más detalla del funcionamiento del conversor.

La entidad ***cnt_adc*** se debe diseñar de forma que cada vez que se activa (nivel alto) ***DATO_OK*** se inicia el proceso de transferencia de los datos ***DATO1*** y ***DATO2*** a los DACs. Cuando ***DATO_OK*** pasa a nivel bajo se generan las señales ***SYNC***, ***SCLK*** y ***DIN***. Una vez que se han transferido todos los bits de los datos, el proceso se para hasta una nueva activación de ***DATO_OK***. La declaración de la entidad ***cnt_dac*** se muestra en la Figura 12 .

```
entity cnt_dac is
  port (
    CLK     : in  std_logic;
    RST     : in  std_logic;
    DATO1   : in  std_logic_vector(7 downto 0);
    DATO2   : in  std_logic_vector(7 downto 0);
    DATO_OK : in  std_logic;
    SYNC    : out std_logic;
    SCLK    : out std_logic;
    D1      : out std_logic;
    D2      : out std_logic);
end cnt_dac;
```

Figura 12 Declaración de la entidad *cnt_dac*.

Como se puede observar en Figura 12, a la entidad *cnt_dac* le llegan sendos datos de 8 bits (***DATO1*** y ***DATO2***) para ser enviados a los DACs, cuando estos trabajan datos de 12 bits. Así, ***DATO1*** y ***DATO2*** se corresponden con los 8 bits de mayor peso (bits 11 a 4) del dato de los DACs, mientras que los 4 de menor peso (bits 3 a 0) serán 0.

El tiempo invertido en convertir un dato digital en una tensión analógica será igual al periodo de la señal ***SYNC***, que es igual al de ***DATO_OK.*** Para que el diseño sea más sencillo se va a establecer para *SCLK* una frecuencia constante, fijando sus tiempos de nivel bajo (t_L) y alto (t_H) a 20 ns, valor mínimo que se puede conseguir, cumpliendo las especificaciones de la Tabla 1 con una señal ***CLK*** de frecuencia 100 MHz. El tiempo t_{SYNC} vendrá determinado por la frecuencia de la señal que se quiere obtener. Teniendo en cuenta que el valor mínimo de t_{SYNC} es de 20 ns, el tiempo mínimo para digitalizar un dato será:

$$TD_{min} = 16(TL_{min} + TH_{min}) + TSYNC_{min} = 16(20\ ns + 20\ ns) + 20\ ns = 660\ ns$$

Considerando que un periodo de una señal está formado por 256 muestras, la mayor frecuencia que se puede conseguir para las señales Vo1 y Vo2 será:

$$f_{max} = \frac{1}{256 * 660\ ns} = 5.92\ KHz$$

6. Memoria dual port.

La memoria dual por será la encargada de almacenar un periodo de cada una de las señales a generar. Esta memoria tendrá una organización de 256x 8 bits, teniendo un puerto de sólo escritura y otro se sólo lectura. Tanto los ciclos de escritura como los de lectura son síncronos, utilizando la

señal ***CLK*** para tal fin. Su declaración es la mostrada en la Figura 13, siendo la funcionalidad de cada uno de los puertos, la siguiente:

DIN. Bus de datos correspondiente al puerto de solo escritura.
ADDR_IN. Bus de dirección del puerto de sólo escritura.
WE. Señal de validación de la operación de escritura. Es activa a nivel alto.
DOUT. Bus de datos correspondientes al puerto de solo lectura.
ADDR_OUT. Bus de dirección del puerto de sólo lectura.

```
entity dpram_mem is
  port (
    DIN      : in  std_logic_vector(7 downto 0);
    ADDR_IN  : in  std_logic_vector(7 downto 0);
    WE       : in  std_logic;
    CLK      : in  std_logic;
    RST      : in  std_logic;
    ADDR_OUT : in  std_logic_vector(7 downto 0);
    DOUT     : out std_logic_vector(7 downto 0));
end entity;
```

Figura 13. Declaración de la entidad *dpram_mem*.

En la Figura 14 se muestra un cronograma de funcionamiento de la entidad ***dpram_men***.

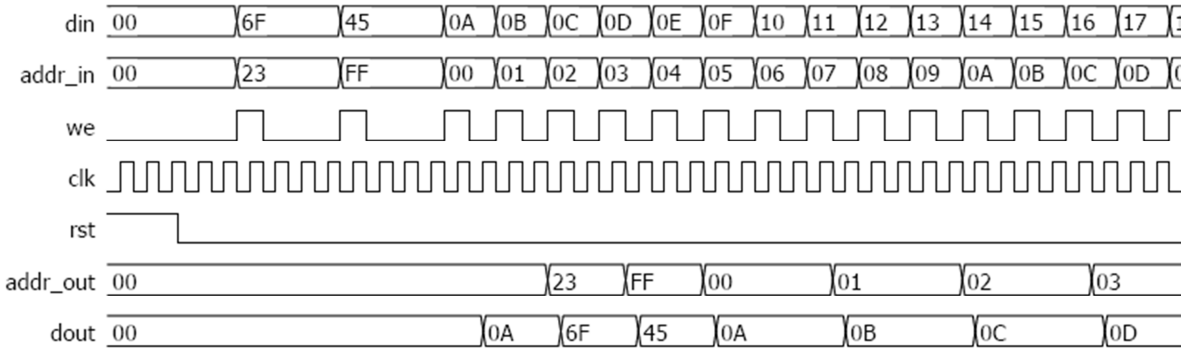

Figura 14. Ciclos de acceso al módulo dual port.

7. Controlador de la memoria dual port

Este módulo es el encargado de escribir en las memorias dual port del diseño la información que se envía a través del puerto paralelo (EPP). Dependiendo de la dirección seleccionada, el dato se almacenará en una u otra memoria. Hay que tener en cuenta que los datos que se envían desde el puerto EPP tienen un tamaño de 8 bits al igual que la memoria. Hay que tener en cuenta las siguientes consideraciones de diseño.

- Cada dato a almacenar en la memoria dual port se proporciona con un ciclo de escritura en la dirección correspondiente.
- En la memoria dual port 1 se escribirán los datos cuya dirección es $A1_{HEX}$ y en la dual port 2 los correspondientes a la dirección $A2_{HEX}$.
- A medida que se van enviando datos con la misma dirección, ser irán almacenando en posiciones consecutivas.

- Una vez almacenado un dato en la última posición de memoria (FF_{HEX}), si se envía datos manteniendo la misma dirección, estos se irán almacenado desde la primera posición (00_{HEX}) y en posiciones consecutivas.
- Cuando se realiza un ciclo de escritura en una dirección diferente al anterior ciclo se entiende que el dato se debe almacenar en la posición 0 de la memoria que corresponde con dicha dirección. Siempre y cuando esta dirección se corresponda con una de las dos utilizadas en este diseño: $A1_{HEX}$ y $A2_{HEX}$.
- Las memorias pueden ser escritas parcialmente

En la Figura 15 se muestra un cronograma de funcionamiento del bloque controlador de las memorias dual port.

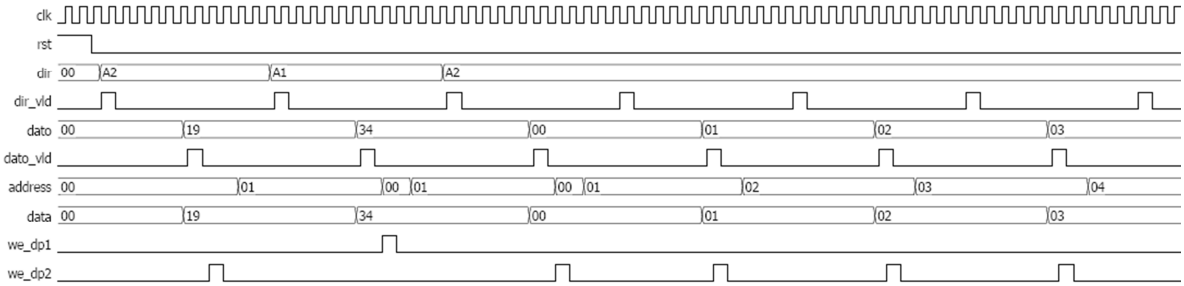

Figura 15. Cronograma de funcionamiento del bloque controlador de las memorias dual port.

En la Figura 16 se muestra la declaración de la entidad ***cnt_dpram*** que controla la gestión de las memorias dual port.

```
entity cnt_dpram is
  port (
    CLK      : in  std_logic;
    RST      : in  std_logic;
    DIR      : in   std_logic_vector (7 downto 0);
    DIR_VLD  : in   std_logic;
    DATO     : in   std_logic_vector (7 downto 0);
    DATO_VLD : in   std_logic;
    ADDRESS  : out  std_logic_vector(7 downto 0);
    DATA     : out  std_logic_vector(7 downto 0);
    WE_DP1   : out  std_logic;
    WE_DP2   : out  std_logic);
end cnt_dpram;
```

Figura 16. Declaración de la entidad cnt_dpram.

8. Generador de la dirección de las memorias dual port.

La entidad ***gen_dir*** se encarga de generar las direcciones de los puertos de sólo lectura de las memorias dual port. El ritmo al que cambian estas direcciones es proporcional al valor del dato seleccionado desde el puerto paralelo al escribir en la dirección $F0_{HEX}$ (Figura 3). Para modelar el funcionamiento de la entidad ***gen_dir*** se va a utilizar la estructura que se muestra en la Figura 17. En ella, *VALOR_FREC* se corresponde con el último dato escrito en la dirección $F0_{HEX}$.

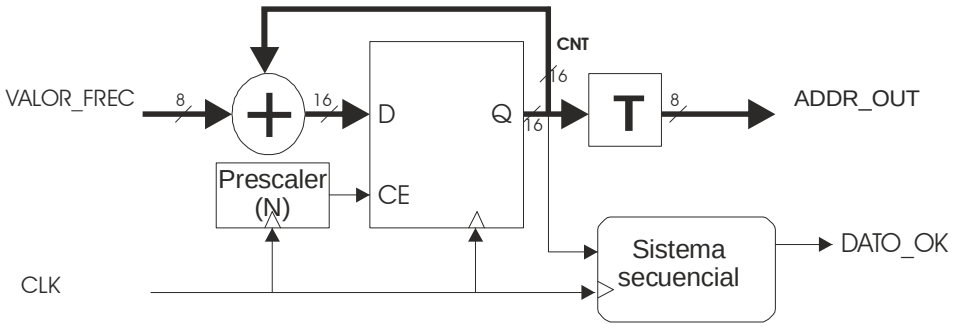

Figura 17 Diagrama de bloques de la entidad gen_dir.

El conjunto registro-sumador-truncador (**T**) modelan el generador de direcciones, en el que la frecuencia a las que estas se incrementan, viene dada por:

$$f \approx \frac{VALOR\_FREC}{N * 2^{16}} * f_{CLK}$$

Donde f_{CLK} representa la frecuencia de reloj de la señal de reloj (***CLK***) del diseño (100 MHz). El factor de división del prescaler (N) es igual al número mínimo de ciclos de reloj que se tarda en digitalizar un dato. Este valor, como se ha visto con anterioridad, sería 66: 64 para transmitir los 16 bits del dato y dos para llevar ***SYNC*** a nivel alto. Si bien se van a utilizar 4 pulsos de reloj para garantizar la temporización; así, se tendrá N=68.

Para informar al módulo controlador de los DACS de que hay un nuevo dato a digitalizar, la entidad *gen_dir* proporciona la salida ***DATA_OK***. Esta señal está retardada, con respecto a la dirección de la memoria dual port (***ADDR_OUT***) un periodo de ***CLK***, para compensar el tiempo de acceso de la memoria (Figura 18).

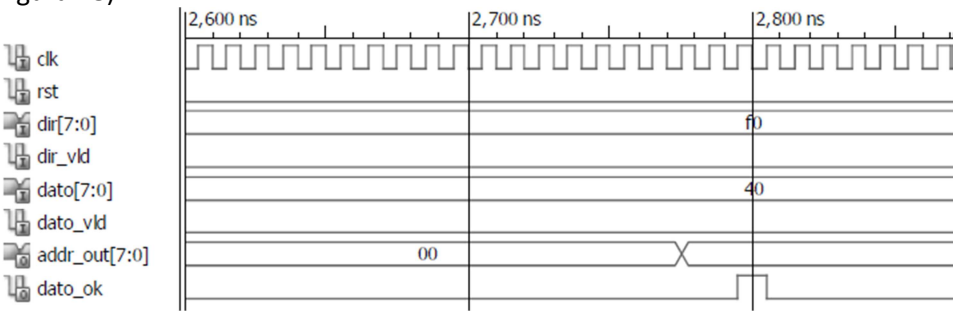

Figura 18. Retardo en la activación de DATO_OK.

En la se muestra la declaración de la entidad ***gen_dir***.

```
entity gen_dir is
  port (
    CLK        : in  std_logic;
    RST        : in  std_logic;
    DIR      : in   std_logic_vector (7 downto 0);
    DIR_VLD  : in   std_logic;
    DATO     : in   std_logic_vector (7 downto 0);
    DATO_VLD : in   std_logic;
    ADDR_OUT     : out std_logic_vector(7 downto 0);
    DATO_OK    : out std_logic);
end gen_dir;
```

Figura 19 Declaración de la entidad gen_dir.

9. Entidad *dcm*

Las FPGAs de la familia Spartan 6 dispone de 4 bloques para el control de todos los aspectos relacionados con la frecuencia, la fase y el *skew* de la red de relojes de la FPGA.

La FPGA recibe una señal de reloj externo de frecuencia 100 MHz. Sin embargo, con objeto de optimizar el funcionamiento del sistema y evitar al máximo problemas de forma y skew en la señal de reloj global del sistema, se propone el uso de un bloque DCM de los 4 disponibles. Por tanto, este elemento tendrá como entrada una señal (***RELOJ***) de 100 MHz y sacará una señal bien conformada de igual frecuencia (***CLK***).

Esta entidad se creará con la herramienta CORE Generator del sistema de desarrollo ISE y tendrá una declaración similar a la mostrada en la Figura 20.

```
entity dcm is
port
 (-- Clock in ports
  CLK_IN           : in     std_logic;
  -- Clock out ports
  CLK_OUT          : out    std_logic;
  -- Status and control signals
  RESET             : in     std_logic
 );
end dcm;
```

Figura 20 Declaración de la entidad *dcm*.

10. Información adicional.

Formando parte del conjunto de archivos que se entregan con la práctica se encuentran los archivos que contienen las entidades definidas en la Figura 3, excepto la entidad ***dcm***. Estos se deberán completar para obtener el funcionamiento deseado. Si bien, **bajo ningún concepto se podrán modificar ni el número ni el nombre de sus puertos.**

En el archivo ***gen_funciones.vhd***, en el que se modela la entidad ***gen_funciones***, se realizará la interconexión de los componentes de la Figura 3.

11. Consideraciones de diseño.

- A la hora de codificar el diseño se aconseja el uso de EMACS ya que su código, mayoritariamente, está libre de errores y ayuda a una mejor comprensión del mismo.
- No abusar en los niveles de anidación *if-else*. Su implementación es complicada y además infiere muchos recursos y penaliza de forma considerable la frecuencia máxima de reloj.
- Con idea de optimizar el diseño, es importante analizar los resultados de los informes de síntesis e implementación. Estos aportan muchas veces información sobre señales que no se usan, señales no conectadas, latches inferidos, relojes derivados, etc.
- No vale la excusa: “la simulación funcional va perfectamente”, ya que sentencias como

por ejemplo *wait for 10 ns* no tiene una traducción sintetizable. Por tanto para validar el diseño se recomienda sintetizar cada módulo por separado, realizar un test-bench de cada uno de ellos y realizar simulaciones funcionales y temporales de todo por separado.

- El diseño debe funcionar obligatoriamente a 100 MHz, sin embargo uno de los índices de mayor bondad de un diseño, además de obviamente gastar el menor número de recursos internos, es la máxima frecuencia de reloj. Esto se logra empleando mayoritariamente sistemas secuenciales los cuales son gobernados con una señale de reloj como máximo.

- El diseño hace uso de la señal del oscilador de la placa Atlys cuya frecuencia de 100 MHz.

- El nombre de todas las señales de reloj de los bloques secuenciales utilizados en el diseño deberá ser *CLK*.

- Todos los elementos secuenciales modelados deben ser activos en el flanco de subida de la señal *CLK*.

- Se debe verificar que en el diseño no se infieran latchs.

12. Desarrollo de la práctica.

Con el desarrollo de la práctica se pretenden abordar tanto aspectos de modelado en VHDL para síntesis como para simulación. Para ello se ha dividido en una serie de apartados de dificultad progresiva que llevan a la consecución del diseño final. Para cada uno de los apartados se indica la nota máxima que aporta (sobre un total de 10 puntos), pudiéndose modificar dicha nota a la baja en función de la consecución de los diferentes subapartados.

Es importante tener en cuenta que la memoria de la práctica deberá incluir no sólo los resultados de la misma sino también los pasos que se han llevado a cabo para su desarrollo. En este sentido, para cada uno de los apartados y subapartados se especifica claramente, qué detalles se deberán incluir en la memoria. De este modo, cada párrafo marcado con el símbolo (*) indica que se deberá generar una entrada en la memoria incluyendo en la misma lo que se pide. Asimismo, se deberá incluir en la memoria, la referencia que aparece después del símbolo anteriormente mencionado.

Apartado 1. Diseño del controlador del puerto paralelo (EPP). (1.5 puntos)

En este apartado el alumno deberá llevar a cabo la especificación en VHDL del módulo que realiza el control del bus EPP (entidad ***cnt_epp***) para su posterior simulación, síntesis, e implementación.

La consecución de este apartado deberá responder a las siguientes cuestiones:

1.1.- Comprobación de la funcionalidad del controlador del puerto paralelo. (1 punto)

El alumno deberá modelar y simular funcionalmente la entidad ***cnt_epp*** para comprobar que funciona correctamente creando el correspondiente test bench (***cnt_epp_tb.vhd***). Para poder simular esta entidad se proporciona la entidad ***epp_device*** que se encarga de modelar el funcionamiento de un puerto EPP, la cual deberá ser completada por los alumnos. Esta es una

entidad que se va a utilizar sólo en simulación con lo que se puede emplear cualquier construcción VHDL sin restricciones. La declaración de la entidad ***epp_device*** se muestra en la Figura 21. La Figura 22 muestra la conexión entre los distintos modelos dentro del banco de pruebas ***cnt_epp_tb.vhd***.

```
entity epp_device is
  port (
    DATA   : inout std_logic_vector(7 downto 0);
    PWRITE : out   std_logic;
    DSTRB  : out   std_logic;
    ASTRB  : out   std_logic;
    PWAIT  : in    std_logic);
end epp_device;
```

Figura 21. Declaración de la entidad epp_device.

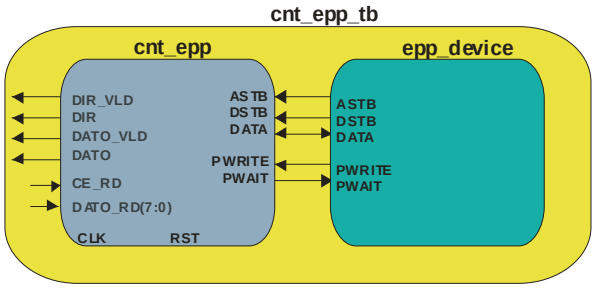

Figura 22. Conexiones a establecer en el test bench *cnt_epp_tb.*

En la entidad ***epp_device***, para modelar los ciclos de acceso (lectura/ escritura), se utiliza el procedure ***epp_cycle*** cuya declaración se muestra en la Figura 23. Donde ***address*** se corresponde con la dirección sobre la que se realiza el ciclo y ***data_io*** se corresponde con el dato a leer o escribir en dicho ciclo y con ***r_w*** el tipo de operación a realizar: ***r*** para la lectura y ***w*** para la escritura. En la Figura 24 se muestra un cronograma de funcionamiento del puerto EPP.

```
procedure epp_cycle ( address : in    std_logic_vector(7 downto 0);
                      data_io : inout std_logic_vector(7 downto 0);
                      r_w     : in    character) is
```

Figura 23. Declaración del procedure *epp_cycle* que modela los ciclos de acceso con el protocolo EPP.

En el archivo ***epp_device.vhd***, que se proporciona con este enunciado se han declarado las señales y constantes que se muestran en la Figura 24:

clk_epp. Señal de reloj interna que sincroniza la activación de las diferentes señales.
T_clk. Periodo de la señal ***clk_epp***.
N. Número de periodos ***clk_epp*** que están a nivel bajo las señales ***DSTRB*** y ***ASTRB***.
read_value. Se corresponde con el dato leído en un ciclo de lectura.
dir_frec. Dirección utilizada para seleccionar la frecuencia de las tensiones de salida.
dir_dpram1. Dirección asignada para los datos correspondientes a la tensión ***Vo1***.
dir_dpram2. Dirección asignada para los datos correspondientes a la tensión ***Vo2***.

```
signal   clk_epp        : std_logic := '0';     -- Internal clock signal
constant T_clk      : time      := 100 ns;  -- Internal clock period.    .
constant N          : natural := 10;    -- Length EPP cicle
signal read_value   : std_logic_vector(7 downto 0) := (others => '0');
constant dir_frec   : std_logic_vector( 7 downto 0) := x"F0";
constant dir_dpram1 : std_logic_vector( 7 downto 0) := x"A1";
constant dir_dpram2 : std_logic_vector( 7 downto 0) := x"A2";
```

Figura 24 señales y contantes declaradas en el archivo *epp_device.vhd*.

En la Figura 25 se muestra como se activan las diferentes señales del puerto EPP sincronizadas con la señal ***clk_epp*** para un ciclo de escritura. Para el de lectura son similares.

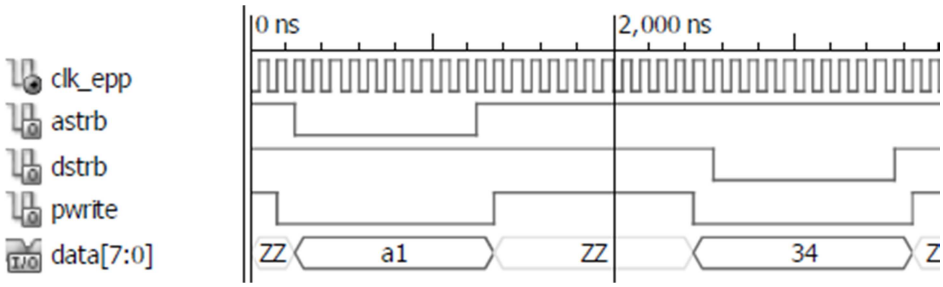

Figura 25. Temporización de un ciclo de escritura realizado por la entidad *epp_device*.

En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)1.1.1.- Código VHDL de la entidad ***cnt_epp*** y del banco de pruebas utilizado en su simulación. También se debe proporcionar el código de la entidad ***epp_device***.
(*)1.1.2- Razonamiento de por qué se ha adoptado la solución presentada.
(*)1.1.3.- Pantallazo de simulación donde se refleje que la entidad ***cnt_epp*** funciona correctamente. Para ello se realizarán tres ciclos de escritura, uno en cada uno de las direcciones que se van a utilizar en el diseño (*dir_frec*, *dir_dpram1* y *dir_dpram2*) y una operación de lectura en la dirección a elegir por el alumno.

1.2.- Simulación temporal del controlador del puerto paralelo. (0.5 puntos).

Con el objetivo de comprobar las características temporales del controlador del puerto EPP, se debe llevar a cabo la síntesis y la implementación de la misma. El alumno debe tener en cuenta que la simulación temporal requiere más tiempo de simulación ya que se utiliza un modelo estructural del circuito junto con los retardos generados por las herramientas de implementación. En la memoria de la práctica se deberá añadir los siguientes datos:

(*)1.2.1.- Tabla donde se incluya los recursos utilizados para la implementación del controlador del puerto EPP. Los datos anteriores serán extraídos del informe de síntesis. La tabla 2 muestra un ejemplo.

Tabla 2. Recursos utilizados

Slice Logic Utilization	
Number of Slice Registers	0 out of 54,576 0%
Number of Slice LUTs	4 out of 27,288 1%
Number used as logic	4 out of 27,288 1%
Number using O6 output only	0
Number using O5 output only	0
Number using O5 and O6	4
Number used as ROM	0
Number used as Memory	0 out of 6,408 0%

(*)1.2.2.- Medida del retardo entre el flanco activo de la señal de reloj y la activación de los puertos DIR_VLD y DATO_VLD. Se debe mostrar el resultado a través de pantallazos de simulación utilizando, para ello, dos cursores.

(*)1.2.3.- Pantallazo de simulación donde se refleje el que la entidad cnt_epp funciona correctamente.

Apartado 2. Descarga en placa del controlador del puerto paralelo. (1 puntos)

En este apartado, el alumno deberá comprobar el funcionamiento del control del puerto EPP en la placa de pruebas. Para la consecución de este objetivo se deberá tener en cuenta los siguientes pasos:

2.1.- Verificación práctica de la funcionalidad del controlador del puerto paralelo. (0.5 puntos)

El alumno deberá especificar el modelo ***top_system1*** cuya entidad se muestra en la Figura 26 y su estructura se muestra en la Figura 27.

```
entity top_system1 is
  port(
    CLK     : in    std_logic;
    RST     : in    std_logic;
    ASTRB   : in    std_logic;
    DSTRB   : in    std_logic;
    DATA    : inout std_logic_vector(7 downto 0);
    PWRITE  : in    std_logic;
    PWAIT   : out   std_logic;
    SWITCHES_I : in  std_logic_vector(7 downto 0);
    PSH_BUTTON:in    std_logic;
    LEDS_O     : out std_logic_vector (7 downto 0));
end top_system1;
```

Figura 26. Declaración de la entidad top_system1.

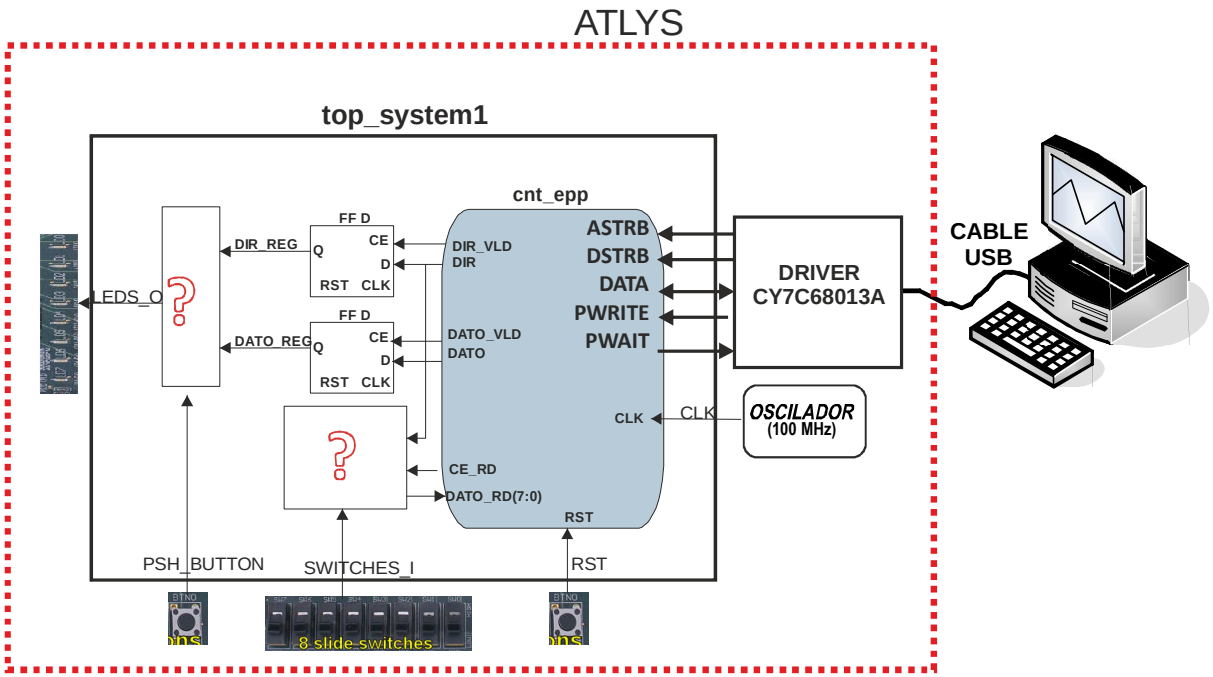

Figura 27. Sistema empleado para verificar el controlador del bus paralelo.

La arquitectura del *top_system1* incluye el componente ***cnt_epp*** y otros elementos adicionales que permitirán, a través del puerto EPP, leer el estado de los switches de la placa y sacar información por los leds en ella presentes.

En la Figura 27 se muestra como se utilizan sendos registros (FF D) para registrar la dirección y el dato escrito. Los dos bloques marcados con ***?***, deben ser diseñados por el alumno teniendo en cuenta que:

El bloque controlado con la señal *PSH_BUTTON* permite seleccionar que información se visualiza en los LEDS de forma que con un nivel bajo se selecciona la dirección registrada y con un alto el dato registrado.

El bloque al que tiene conectado los 8 switches de la placa (puerto *SWITCHES_I*) deberá llevarlos a la entrada ***DATO_RD*** del componente ***cnt_epp*** cuando se realice un ciclo de lectura en la posición 32_{HEX}.

Para comprobar que el modelo funciona se deberá llevar a cabo la descarga en placa. En los ficheros que se adjuntan a la práctica se incluye el fichero *top_system1.ucf* que contiene las restricciones asociadas a la asignación de los puertos de la entidad *top_system1* con los pines de la FPGA. La Figura 28 muestra el contenido del fichero *top_system1.ucf.*

```
NET "CLK"   LOC = "L15";
# onBoard USB controller
 NET "ASTRB"   LOC = "B9";
 NET "DSTRB"   LOC = "A9";
 NET "PWRITE"   LOC = "C15";
 NET "PWAIT"   LOC = "F13";
 NET "DATA<0>"  LOC = "A2";
 NET "DATA<1>"  LOC = "D6";
 NET "DATA<2>"  LOC = "C6";
 NET "DATA<3>"  LOC = "B3";
 NET "DATA<4>"  LOC = "A3";
 NET "DATA<5>"  LOC = "B4";
 NET "DATA<6>"  LOC = "A4";
 NET "DATA<7>"  LOC = "C5";

# onBoard Pushbuttons
 NET "PSH_BUTTON"  LOC = "F5";
 NET "RST"   LOC = "N4";

# onBoard LEDS
 NET "LEDS_O<0>" LOC = "U18";
 NET "LEDS_O<1>" LOC = "M14";
 NET "LEDS_O<2>" LOC = "N14";
 NET "LEDS_O<3>" LOC = "L14";
 NET "LEDS_O<4>" LOC = "M13";
 NET "LEDS_O<5>" LOC = "D4";
 NET "LEDS_O<6>" LOC = "P16";
 NET "LEDS_O<7>" LOC = "N12";

# onBoard SWITCHES
 NET "SWITCHES_I<0>" LOC = "A10";
 NET "SWITCHES_I<1>" LOC = "D14";
 NET "SWITCHES_I<2>" LOC = "C14";
 NET "SWITCHES_I<3>" LOC = "P15";
 NET "SWITCHES_I<4>" LOC = "P12";
 NET "SWITCHES_I<5>" LOC = "R5";
 NET "SWITCHES_I<6>" LOC = "T5";
 NET "SWITCHES_I<7>" LOC = "E4";
```

Figura 28. Contenido del fichero top_system1.ucf.

En la memoria de la práctica se deberá añadir los siguientes datos:

(*)2.1.1.- Especificación en VHDL de los modelos ***top_system1*** y el correspondiente banco de pruebas ***top_sytem1_tb***.
(*)2.1.2.- Pantallazos de la simulación funcional de los modelos anteriores donde se demuestre que funcionan correctamente.
(*)2.1.3.- Pantallazos de la simulación temporal de los modelos anteriores donde se demuestre que funcionan correctamente. En este caso se deberá añadir una tabla donde se reflejen los recursos utilizados en su implementación (ver Tabla 2).

2.2.- Descarga en placa de la entidad *top_system1*. (0.5 puntos)

En este punto el alumno deberá descargar en placa la entidad ***top_system1***. Los datos de entrada al diseño *top_system1* se suministrarán a través del entorno mostrado en la Figura 29.

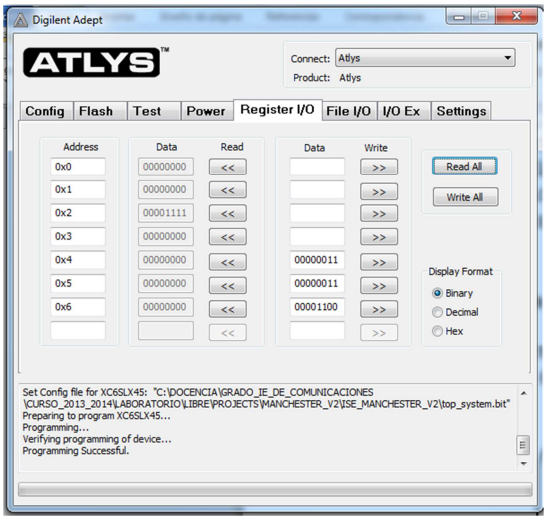

Figura 29.Entorno para leer y escribir datos al díselo top_sytem1.

En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)2.2.1.- Pantallazos como el de la Figura 29 donde se refleje que la entidad top_system1 correctamente para las distintas operaciones.

Para que este apartado sea considerado cómo válido, el alumno deberá mostrar al profesor que la descarga en placa funciona correctamente.

Apartado 3. Diseño del controlador de los DACs . (1 puntos)

En este apartado el alumno deberá especificar un módulo VHDL que implemente el sistema correspondiente al controlador de los DACs (entidad ***cnt_dac***).

Este apartado se estructura en los siguientes subapartados.

3.1.- Simulación funcional del controlador de los DACs. (0.75 puntos)

En este apartado se deberá realizar la simulación funcional de la entidad ***cnt_dac***, realizando el correspondiente test bench. Para ello, junto con el enunciado de esta práctica se proporciona en el archivo ***DAC121S101.vhd*** en el que se modela el funcionamiento de dicho DAC. Este componente deberá ser utilizado por duplicado, ya que la entidad ***cnt_dac*** realiza el control de los dos DAC121S101 presentes en la placa PmodDAC1 que se va a conectar a la tarjeta Atlys. Así el test bench tendrá la estructura mostrada en la Figura 30.

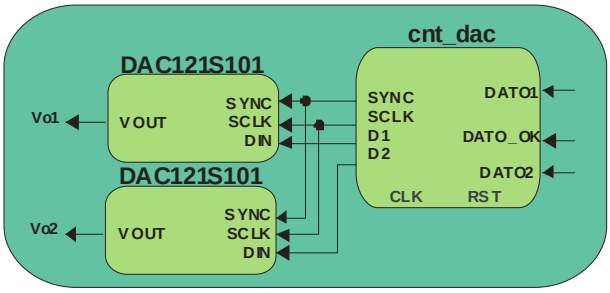

Figura 30 estructura del test bench utilizado para simular la entidad *cnt_dac.*

En la memoria de la práctica se deberá añadir los siguientes datos:

(*)3.1.1.- Código VHDL del módulo de control de los DACs y del banco de pruebas utilizado en su simulación.
(*)3.1.2- Razonamiento de por qué se ha adoptado la solución presentada.
(*)3.1.3.- Pantallazo de simulación funcional donde se refleje que le mencionado circuito funciona correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para comprobar que para todas las operaciones el módulo funciona correctamente.

3.2.- Simulación temporal del controlador de los DACs. (0.25 puntos)

En este punto se llevará a cabo la simulación temporal del modelo VHD creado. En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)3.2.1.- Tabla donde se incluyan los recursos utilizados para la implementación (ver tabla 2).
(*)3.2.2.- Pantallazo de simulación donde se refleje que el controlador de los DACs funciona correctamente.

Apartado 4. Modelado de la memoria dual port. (0.75 puntos)

En este apartado el alumno deberá analizar el módulo VHDL que implemente la memoria dual port (entidad ***dpram_mem***).

Este apartado se estructura en los siguientes subapartados.

4.1.- Simulación funcional la memoria dual port. (0.5 puntos)

En este apartado se deberá realizar la simulación funcional de la entidad ***dpram_mem*** para comprobar que funciona correctamente de forma individual. En la memoria de la práctica se deberá añadir los siguientes datos:

(*)4.1.1.- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.
(*)4.1.2- Estudio detallado del funcionamiento de la memoria dual port
(*)4.1.3.- Pantallazo de simulación donde se refleje que el mencionado circuito funciona correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para comprobar que el módulo funciona correctamente.

4.2.- Simulación temporal del módulo la memoria dual port. (0.25 puntos)

En este punto se llevará a cabo la simulación temporal de la entidad ***dpram_mem***. En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)4.2.1.- Tabla donde se incluyan los recursos utilizados para la implementación (ver tabla 2).
(*)4.2.2.- Pantallazo de simulación donde se refleje que el módulo funciona correctamente.

Apartado 5. Implementación del controlador de las memorias dual port (2.5 puntos)

En este apartado el alumno deberá especificar un módulo VHDL que implemente el controlador de las memorias dual port (entidad ***cnt_dpram***).

Este apartado se estructura en los siguientes subapartados.

5.1.- Simulación funcional del controlador de la memoria dual port. (1.5 punto)

En este apartado se deberá realizar la simulación funcional para comprobar que el mencionado módulo funciona correctamente de forma individual. En el banco de pruebas utilizado se deberá utilizar un procedimiento (*procedure*) para generar las señales ***DIR***, ***DIR_VLD***, ***DATO*** y ***DATO_VLD***.

En la memoria de la práctica se deberá añadir los siguientes datos:

(*)5.1.1.- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.
(*)5.1.2- Razonamiento de por qué se ha adoptado la solución presentada.
(*)5.1.3.- Pantallazo de simulación donde se refleje que el mencionado circuito funciona correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para comprobar que el módulo funciona correctamente.

5.2.- Simulación temporal del controlador de las memorias dual port. (0.5 puntos)

En este punto se llevará a cabo la simulación temporal del controlador de las memorias dual port. En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)5.2.1.- Tabla donde se incluyan los recursos utilizados para la implementación (ver tabla 2).

(*)5.2.2.- Pantallazo de simulación donde se refleje que el módulo de representación funciona correctamente.

Apartado 6. Implementación del generador de la dirección de las memorias dual port (0.75 puntos)

En este apartado el alumno deberá especificar un módulo VHDL que implemente la entidad ***gen_dir***. Este apartado se estructura en los siguientes subapartados.

6.1.- Simulación funcional del generador de la dirección de las memorias dual port. (0.5 punto)

En este apartado se deberá realizar la simulación funcional para comprobar que el mencionado módulo funciona correctamente de forma individual. En el banco de pruebas utilizado deberá utilizar el procedimiento (*procedure*) utilizado en la simulación del controlador de la memoria dual port.

En la memoria de la práctica se deberá añadir los siguientes datos:

(*)6.1.1.- Código VHDL del módulo y del banco de pruebas utilizado en la simulación.
(*)6.1.2- Razonamiento de por qué se ha adoptado la solución presentada.
(*)6.1.3.- Pantallazo de simulación donde se refleje que el mencionado circuito funciona correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para comprobar que el módulo funciona correctamente.

6.2.- Simulación temporal del generador de la dirección de las memorias dual port. (0.25 puntos)

En este punto se llevará a cabo la simulación temporal del generador de la dirección de las memorias dual port. En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)6.2.1.- Tabla donde se incluyan los recursos utilizados para la implementación (ver tabla 2).
(*)6.2.2.- Pantallazo de simulación donde se refleje que el módulo de representación funciona correctamente.

Apartado 7. Implementación del diseño completo (2.5 puntos)

En este apartado el alumno deberá especificar el módulo VHDL que implemente el diseño completo de esta práctica (entidad ***gen_funciones***).

Este apartado se estructura en los siguientes subapartados.

7.1.- Simulación funcional de la entidad *gen_funciones*. (1 punto)

En este apartado se deberá realizar la simulación funcional para comprobar que el diseño final funciona correctamente. Para esta simulación se proporcionan sendos archivos de datos ***Vo1.dat*** y ***Vo2.dat*** que contienen un periodo completo de dos señales en formato hexadecimal de 4 dígitos. Estos archivos deberán ser leídos por el módulo de simulación del puerto EPP (entidad *epp_device1*) para ser almacenados en la memoria dual port. La entidad ***epp_device1*** se debe crear a partir de la entidad ***cnt_epp***.

En la memoria de la práctica se deberá añadir los siguientes datos:

(*)7.1.1.- Código VHDL del banco de pruebas utilizado para su simulación. La entidad ***gen_funciones*** se proporciona con el enunciado.
(*)7.1.2.- Pantallazo de simulación donde se refleje que el mencionado circuito funciona correctamente. Para ello en el banco de pruebas deberá incluir los estímulos necesarios para comprobar que el módulo funciona correctamente.

7.2.- Simulación temporal de la entidad *gen_funciones*. (0.5 puntos)

En este punto se llevará a cabo la simulación temporal de todo el diseño. En la memoria de la práctica se deberá añadir los siguientes puntos:

(*)7.2.1.- Tabla donde se incluyan los recursos utilizados para la implementación (ver tabla 2).
(*)7.2.2.- Pantallazo de simulación donde se refleje que el sistema funciona correctamente.

7.3.- Descarga en placa del modelo completo. (1 puntos)

Para comprobar que el modelo funciona se deberá llevar a cabo la descarga en placa. En los ficheros que se adjuntan a la práctica se incluye el fichero ***gen_funciones.ucf*** que contiene las restricciones asociadas a la asignación de los puertos de la entidad ***gen_funciones*** con los pines de la FPGA. La Figura 31 muestra el contenido del fichero ***gen_funciones.ucf.***

```
NET "RELOJ"   LOC = "L15";

# onBoard USB controller
 NET "ASTRB"   LOC = "B9";
 NET "DSTRB"   LOC = "A9";
 NET "PWRITE"   LOC = "C15";
 NET "PWAIT"   LOC = "F13";
 NET "DATA<0>"  LOC = "A2";
 NET "DATA<1>"  LOC = "D6";
 NET "DATA<2>"  LOC = "C6";
 NET "DATA<3>"  LOC = "B3";
 NET "DATA<4>"  LOC = "A3";
 NET "DATA<5>"  LOC = "B4";
 NET "DATA<6>"  LOC = "A4";
 NET "DATA<7>"  LOC = "C5";

# onBoard Pushbuttons
 NET "RST"   LOC = "N4";

# DAC
NET "SYNC"  LOC = "T3"   ;
NET "D1"  LOC = "R3"   ;
NET "D2"  LOC = "P6"   ;
NET "SCLK"  LOC = "N5"   ;
```

Figura 31. Contenido del fichero gen_funciones.ucf.

Para que este apartado sea considerado cómo válido, el alumno deberá mostrar al profesor que la descarga en placa funciona correctamente.

13. Documentación a entregar.

La documentación a entregar, para cada uno de los diseños, debe incluir la siguiente información:

- CD conteniendo **única y exclusivamente** los archivos VHDL que modelan y simulan el diseño, sus test-bench y scripts. También se entregará el archivo de configuración: ***gen_funciones.bit***

- La memoria impresa del diseño.

- Listado, en papel, de todos los archivos VHDL. Cada entidad se debe imprimir en hojas independientes. Una línea de código deberá corresponder con una única línea impresa.

La falta de alguno de los requisitos anteriores (consideraciones de diseño, especificaciones para la simulación y documentación a entregar) conlleva a una calificación de **No Apta**.

14. Evaluación.

A la hora de evaluar esta práctica, la cual tiene un peso muy significativo en la calificación final del laboratorio, la nota final vendrá combinada por el valor de la memoria entregada a cada profesor correspondiente, así como por la presentación/examen de dicha práctica.

La memoria de la práctica deberá ser enviada, por correo electrónico, al profesor que imparte el grupo al que pertenece el alumno antes del día 29 de abril de 2015.

La presentación y defensa de la práctica se realizará de forma individual le día 4 de mayo de 2015 a las 10:00.

Cada grupo deberá presentar la documentación exigida en el punto 13.

El no cumplimiento de las fechas establecidas supondrá el suspenso de la práctica.