Universidad de Alcalá

Práctica libre: Generador de señal controlado desde un puerto serie RS232

ASIGNATURA

Modelado de Sistemas Computacionales

Grado en Ingeniería de Computadores

Universidad de Alcalá

Curso Académico 2022/2023

Curso 3º – Cuatrimestre 2º

Dpto. Electrónica

Contenido

1.- Introducción.

La práctica que se describe en este documento tiene como objetivo el desarrollo en VHDL de un generador de funciones que permite cubrir todas las fases del flujo de diseño de sistemas electrónicos para configurar FPGAs u otro tipo de ASICs, a saber: modelado, simulación, síntesis, implementación y configuración del dispositivo.

Para la implementación de esta práctica se utilizará la tarjeta de pruebas del laboratorio *Basys 3* de *Digilent* (https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual) basada en una *FPGA Artix-7* (*XC7A35T-ICPG236C*) de la firma comercial *Xilinx* que incorpora distintos periféricos de entrada/salida. Entre ellos caben destacar varios conectores para unir la FPGA con el exterior (*Pmod ports*), que se van a utilizar para conectar un conversor digital analógico (DAC) (https://digilent.com/reference/_media/reference/pmod/pmodda2/pmodda2_rm.pdf), también de *Digilent*.

En general, esta 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 conceptualización del diseño en hardware en base a circuitos combinaciones y/o secuenciales, la búsqueda de los circuitos digitales que satisfagan las condiciones de funcionamiento, la sincronización entre los distintos circuitos, el diseño y especificación de máquinas de estado (FSMs) así como la instanciación de componentes diseñados previamente que pone de manifiesto la importancia de la reutilización en el mundo hardware.

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

2.- Descripción.

El diseño a realizar consiste en el modelado en VHDL de un sistema digital que permite, una vez programado en una FPGA, generar una señal analógica ***VOUT*** (figura 1) cuya forma de onda, amplitud y frecuencia se van a configurar a través del puerto serie de un ordenador utilizando el protocolo RS232. Asimismo, en los displays de siete segmentos de la placa Basys3 se representará la frecuencia de la señal generada.

Con los datos de configuración de la forma de onda enviados se generará una señal digital con las características especificadas que, a través de un conversor digital analógico *DAC121S101* de *National Semiconductor* presente en el módulo periférico PmodDA2 de *Digilent*, se transformará en una señal analógica. Para introducir los datos en el DAC se emplea un protocolo serie síncrono que utiliza tres líneas: reloj (***SCLK***), dato de entrada (***DIN***) y sincronismo (***SYNC***).

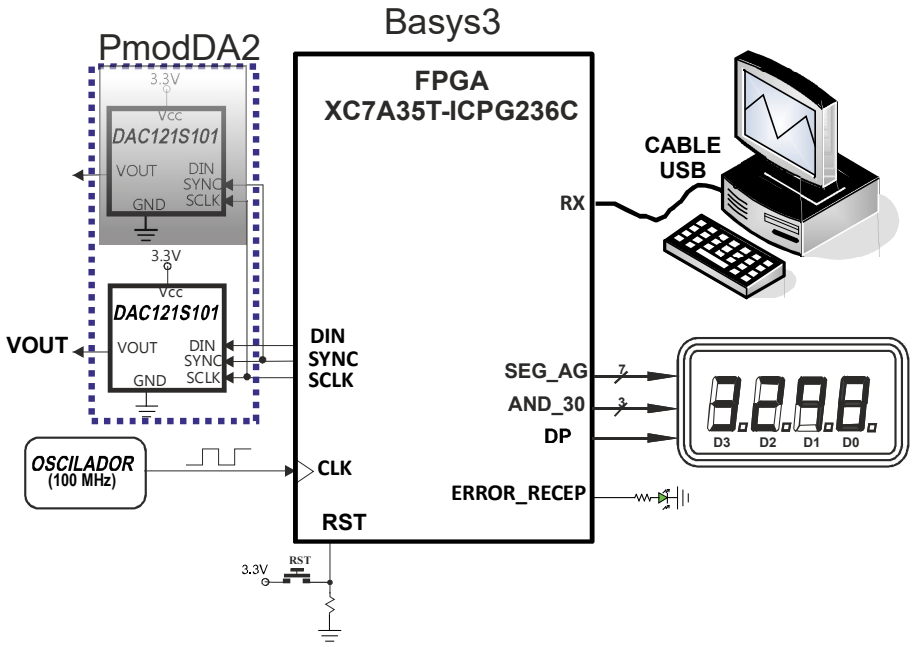

Figura 1. Conexión del hardware a utilizar.

El sistema a diseñar hace uso de una señal de reloj (***CLK***) de frecuencia igual a 100 MHz (periodo T=10 ns), que procede de un oscilador incluido en la placa *Basys 3*. Esta señal se utilizará como señal de sincronismo (reloj) de todos los módulos secuenciales del diseño. Por otro lado, la entrada ***RST*** funciona como señal ***ASÍNCRONA*** de inicialización **activa a nivel alto** **para llevar todos los bloques secuenciales** a su estado inicial. Además, estos bloques **no podrán utilizar ninguna otra señal como entrada asíncrona o de reloj**.

3.- Estructuración del diseño.

Con el fin de facilitar la tarea, para el modelado del sistema se va a realizar un diseño jerárquico en el que la entidad de mayor nivel se llama ***top_system***, entidad que incluye a su vez cinco módulos (Figura 2). En la documentación de la práctica se proporcionan "los esqueletos" de los archivos VHDL que se van a utilizar para modelar estos módulos. En este sentido, el alumno deberá completar los módulos VHDL con el código necesario para que realicen las funciones que posteriormente se detallarán, teniendo en cuenta que **no se podrán modificar los puertos de cada una de las entidades** que se proporcionan y que el diseño de cada uno de los módulos anteriores no podrá ser jerárquico. En otras palabras, cada uno de los cinco módulos que componen el diseño no podrá contener otros componentes, a excepción del módulo *signal_generator*. De igual manera, los nombres de las señales a utilizar para conectar los diferentes componentes de la entidad ***top_system*** deberán ser las que se muestran en la Figura 2. El incumplimiento de las restricciones anteriores conlleva a una calificación de **No apto en la práctica.**

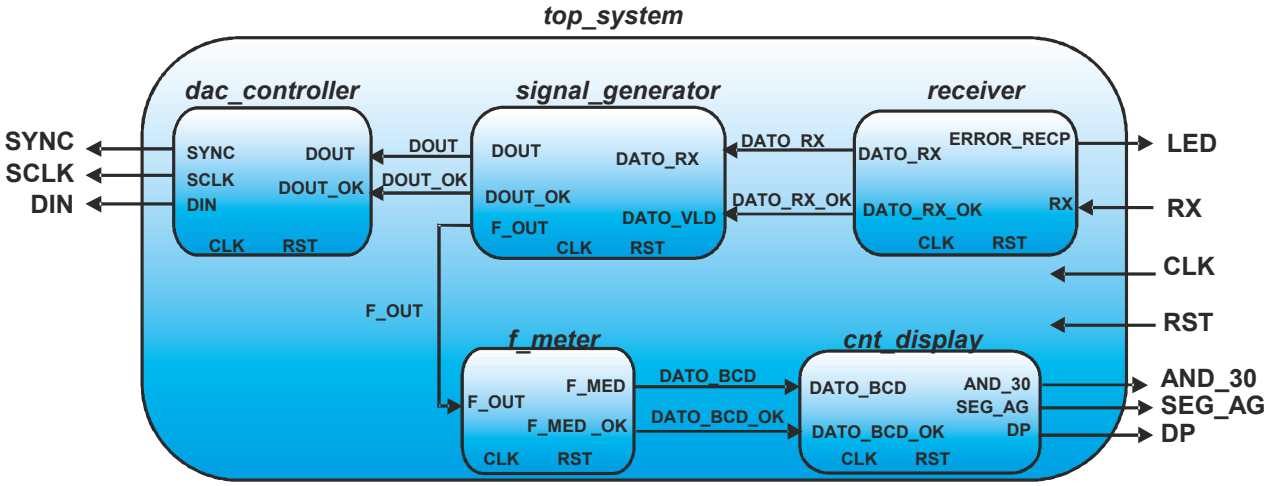

Figura 2. Estructura del diseño a realizar.

La declaración de la entidad ***top_system*** se muestra en el Listado 1.

```
entity top_system is
  port(RST     : in  std_logic;
       CLK     : in  std_logic;
       --SERIAL PORT
       RX      : in  std_logic;
       LED     : out std_logic;
       -- DAC
       SYNC    : out std_logic;
       SCLK    : out std_logic;
       D_OUT   : out std_logic;
       --display
       DP      : out std_logic;
       SEG_AG  : out std_logic_vector(6 downto 0);  -- gfedcba
       AND_30  : out std_logic_vector(3 downto 0));
end top_system;
```

Listado 1. Declaración de la entidad top_system.

Los puertos de la entidad ***top_system*** se describen a continuación:

CLK: entrada de sincronismo para los elementos secuenciales. Tiene un periodo T=10 ns, frecuencia de 100 MHz.

RST: entrada activa a nivel alto, procedente de un pulsador de la placa de pruebas que se utiliza para inicializar, de forma asíncrona, los elementos secuenciales del diseño.

RX: entrada de datos proveniente de un ordenador y que utiliza el protocolo RS232.

LED: salida activa a nivel alto que indica que se ha producido un error en la recepción del dato serie.

SYNC : salida que indica que se está enviando un nuevo dato al DAC.

SCLK: salida de sincronismo utilizada por el DAC para almacenar los bits del dato a convertir.

DIN: salida serie de los datos que se envían al DAC y que se transformarán en una señal analógica.

AND_30: salidas activas a nivel bajo que se utilizan para activar los displays a través de sus ánodos. Hay que tener en cuenta que los displays son de ánodo común y los ánodos se excitan a través de transistores bipolares PNP lo que hace que los niveles de activación de los ánodos sean los contrarios: se activan con un nivel bajo.

DP: salida correspondiente al punto de los displays.

SEG_AG: salidas correspondientes a los segmentos (a, b, ... g) de los *displays* de la placa.

A continuación, se describe brevemente la funcionalidad de cada una de las entidades/módulos de la Figura 2:

Cnt_display: entidad que permite visualizar en los 4 *displays* de 7 segmentos de la placa *Basys 3* el valor de la frecuencia de la señal generada.
F_meter: módulo encargado de medir la frecuencia de la señal generada, frecuencia que se proporciona en formato BCD por la salida ***F_MED*** y validada con ***F_MED_OK***.
DAC_controller: entidad que se encarga de generar las señales de control necesarias para enviar un dato de 12 bits al DAC en formato serie utilizando el protocolo SPI.
Signal_generator: entidad que se encarga de generar los datos a enviar al DAC. Este módulo proporciona la salida ***DATO_OK*** que se utiliza como sincronismo, de forma que pasa a nivel alto, durante un periodo de la señal de ***CLK***, para indicar que hay un nuevo dato a enviar al DAC. Además, el módulo proporciona una señal periódica ***F_OUT*** que tiene una frecuencia igual a la de la señal generada y está a nivel alto un periodo de la señal ***CLK***. El módulo *f_meter* utiliza la señal ***F_OUT*** para obtener la frecuencia de la señal generada.
Receiver: entidad que se encarga de recibir el dato enviado desde el PC utilizando el protocolo RS232 y proporcionar un dato paralelo de 8 bits, que se valida activando (a nivel alto) durante un periodo de ***CLK*** la señal ***DATO_RX_OK***.

En las siguientes secciones de esta memoria se describe de una forma más detallada el funcionamiento de las entidades arriba mencionadas. Aunque la funcionalidad de alguna de las entidades se proporcionará al alumno en formato encriptado, es esencial que se entienda el funcionamiento conjunto e individual de todas las entidades.

4.- Entidad *cnt_display*.

Para realizar la visualización del dato de frecuencia, codificado en BCD de 4 dígitos, obtenido por la entidad *f_meter,* se van a utilizar los cuatro *displays* que hay en la placa *Basys3*. Estos *displays* son de ánodo común y comparten las líneas asociadas a sus segmentos y punto como se puede apreciar en la Figura 3.

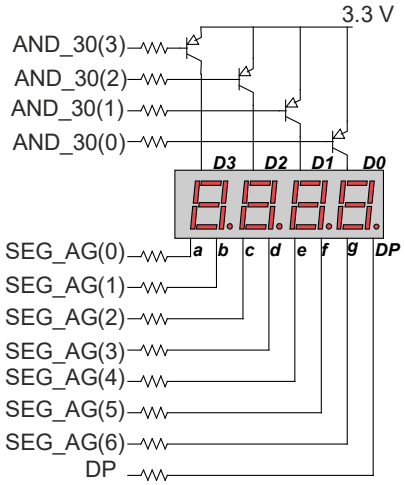

Figura 3. Conexionado al módulo de visualización

El módulo a diseñar deberá realizar la activación secuencial de cada uno de los ánodos de los *displays* ***D3***, ***D2***, ***D1*** y ***D0***, mientras que en las líneas ***SEG_AG*** se pone el valor necesario para poder visualizar un dato en dicho *display* (Figura 4). En la visualización no se deben representar los ceros a la

izquierda. En el caso de que el valor del dato BCD sea 0000, se visualizará un 0 en el display ***D0***. Así mismo, cuando alguno de los dígitos del puerto ***DATO_BCD*** a representar en los displays no se corresponda con un valor BCD, el sistema deberá detectarlo representando la palabra FAIL (Figura 5).

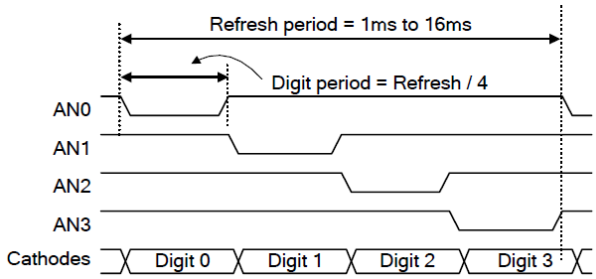

Figura 4. Secuencia de activación de los ánodos de los displays.

Figura 5. Aspecto de la visualización en los displays cuando hay un dato erróneo en DATO_BCD.

Un aspecto importante a tener en cuenta es que el periodo de refresco (*Refresh period*) (Figura 4), en milisegundos (ms), coincidirá con el número del puesto del laboratorio que ocupa el alumno/los alumnos. **El valor de este factor de división se deberá asignar a una constante de nombre *CTE_DISP*.** Para realizar las simulaciones de este apartado y con el objetivo de reducir el tiempo de simulación, se optará por el mismo valor, pero expresado en microsegundos (µs).

El control del sistema de visualización se realiza en la entidad ***cnt_display***, cuya declaración se muestra en el Listado 2.

```
entity cnt_display is
  port (
    CLK         : in  std_logic;
    RST         : in  std_logic;
    DATO_BCD    : in  std_logic_vector(15 downto 0);
    DATO_BCD_OK : in  std_logic;
    AND_30      : out std_logic_vector(3 downto 0);
    DP          : out std_logic;
    SEG_AG      : out std_logic_vector(6 downto 0));
end cnt_display;
```

Listado 2. Declaración de la entidad cnt_display

Finalmente, hay que considerar que en el puerto ***SEG_AG***, el bit de mayor peso se corresponde con el segmento ***g*** del display (Figura 3).

Como no se van a utilizar los puntos de los displays estos deberán permanecer siempre apagados.

5.- Entidad *f_meter*.

La entidad ***f_meter*** es la encargada de obtener el valor de la frecuencia de las señales de salida generadas. La declaración de esta entidad se muestra en el Listado 3.

```
entity f_meter is
  port(
    CLK       : in  std_logic;
    RST       : in  std_logic;
    F_OUT     : in  std_logic;
    F_MED_OK  : out std_logic;
    F_MED     : out std_logic_vector(15 downto 0));
end f_meter;
```

Listado 3. Declaración de la entidad f_meter.

La frecuencia de una señal *Vout* (f_{Vout}) se puede definir como el número de repeticiones por unidad de tiempo de cualquier fenómeno o suceso periódico. En consecuencia, su valor vendrá dado por el número de veces que pasa por un determinado valor durante un intervalo de tiempo y dividido por la duración de este último, al que se conoce con el nombre de tiempo de puerta (T_{PTA}).

Dada una señal digital que durante un tiempo T_{PTA} presenta N (Figura 6) flancos de subida (o bajada), su periodo (T_{Vout}) y frecuencia (f_{Vout}) serán igual a:

$$T_{Vout} = \frac{T_{PTA}}{N} \quad \Rightarrow f_{Vout} = \frac{N}{T_{PTA}} = N \cdot f_{PTA}$$

Si en la ecuación anterior el tiempo de puerta es igual a 1 milisegundo la frecuencia de *Vout* (f_{Vout}) tomará el valor de N expresada en kHz; pero si T_{PTA} es 1 segundo la frecuencia será de N expresada en Hz. Debido al desfase entre las señales *Vi* y S_{PTA}, señal esta última que incluye el pulso T_{PTA} (Figura 8), el error de medida puede ser de ± 1 f_{PTA}. Así, para que la medida sea lo más exacta posible, el tiempo de puerta se elige de forma que para el valor más pequeño de la frecuencia que se quiera medir, N sea 10 veces mayor a ese valor. A pesar de esto último, en esta práctica se va a trabajar con un tiempo de puerta igual a 1 segundo, siendo conscientes de que medidas inferiores a 10 Hz pueden ser poco precisas debido al error que se introduce. Para depurar el código y para que las simulaciones duren poco tiempo, se deberá tomar, para ellas, un tiempo de puerta igual a **1 milisegundo**, de tal forma que la frecuencia vendrá dada en kHz. Si bien, para la implementación se tomará en valor de **1 segundo**. El valor de este tiempo está controlado por una constante de nombre ***T_PUERTA.***

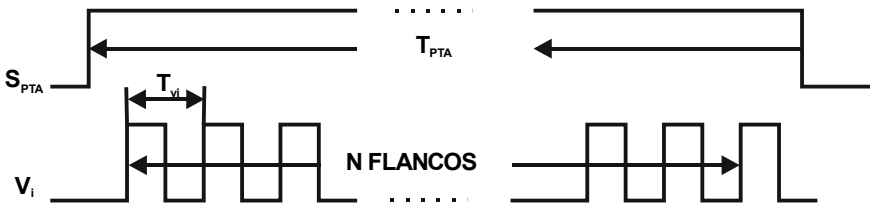

Figura 6. Cronograma utilizado para medir la frecuencia de una señal

Tal y como se ha diseñado la entidad ***signal_generator*** (ver apartado 7), la frecuencia de las señales de salida es la misma que la del puerto ***F_OUT***. Los valores de la frecuencia de ***F_OUT*** estarán comprendidos entre 6 kHz y 90 Hz, aproximadamente.

6.- Entidad *signal_generator*.

Este módulo se encarga de decodificar el dato recibido por el puerto serie (***DATO_RX***) y generar los datos a enviar al DAC. El dato recibido por el puerto serie se decodifica de acuerdo con la Tabla 1 para seleccionar los diferentes parámetros de la señal analógica de salida ***VOUT***. En el Listado 4 se muestra la declaración de la entidad *signal_generator*.

```
entity signal_generator is
  port ( RST         : in  std_logic;
         CLK         : in  std_logic;
         DATO_RX     : in  std_logic_vector (7 downto 0);
         DATO_RX_OK  : in  std_logic;
         DOUT        : out std_logic_vector (11 downto 0);
         F_OUT       : out std_logic;
         DOUT_OK     : out std_logic);
end signal_generator;
```

Listado 4. Declaración de la entidad signal_generator.

Tabla 1. Funcionalidad de los bits del dato recibido.

	Bit de DATO_RX								
	B_7	**B_6**	**B_5**	**B_4**	**B_3**	**B_2**	**B_1**	**B_0**	**Selección**
VOUT	0	0	X	X	X	1	1	0	Señal sinusoidal
	0	0	X	X	X	1	0	1	Señal triangular
	0	0	X	X	X	1	0	0	Diente de sierra
	0	0	X	X	X	0	1	1	Señal cuadrada D=50%
	0	0	X	X	X	0	1	0	Señal cuadrada D=25%
	0	0	X	X	X	0	0	1	Señal cuadrada D=75%
	0	1	F_5	F_4	F_3	F_2	F_1	F_0	Frecuencia de la señal
	1	0	X	X	G_3	G_2	G_1	G_0	Ganancia
	1	1	DC_5	DC_4	DC_3	DC_2	DC_1	DC_0	Nivel de continua (DC)

En la Figura 7 se muestra una posible estructura del módulo *signal_generator*. Dentro de este módulo, el conjunto formado por el prescaler, el registro, el sumador y el truncador (**T**) modelan un generador de direcciones de 8 bits (*address*) cuya frecuencia va a ser proporcional al valor de ***F*** (ver Tabla 1). Estas direcciones se van a utilizar para generar el tipo de señal deseada: seno, cuadrada, triangular y diente de sierra, todas ellas con un tamaño de 8 bits y codificadas en binario natural. Para generar la señal seno se va a utilizar una memoria ROM la cual tiene almacenadas las muestras de un periodo de un seno y se va a direccionar con la salida del bloque truncador (*address*). Esta memoria se encuentra modelada en el archivo *seno.vhd*, el cual se proporciona junto con el enunciado de esta práctica. Para obtener los otros tipos de señales (cuadrada, triangular y diente de sierra) se utiliza directamente la salida *address*.

La frecuencia de la señal obtenida viene dada por:

$$f_{VOUT} \approx \frac{F}{2^{20}} * f_{CLK}$$

Donde f_{CLK} representa la frecuencia de la señal de reloj (***CLK***) del diseño (100 MHz), ***F*** el valor de la frecuencia de la señal a generar (ver Tabla 1).

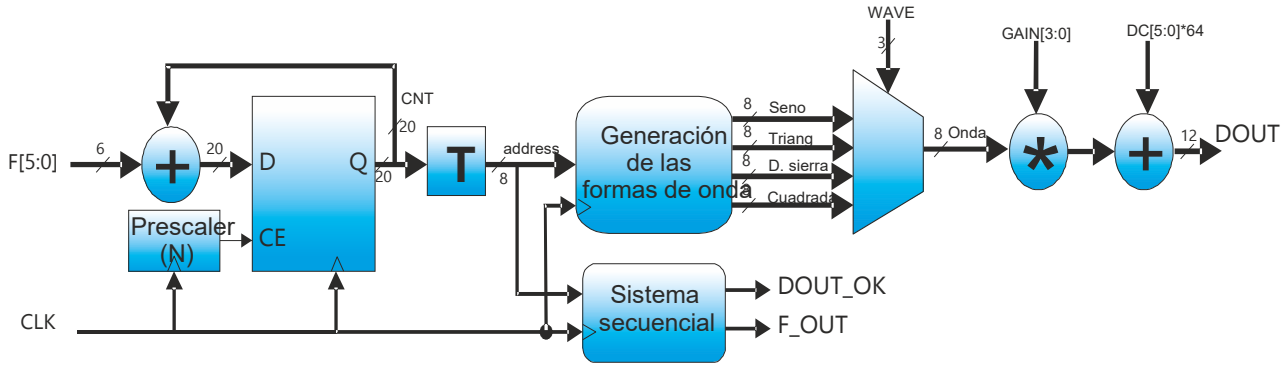

Figura 7. Diagrama de bloques del módulo signal_generator.

El valor del dato de salida ***DOUT*** (Figura 7) viene dado por :

$$DOUT \quad = Onda * GAIN * (DC * 64)$$

Donde ***Onda*** se corresponde con un dato de 8 bits y *GAIN* y **DC** representan los valores de la ganancia y nivel de continua, respectivamente, cuya definición viene dada en la Tabla 1.

Para informar al módulo que controla el DAC de que hay un nuevo dato a digitalizar, la entidad *signal_generator* proporciona la salida ***D_OK***, la cual se activa a nivel alto y durante un periodo de ***CLK*** cada vez que hay un nuevo dato en la salida ***DOUT***.

Asimismo, la entidad *signal_generator* también proporciona la salida ***F_OUT*** que tiene una frecuencia igual a la frecuencia de la señal generada (f_{VOUT} en la ecuación anterior), pero que sólo estará a nivel alto durante un periodo de la señal ***CLK***. Como se ha comentado en apartados anteriores, la salida ***F_OUT*** será utilizada por la entidad *f_meter* para que se pueda obtener la frecuencia de la señal generada, para ser visualizada en los displays.

En la Tabla 2 se indican los valores por defecto de la señal generada.

Tabla 2. Valores por defecto de la señal generada.

Tipo de señal	Sinusoidal
Frecuencia	Un valor distinto de cero
Ganancia	3
Nivel de continua (DC)	0

7.- Entidad *dac_controller*.

La entidad ***dac_controller*** es la encargada de proporcionar las señales de control del *DAC121S101* de la placa *PmodDA2*, el cual va a generar la salida analógica ***VOUT*** a partir del dato digital (***DOUT***) proporcionado por el módulo ***signal_generator***.

El *DAC121S101* es un convertidor digital analógico de 12 bits con una interfaz 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. Tal y como se puede observar en el cronograma de la Figura 8, la transmisión de un dato se inicia con la puesta a nivel bajo de ***SYNC***. A continuación el DAC toma los datos coincidiendo con los flancos de bajada de ***SCLK***. La transmisión del dato comienza con el bit de mayor peso (DB15 en la Figura 6). En consecuencia, el módulo ***dac_controller***, con el fin de garantizar las temporizaciones de la Tabla

3, deberá proporcionar el dato ***DIN*** sincronizado con los flancos de subida de ***SCLK*** y la señal ***SYNC*** se debe poner a nivel bajo durante el tiempo que se envía el dato. Las señales mostradas en el cronograma de la Figura 8 deben cumplir los tiempos mostrados en la Tabla 3.

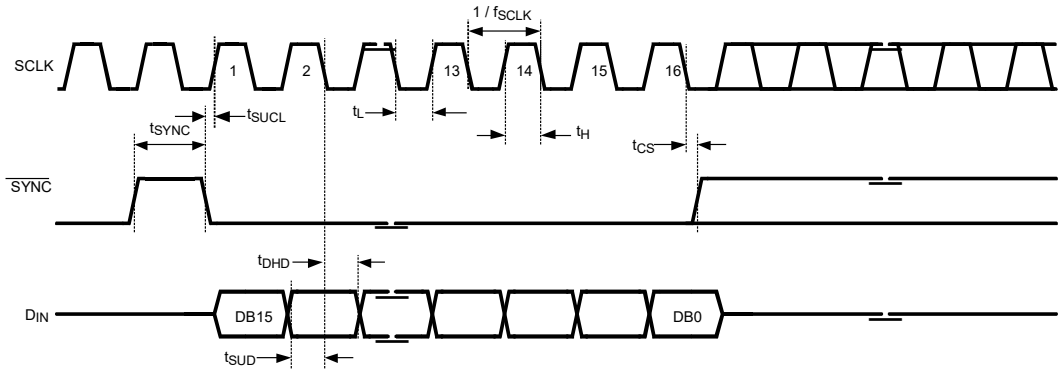

Figura 8. Cronograma de funcionamiento del DAC121S101.

Tabla 3. Valores límites de los tiempos de la Figura 8.

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	$C_L \leq$ 200 pF	8	10	µs (max)
			C_L = 500 pF	12		µs
		00Fh to FF0h code change, R_L = 2kΩ	$C_L \leq$ 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 $\overline{SYNC}$	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 8 el DAC recibe 16 bits: 12 del dato a convertir, más 4 (los de mayor peso) que se utilizan para el control interno (Figura 9). Para esta práctica estos bits de control tomarán el valor 0.

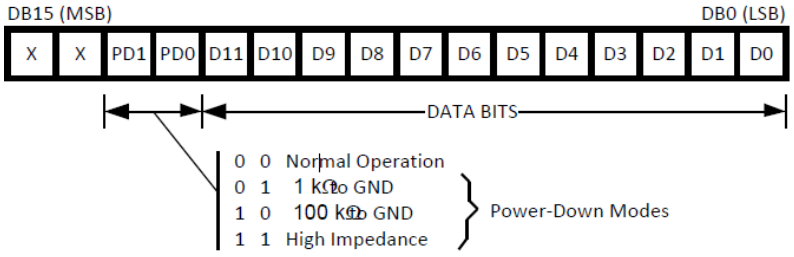

Figura 9. Formato de la trama del dato a enviar al DAC.

El DAC proporciona una tensión *V* comprendida en el rango de 0 V a 3,3 V, de acuerdo con la siguiente ecuación:

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

donde ***DATO*** se corresponde con el dato de 12 bits a convertir y está codificado en binario natural. Junto con este enunciado se proporciona el archivo ***DAC121S101.pdf*** donde se puede encontrar una descripción más detallada del funcionamiento del conversor.

La entidad ***dac_controller*** se debe diseñar de forma que cada vez que se activa, a nivel alto, ***DOUT_OK*** se inicia el proceso de transferencia del dato ***DOUT*** al DAC, generándose las señales ***SYNC***, ***SCLK*** y ***DIN***. Una vez que se han transferido todos los bits del dato a convertir ***SCLK*** y ***DIN*`* permanecerán a nivel bajo y ***SYNC*** a nivel alto. Estas señales se volverán a activar cuando haya un nuevo pulso en ***DOUT_OK***, indicándose que hay que enviar un nuevo dato al DAC. La declaración de la entidad ***dac_controller*** se muestra en el Listado 5.

```
entity dac_controller is
  port (
    CLK      : in  std_logic;
    RST      : in  std_logic;
    DOUT     : in  std_logic_vector(11 downto 0);
    DOUT_OK  : in  std_logic;
    SYNC     : out std_logic;
    SCLK     : out std_logic;
    DIN      : out std_logic);
end dac_controller;
```

Listado 5.Declaración de la entidad dac_controller.

El tiempo invertido en convertir un dato digital en una tensión analógica dependerá del valor de la frecuencia de ***SCLK.*** Para simplificar el diseño y cumplir los tiempos de la Tabla 3 se establece para ***SCLK*** un periodo de 40 ns (T_{LSCLK}= T_{HSCLK}= 20 ns). A este valor hay que añadir el tiempo mínimo que la señal ***SYNC*** debe estar a nivel alto (t_{SYNC}), el cual vendrá determinado por la frecuencia de la señal que se quiere obtener. Considerando el valor mínimo posible de t_{SYNC} en la Tabla 3, y para garantizar la temporización, se ha elegido 20 ns para este parámetro. Así, el tiempo mínimo, aproximado, para convertir un dato digital a analógico será:

$$TC_{min} = 16(TL_{SCLK} + TH_{SCLK}) + TSYNC_{min} = (16.40 + 20)\,ns = 660\,ns$$

Como se ha comentado con anterioridad, este es un valor aproximado que dependerá de la solución tomada para implementar el módulo ***dac_controller.***

Hay que indicar que mientras se esté enviando un dato al DAC, se ignorará toda activación del puerto ***DOUT_OK.***

Finalmente, junto con este enunciado se proporciona el archivo ***DAC121S101.vhd*** en el que se modela el funcionamiento de la recepción de datos por parte del DAC. Este archivo se debe utilizar en la **SIMULACIÓN** de las entidades ***dac_controller*** y ***top_system***. Se recomienda analizar su estructura ya que hay varias señales cuya observación en la simulación ayuda a verificar si el modelo especificado funciona correctamente a la hora de capturar los datos recibidos.

8.- Entidad *receiver*.

La entidad *receiver* (Listado 6), la cual **no hay que diseñar**, implementa el receptor del puerto serie (protocolo RS-232) proporcionando en su salida de 8 bits denominada ***DATO_RX,*** el dato transmitido, siempre y cuando este sea correcto, junto con una salida adicional, ***DATO_RX_OK***, que se pone a nivel alto durante un periodo de la señal ***CLK*** para indicar que se ha recibido un dato correcto. Por otro lado, la salida ***error_recep*** se pone a nivel alto cuando el dato recibido sea erróneo. Se considera que el dato recibido es erróneo cuando no sean correctos los bits de *start, stop* o *paridad*. Esta salida permanecerá a nivel alto hasta que se transmita un nuevo dato.

```
entity receiver is
  port (
    CLK          : in  std_logic;
    RST          : in  std_logic;
    RX           : in  std_logic;
    DATO_RX      : out std_logic_vector(7 downto 0);
    ERROR_RECEP  : out std_logic;
    DATO_RX_OK   : out std_logic);
end receiver;
```

Listado 6. Puertos del módulo/entidad receiver.

Como se ha comentado anteriormente, el módulo *receiver* no se deberá crear, ya que se proporciona una versión encriptada junto con este enunciado

El puerto serie es un dispositivo utilizado en los ordenadores y otros equipos para realizar una trasmisión de datos serie y asíncrona. En ella, la duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos. El protocolo de comunicaciones está estandarizado en la norma RS-232. Todas las normas RS-232 cumplen con los siguientes niveles de tensión (Figura 10):

- Un "1" lógico es un voltaje comprendido entre –3 V y –15 V.
- Un "0" lógico es un voltaje comprendido entre +3 V y +15 V.

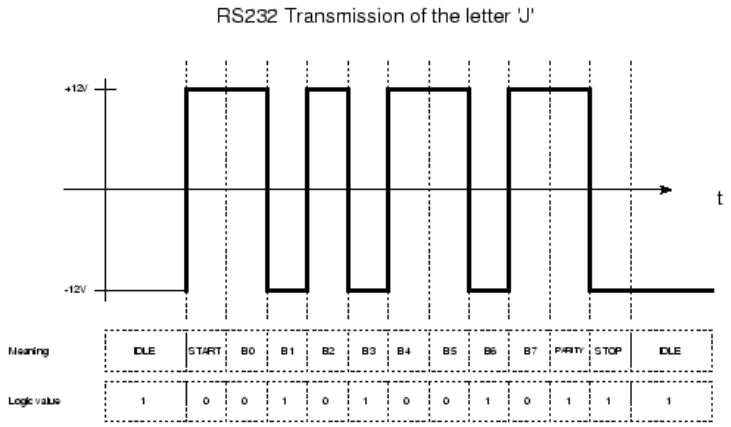

Figura 10. Trasmisión del carácter J (x"4A") con el estándar RS-232

Independientemente de la funcionalidad de cada bit (datos, paridad, etc.) todos tienen la misma duración, e igual a la inversa de la velocidad de transmisión.

En la Figura 11 se muestran los diferentes parámetros de la norma RS-232 que pueden seleccionarse para configurar la transmisión de una trama de datos.

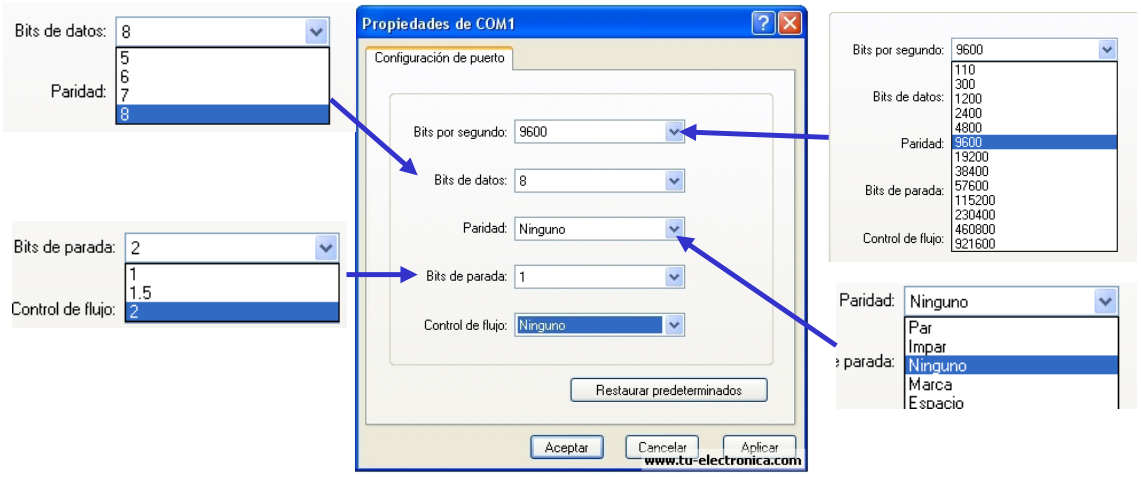

Figura 11. Posibles valores de los parámetros del protocolo RS-232.

Cuando no se envían datos, el puerto se mantiene a nivel alto, también conocido como *RS-232 idle state*. La transmisión se inicia con el bit de arranque (*start*)y, a continuación, se transmiten los bits correspondientes al dato, comenzando por el de menor peso (LSB) y terminando con el de mayor peso (MSB). El tamaño del dato puede ser 5, 6, 7, u 8 bits. **En esta práctica se va a utilizar el tamaño de 8 bits**.

Después del bit MSB del dato, se envía un bit de paridad que permite determinar si se ha producido un error durante la transmisión. Este bit puede ser suprimido de la trama (Figura 11). Cuando el valor del bit de paridad del dato recibido no coincide con el transmitido se dice que ha habido un error de paridad (*Parity Error*) y el dato debe rechazarse. La paridad puede configurarse como:

- Ninguno. No se transmite bit de paridad.
- Par. El bit de paridad es cero (0) si el dato tiene un número par de unos.
- Impar. El bit de paridad es cero (0) si el dato tiene un número impar de unos. **Este será el tipo de paridad empleado en esta práctica.**
- Marca. El bit de paridad siempre es uno.
- Espacio. El bit de paridad siempre es cero.

Después del bit de paridad, si lo hay, vienen los bits de parada (*stop bits*). Estos sirven para decir dónde termina la transmisión. Este puede ser uno, o dos y medio, en este caso, el segundo bit dura la mitad. Los bits de parada se transmiten como unos lógicos. Para esta práctica se va a considerar que el número de bits de parada es igual a uno.

Uno de los parámetros más importante del estándar *RS-232* es la velocidad de transmisión, o su inversa que es la duración de cada uno de los bits de la trama. Para esta práctica se va a utilizar una velocidad de transmisión de 921600 baudios.

Para realizar la recepción de un dato, en primer lugar hay que detectar cuando se inicia una transmisión, o lo que es lo mismo cuando la señal ***RX*** pasa a nivel bajo, a partir de aquí es necesario realizar tantas lecturas como bits tenga el dato más los bits de paridad (si los hay) y *stop*. Para evitar posibles errores en la transmisión debido a la presencia de ruido durante la transmisión de cada bit se deben tomar varias muestras de este, asignando el valor 0 o 1 según lo que se produzca en más ocasiones. El número de muestras que se deberán tomar para cada bit será igual al número del puesto más 5.

Una vez que se reciben todos los bits del dato, se deberá calcular el valor de bit de paridad, el cual debe ser comparado con el valor del bit de paridad recibido. Si estos son iguales y el bit de fin (*stop*) es correcto, se saca el dato por el puerto ***DATO_RX***, poniendo un nivel alto en ***DATO_RX_OK*** un tiempo igual a un periodo de la señal ***CLK*** para indicar que se ha recibido un nuevo dato y este es correcto. En el caso de que una de las dos comprobaciones anteriores falle se desecha el dato recibido, no se activa ***DATO_RX_OK*** y ***ERROR_RECEP*** se pone a nivel alto para indicar que el dato recibido es erróneo.

9.- Consideraciones de diseño.

A la hora de realizar el diseño de las diferentes entidades se deberá tener en cuenta una serie de recomendaciones para que los diseños sean óptimos:

- 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 que infieren un *latch*, señales no conectadas, listas de sensibilidad incompletas, etc.
- El diseño hace uso de la señal del oscilador de la placa *Basys3* cuya frecuencia de 100 MHz (T= 10 ns).
- El diseño debe funcionar obligatoriamente a 100 MHz. Uno de los índices de mayor calidad de un diseño, además de utilizar el menor número de recursos internos, es la máxima frecuencia de reloj. Esto se logra empleando mayoritariamente sistemas secuenciales, síncronos los cuales son gobernados con una única señal de reloj.
- El nombre de todas las señales de reloj de los bloques secuenciales utilizados en el diseño deberá ser ***CLK***, siendo el flanco activo el de subida.
- Todos los bloques secuenciales, y sólo ellos, utilizaran la señal ***RST*** como señal de inicialización asíncrona.

10.- Desarrollo y evaluación 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 la práctica 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.

La no entrega de alguno de los apartados supone una calificación de No Apto en el Laboratorio.

La nota final del laboratorio de basa en la evaluación de esta práctica libre y estará compuesta por las siguientes partes según la ponderación que aparece entre paréntesis:

1. Examen teórico de aspectos prácticos de los diferentes apartados de la práctica. Esta prueba tiene un peso del 35% de la nota final y se realizará el día **8 de mayo de 2023**.

2. Defensa individual de la práctica en la que el alumno deberá demostrar el conocimiento de las herramientas de diseño y de la práctica. Para ello, el alumno tendrá que realizar simulaciones funcionales y/o temporales de varios de los apartados en un tiempo limitado y responder a las preguntas que se planteen en relación con el código VHDL utilizado para modelar los diferentes módulos. Esta prueba tiene un peso del 40% de la nota final y se realizará el día **8 de mayo de 2023**.
3. Evaluación de los fuentes del diseño entregados (25% de la nota final). La fecha límite para la entrega de los fuentes será el jueves 4 de mayo de 2023 hasta las 14:00 horas.

11.- Descripción de los apartados a diseñar.

Apartado 1. Diseño del módulo *cnt_display*. (2 puntos)

1.1.- *Diseño del módulo **cnt_display**.*

El alumno deberá especificar el código VHDL que modele el controlador del display. Para que este apartado sea considerado como válido se deberá llevar a cabo la simulación funcional y temporal del modelo. Por consiguiente, será necesario realizar el correspondiente banco de prueba.

Para realizar las simulaciones del módulo *cnt_display*, junto con este enunciado se proporciona la entidad *display* que emula el funcionamiento del módulo de visualización de la placa Basys 3. La entidad *display* es sólo para simulación y, por tanto, no debe formar parte de la jerarquía para síntesis e implementación (Design Sources). Los alumnos deberán abrir, **nunca modificar**, este módulo para analizar y comprender su funcionamiento.

1.2.- *Descarga en placa del modelo del controlador del display.*

Los alumnos podrán verificar el funcionamiento de su módulo *cnt_display* en la placa de pruebas. Para ello, se proporciona el archivo de restricciones *cnt_display.xdc* con la asignación de los pines de la FPGA a los puertos de la entidad. Este archivo se va a utilizar para poder implementar y generar el archivo de configuración (*bitstream*) de la FPGA.

Una vez descargado en la placa el archivo de configuración se puede verificar la correcta visualización del dato seleccionado por los 16 switches (***SW15-SW0***) existentes en la placa *Basys3*. **Esta descarga es imprescindible para la evaluación del apartado. En el caso de que no funcione, se considerará el apartado como no apto.**

Apartado 2. Diseño del módulo *f_meter*. (2 puntos)

2.1.- *Diseño del módulo **f_meter**.*

El alumno deberá especificar el código VHDL que modele el medidor de frecuencia. Para que este apartado sea considerado como válido se deberá llevar a cabo la simulación funcional y temporal de los modelos. Por consiguiente, será necesario realizar el correspondiente banco de prueba.

2.2.- *Descarga en placa del modelo del módulo **f_meter**.*

Los alumnos podrán comprobar el funcionamiento de los módulos *f_meter* y *cnt_display* en la placa de pruebas. Para ello se proporcionan los ficheros *test_f_meter.vhd* y *test_f_meter.xdc*. Estos archivos se van a utilizar para poder implementar y generar el archivo de configuración (*bitstream*) de la FPGA.

Una vez descargado en la placa el archivo de configuración se puede verificar la correcta visualización de diferentes frecuencias en función del dato seleccionado por los switches (***SW12-SW0***) existentes en la placa *Basys3*. Los alumnos deberán abrir, **nunca modificar**, este módulo para analizar y comprender su funcionamiento obteniendo la relación entre la frecuencia de la señal generada y el valor seleccionado en los switches. **Esta descarga es imprescindible para la evaluación del apartado. En el caso de que no funcione, se considerará el apartado como no apto.**

Apartado 3. Diseño del módulo *signal_generator*. (2.5 puntos)

3.1.- *Diseño del módulo **signal_generator**.*

En este apartado el alumno deberá especificar un módulo VHDL que genera la señal a convertir a formato analógico con los parámetros seleccionados a través del puerto serie.

Para que este apartado sea considerado como válido se deberá llevar a cabo la simulación funcional y temporal del modelo.

Apartado 4. Diseño del módulo *dac_controller*. (2.5 puntos)

4.1.- *Diseño del módulo **dac_controller**.*

El alumno deberá especificar el código VHDL que modele el controlador que realiza la transmisión de un dato al DAC utilizando, para ello, el protocolo serie síncrono explicado en apartados anteriores. Para que este apartado sea considerado como válido se deberá llevar a cabo la simulación funcional y temporal del modelo.

Para realizar las simulaciones del módulo *dac_controller*, junto con este enunciado se proporciona la entidad *DAC121S101* que emula el funcionamiento del DAC. Esta entidad es sólo para simulación, no debiendo ser sintetizada. Los alumnos deberán abrir, **nunca modificar**, este módulo para analizar y comprender su funcionamiento.

4.2.- *Descarga en placa del módulo **test_ dac_controller**.*

En este subapartado, el alumno deberá comprobar el funcionamiento del módulo *dac_controller* en la placa de pruebas. Para ello, se proporciona el archivo fuente *test_ dac_controller*, así como el archivo de restricciones *test_ dac_controller.xdc* con la asignación de los pines de la FPGA a los puertos de la entidad. Este archivo se va a utilizar para poder implementar y generar el archivo de configuración (bitstream) de la FPGA. **Esta descarga es imprescindible para la evaluación del apartado. En el caso de que no funcione, se considerará el apartado como no apto.**

Apartado 5. Implementación del diseño completo (1 punto)
5.1.- *Simulación del módulo* ***top_system***.

El alumno deberá simular funcional y temporalmente la entidad *top_system*. Para ello, junto con este enunciado se proporciona la entidad *pc_tx* que emula la trasmisión de datos, la cual junto con los módulos *display* y *DAC121S101* deben estar instanciados en el test-bench. Asimismo, se debe instanciar la entidad *pc_tx*, entidad que emula el funcionamiento del puerto serie y al igual que las anteriores es sólo para simulación El alumno deberá analizar los códigos de estas entidades para comprender su funcionalidad a fin de analizar aquellas señales internas que puede ser necesario visualizarlas en la simulación.

5.2.- *Descarga en placa del modelo del controlador del módulo top_system.*

En este apartado el alumno deberá realizar la descarga en placa. Para esté último fin, se adjuntan el fichero *top_system.xdc* que contiene las restricciones asociadas a la asignación de los puertos de la entidad *top_system* con los pines de la FPGA.

12.- Documentación que entregar.

Para entregar la documentación correspondiente a la práctica libre se abrirá una Actividad en el Aula Virtual, en las que se indicará el listado de archivos a entregar. **LA ENTREGA DE MÁS O MENOS ARCHIVOS DE LOS SOLICITADOS CONLLEVARÁ SUPONE UNA CALIFICACIÓN DE NO APTO EN EL LABORATORIO.**

Aunque se publicará un anuncio con la información necesaria para la entrega de los ficheros de la práctica, en general los archivos solicitados serán los siguientes:

- Archivos fuentes (*.vhd) que se hayan utilizado para modelar y simular las diferentes entidades del diseño. Aunque se proporcionan los bancos de pruebas de todos los apartados, se deberán incluir como fuentes a entregar.
- El archivo de configuración *top_system.bit* utilizado para la descarga del módulo del nivel superior de la jerarquía.
- Los archivos de configuración de las ventanas de formas de onda (*.wcfg*). Indicar que será de obligado cumplimiento la utilización de estos archivos para cada una de las simulaciones (funcionales y temporales) que se lleven a cabo en los diferentes apartados.

En el caso de grupos de trabajo con dos alumnos, solamente se tendrá que hacer una entrega de los ficheros enumerados arriba.

14.- Planificación de entrega, explicación de apartados y evaluación final.

20-feb-2023			
Sem. 4	8:00-10:00	(P2-1)	Práctica libre: **Explicación apartado 1.**
	12:00-14:00	(P2-2)	Práctica libre: Trabajo apartado 1.
27-feb-2023			
Sem. 5	8:00-10:00	(P2-3)	Práctica libre: Trabajo apartado 1.
	12:00-14:00	(P2-4)	Práctica libre: Trabajo apartado 1.
06-mar-2023			
Sem. 6	8:00-10:00	(P2-5)	Práctica libre: **Explicación apartado 2.**
	12:00-14:00	(P2-6)	Práctica libre: Trabajo apartados 1 y 2
13-mar-2023			
Sem. 7	8:00-10:00	(P2-7)	Práctica libre: Trabajo apartados 1 y 2. **Explicación apartado 3.**
	12:00-14:00	(P2-8)	Práctica libre: Trabajo apartados 2 y 3**. Entrega apartado 1.**
27-mar-2023			
Sem. 8	8:00-10:00	(P2-9)	Práctica libre: Trabajo apartados 2 y 3.
	12:00-14:00	(P2-10)	Práctica libre: Trabajo apartados 2 y 3.
17-abr-2023			
Sem. 9	8:00-10:00	(P2-11)	Práctica libre: **Explicación apartado 4 y 5.**
	12:00-14:00	(P2-12)	Práctica libre: Trabajo apartados 2, 3, 4 y 5.
24-abr-2023			
Sem. 10	8:00-10:00	(P2-13)	Práctica libre: Trabajo apartados 2, 3, 4 y 5.
	12:00-14:00	(P2-14)	Práctica libre: Trabajo apartados 2, 3, 4 y 5.
08-may-2023			
Sem. 11	10:00-11:00		Evaluación individual de la Práctica libre.
	11:00-12:00		Evaluación individual de la Práctica libre.
	12:00-13:00		Evaluación individual de la Práctica libre.
	13:00-14:00		Evaluación individual de la Práctica libre.