PRÁCTICA 6

IMPLEMENTACIÓN DE UN SISTEMA DE PROCESADO EN TIEMPO REAL

Palabras clave: ADC, DAC, filtro digital, FPGA, SPI, VHDL.

1. INTRODUCCIÓN

Una FPGA es una plataforma adecuada para la implementación de sistemas de procesado que pueden demandar un gran número de recursos lógicos y una velocidad de ejecución elevada frente a una ejecución paso a paso típica de un sistema programado como puede ser un procesador. Por este motivo se presenta como una buena elección como plataforma de sistemas en tiempo real. El principal objetivo de esta práctica es el de desarrollar un filtro digital de promediado en tiempo real o de media móvil.

Para conseguir el objetivo de la práctica se integran los módulos desarrollados en las prácticas anteriores con uno nuevo que implementa el filtro de promediado de forma que se consiga un **sistema completo de procesado digital** de datos analógicos.

El desarrollo de la práctica se divide en dos fases:

1.- Se integran los módulos ya desarrollados en las prácticas 6 y 7, y se conectan los datos de entrada directamente a la salida (*by-pass*) sin realizar ningún procesado (Figura 3). Se adquiere la señal analógica (conversión A/D) y se reconstruye en la salida (conversión D/A). Se comprueba cómo el efecto del muestreo y reconstrucción afecta más al aumentar la frecuencia de la señal digitalizada.

2.- Se intercala el filtro de promediado de forma que funciona como un filtro paso-bajo (Figura 56). El sistema completo está formado por la unión de las tres etapas básicas (conversión A/D – filtrado – conversión D/A).

2. OBJETIVOS DE APRENDIZAJE

1.- Aprender a implementar sistemas complejos en VHDL organizados en diseños con varios niveles de jerarquía.
2.- Probar el funcionamiento de la arquitectura de un filtro digital sencillo en una FPGA.
3.- Saber utilizar módulos o componentes ya desarrollados.
4.- Saber simular y probar componentes.
5.- Utilizar recursos periféricos conectados a una FPGA. Realización del mapa de entradas y salidas. Asignación de pines de la FPGA.

3. TAREAS PREVIAS

Para una preparación adecuada de la práctica, además de la lectura detallada de este enunciado, es recomendable que el alumno realice los proyectos en *Quartus II* correspondientes a las tareas 1 y 2.

4. ESPECIFICACIONES DE LA INTERFAZ DEL SISTEMA DE PROCESADO DIGITAL

Tanto para la Tarea 1 como la Tarea 2, la interfaz del sistema de procesado que se implementa en la FPGA debe tener las siguientes entradas y salidas:

Señales de entrada:

clk: Señal de reloj principal. Se conecta al reloj de 50 MHz **CLOCK_50**.

reset: Señal de Inicialización activa a nivel bajo. Se conecta al pulsador **BUTTON2**.

en: Señal que activa el procesador. Se conecta al interruptor **SW0**.

sdi: Dato serie de entrada a la unidad de control del ADC.

Señales de salida:

1. Control del ADC:

   cs_adc: Selección del ADC.

   sck_adc: Señal de sincronismo y temporización del ADC.

2. Control del DAC:

   cs_dac: Selección del DAC.

   sck_dac: Señal de sincronismo y temporización del DAC.

3. Salidas de datos al DAC:

   sdo: Dato serie de salida de la unidad de control al DAC.

   eop: Indica el fin del procesado de un dato. Equivale al fin de una transmisión serie del dato de salida desde el procesador al DAC. Se conecta al **LEDG0**.

5. DESARROLLO DE LA PRÁCTICA

Tarea 1: Con los recursos hardware preparados en las anteriores prácticas realizar un bypass (muestreo, retención y reconstrucción) con una señal analógica continua de entrada. Visualizar en el osciloscopio la entrada y la salida analógicas.

Es necesario establecer una señal de muestreo de datos de entrada analógicos. Dicha señal tendrá una frecuencia de 25 kHz y se obtiene dividiendo por 2000 el reloj principal de 50 MHz. Para ello se utiliza un módulo **divisor_reloj.vhd** como el que se utilizó en el controlador del DAC.

El nivel superior de la jerarquía del diseño está formado por tres módulos. Dos correspondientes a los controladores SPI de los convertidores y otro correspondiente al divisor que proporciona el reloj de muestreo.

La salida **dout** del controlador **spi_adc** se debe conectar a la entrada **din** del módulo **spi_dac**.

También se debe tener en cuenta que las señales de **reset** de los dos convertidores no son activas con el mismo nivel. Por tanto, como el **reset** se obtiene del pulsador, que proporciona un '0' cuando se pulsa, se debe conectar directamente al **reset** del módulo **spi_dac** e invertido al módulo **spi_adc**.

Para realizar el diseño de este sistema se deben seguir los siguientes pasos:

1. Crear un proyecto nuevo en el entorno de *Quartus II*.
2. Añadir al proyecto los ficheros de diseño utilizados en las prácticas anteriores. Los correspondientes al PLL, se pueden generar de nuevo por el asistente manteniendo el mismo nombre (**clkout**) (Figura 1).
3. Crear un nuevo fichero en VHDL en el que se diseñe una entidad de jerarquía superior con una arquitectura de tipo estructural que agrupe los dos controladores y el divisor de reloj según el esquema de la Figura 3.
4. Asignar los terminales de entrada y salida de la FPGA con la ayuda de la herramienta ***Pin Planner*** (Figura 2) teniendo en cuenta que las conexiones entre los conectores de expansión GPIO 0 y GPIO 1 y los circuitos convertidores de la placa de prototipos deben ser las mismas que la de las prácticas anteriores. Además, asignar las señales del LED (eop), el interruptor (en) y del pulsador (reset).

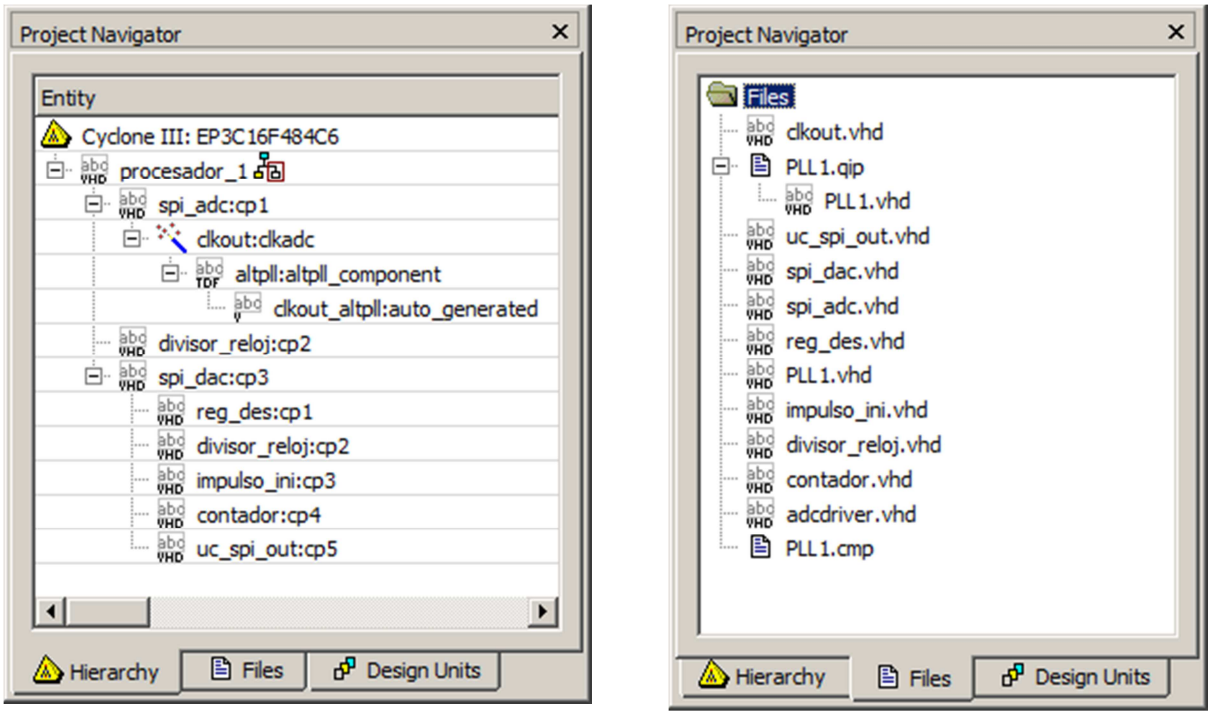

Figura 1. Ficheros del proyecto de la Tarea 1.

Node Name	Direction	Location
clk	Input	PIN_G21
cs_adc	Output	PIN_U15
cs_dac	Output	PIN_AA16
en	Input	PIN_J6
eop	Output	PIN_J1
reset	Input	PIN_F1
sck_adc	Output	PIN_R16
sck_dac	Output	PIN_AB16
sdi	Input	PIN_V7
sdo	Output	PIN_AA15

Figura 2. Asignación de los terminales de entrada/salida de la FPGA.

Una vez compilado el diseño del sistema completo, comprobar con el visor RTL que presenta la siguiente arquitectura:

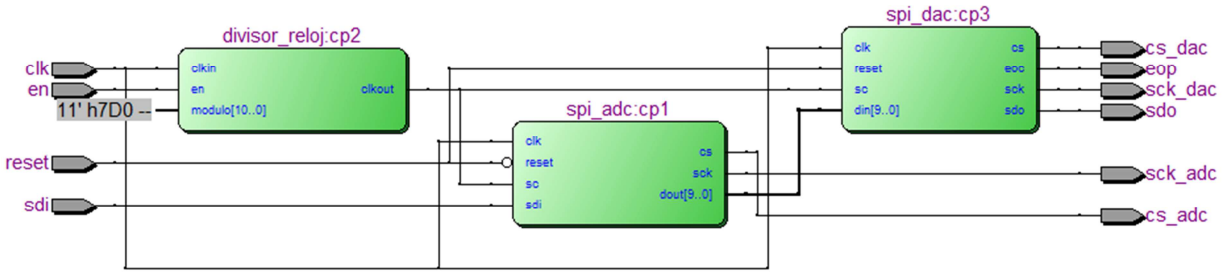

Figura 3. Diagrama RTL del diseño de la Tarea 1.

Después de comprobar que se ha realizado correctamente el diseño, se realizan las conexiones entre la placa DE0 y los dispositivos de la placa de prototipos. A continuación se conecta a la entrada analógica (patilla 2 del IC MCP3001) una señal sinusoidal de valor mínimo 0.1 V y valor máximo de 3.1 V, es decir, que sea de 3 Vpp.

Finalmente, se completa la tarea comprobando con el osciloscopio que aparece reconstruida la señal de entrada en la salida analógica (terminal 8 del IC MCP4911). Visualizar en la pantalla del osciloscopio las señales de salida para señales de entrada de 100 Hz y 2.7 kHz aprox. El resultado debe ajustarse a la siguiente imagen:

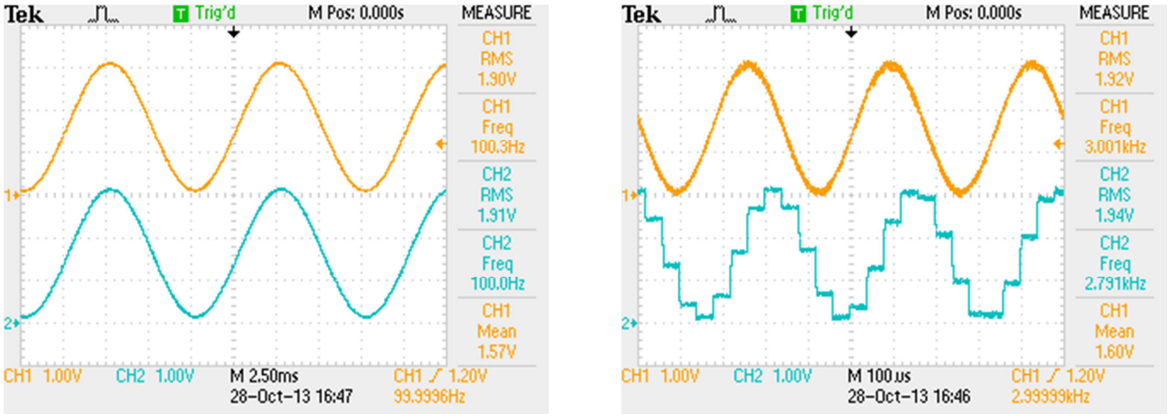

Figura 4. Oscilogramas con la señal de entrada (CH1) y la de salida (CH2) a distintas frecuencias. En la izquierda las señales son de 100 Hz y no se observa ningún desfase ni atenuación en la señal. En la de la derecha, se observa un cierto retraso y, sobre todo, el efecto de la cuantificación (escalones de tensión).

Tarea 2: Implementación de un filtro digital de promediado con entrada y salida analógicas para intercalar en el circuito de la tarea anterior: entrada analógica – filtro digital – salida analógica.

La funcionalidad de este sistema, consiste en muestrear una **señal analógica de entrada**, digitalizar sus valores y aplicarles un proceso de filtrado, concretamente un filtro de promediado, y mostrar el valor promedio (señal digital) como **valor analógico de salida**. La parte que se implementa en la FPGA se muestra en la figura:

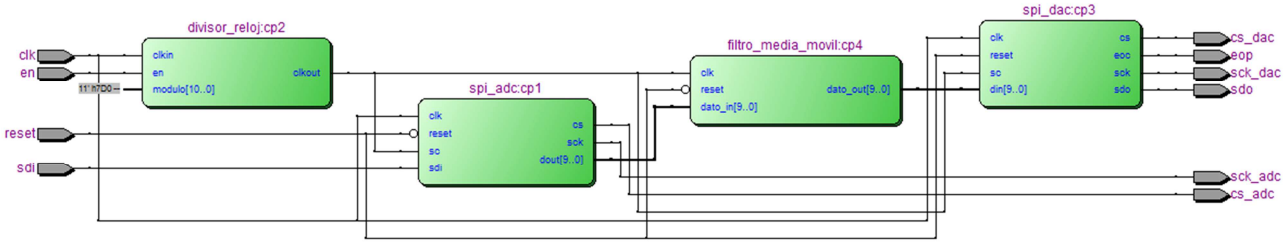

Figura 5. Diagrama con la estructura del sistema de procesado que se configura en la FPGA.

Como se puede observar, se puede tomar como punto de partida el proyecto desarrollado en la tarea anterior, solo es necesario incluir el filtro entre los dos controladores. Tras digitalizar los valores analógicos de entrada, estos constituyen la entrada del filtro y el resultado de este proceso se incorpora al módulo de conversión analógica. Después de pasar este proceso de promediado la señal de salida no debería presentar el aspecto escalonado derivado del muestreo en señales de entrada de frecuencia del orden de Khz.

El filtro de promediado se configura con 32 etapas (**longitud**) y con un ancho de datos de 10 bits. Para implementarlo se sigue la descripción del ANEXO I, que consiste en almacenar las 32 últimas muestras, sumarlas y dividirlas por el número de etapas o veces que se han sumado, es decir, por 32.

En el ANEXO I se muestra el nivel superior de la jerarquía del diseño y en la Figura 6 dos oscilogramas que deben servir de referencia para probar el filtro.

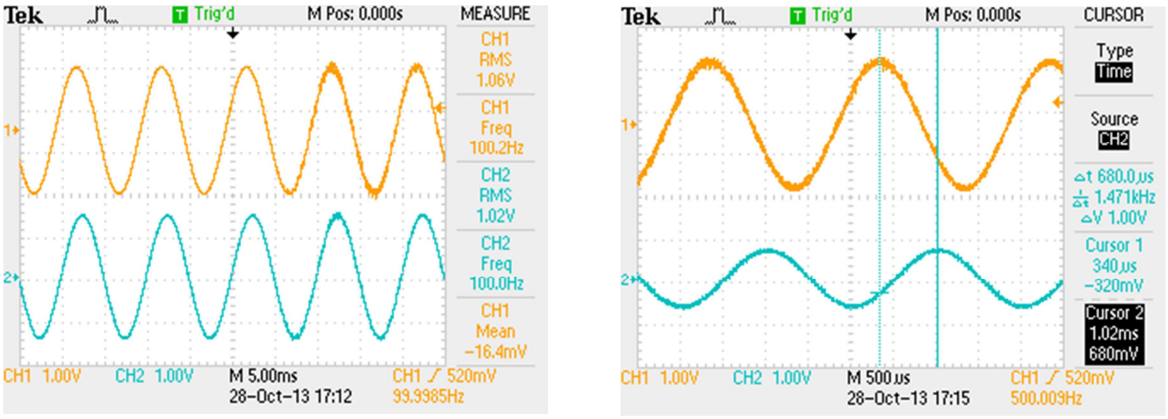

Figura 6. Oscilogramas de prueba del filtro. En el de la izquierda se aprecia cómo a 100 Hz no hay atenuación ni desfase apreciables entre las señales de entrada y de salida. En el de la derecha, que corresponde a una señal de 500 Hz, sí que ya hay cierta atenuación y desfase.

6. EVALUACIÓN

	calificación
1.- Realización del *by-pass* en *Quartus II* (Tarea 1).	
3.- Análisis de resultados: Oscilogramas (Tarea 1).	
3.- Sistema de muestreo y filtrado en *Quartus II* (Tarea 2).	
4.- Análisis de resultados: Oscilogramas (Tarea 2).	

ANEXO I Descripción VHDL del filtro de promediado.

```
------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE IEEE.math_real.all;

ENTITY filtro_media_movil IS
--longitud: numero de datos que se promedian
--n: resolucion de los datos (numero de bits)
GENERIC(longitud:INTEGER:=4;
        n:INTEGER:=8);
PORT(clk,reset: IN STD_LOGIC;
     dato_in: IN UNSIGNED((n-1) DOWNTO 0);
     dato_out: OUT UNSIGNED((n-1) DOWNTO 0));
END filtro_media_movil;

ARCHITECTURE descripcion OF filtro_media_movil IS
SIGNAL media: UNSIGNED((integer(ceil(log2(real(longitud))))+n-1) DOWNTO 0);
TYPE vector IS ARRAY (0 TO (longitud-1)) OF UNSIGNED ((n-1) DOWNTO 0);
SIGNAL fifo:vector;
BEGIN

 PROCESS(reset,clk)
   BEGIN
      IF reset='1' THEN
         FOR i IN 0 TO (longitud-1) LOOP
           fifo(i)<=(OTHERS =>'0');
         END LOOP;

      ELSIF (clk='1' AND clk'event) THEN
          FOR i IN 1 TO (longitud-1) LOOP
            fifo(i)<=fifo(i-1);
          END LOOP;
          fifo(0)<=dato_in;
       END IF;
  END PROCESS;

 PROCESS(fifo)
    VARIABLE suma: UNSIGNED((integer(ceil(log2(real(longitud))))+n-1) DOWNTO 0);
    BEGIN
       suma:=(OTHERS =>'0');
       FOR i IN 0 TO (longitud-1) LOOP
         suma:=suma+fifo(i);
       END LOOP;
       media<=(suma/(longitud));
  END PROCESS;

  dato_out<=media((n-1) DOWNTO 0);
END descripcion;
```

Nivel superior del diseño del procesador digital.

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

ENTITY procesador_2 IS
GENERIC(nbits: integer:=10);
PORT(clk:  IN std_logic;
     reset:IN std_logic;
     sdi:  IN std_logic;
     en:   IN std_logic;
     sck_dac: OUT std_logic;
     cs_dac:  OUT std_logic;
     sck_adc: OUT std_logic;
     cs_adc:  OUT std_logic;
     eop:     OUT  STD_LOGIC;
     sdo:     OUT  STD_LOGIC);
END;

ARCHITECTURE circuito OF procesador_2 IS

 SIGNAL dato_in,dato_out:  UNSIGNED(nbits-1 DOWNTO 0);
 SIGNAL dato_1,dato_2:  STD_LOGIC_VECTOR(nbits-1 DOWNTO 0);

 SIGNAL not_reset, reloj_muestreo:  std_logic;
 SIGNAL valor_division :  STD_LOGIC_VECTOR(10 DOWNTO 0);

COMPONENT spi_adc IS
GENERIC(n: INTEGER:=10);
PORT(clk,reset,sc,sdi: IN STD_LOGIC;
     sck,cs: out STD_LOGIC;
     dout: out STD_LOGIC_VECTOR(n-1 DOWNTO 0) );
END COMPONENT;

COMPONENT spi_dac
   PORT(reset,sc,clk:  IN  STD_LOGIC;
        din: IN  STD_LOGIC_VECTOR(9 DOWNTO 0);
        sdo,eoc,sck,cs:  OUT  STD_LOGIC);
END COMPONENT;

COMPONENT divisor_reloj
GENERIC (n: INTEGER);
   PORT(clkin,en: IN STD_LOGIC;
        modulo : IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
        clkout : OUT STD_LOGIC);
END COMPONENT;

COMPONENT filtro_media_movil
GENERIC(longitud:INTEGER:=32;
              n:INTEGER:=10);
PORT(clk,reset: IN STD_LOGIC;
     dato_in: IN UNSIGNED((n-1) DOWNTO 0);
     dato_out: OUT UNSIGNED((n-1) DOWNTO 0));
END COMPONENT;

BEGIN
not_reset <= NOT reset;
--el reloj de 50 MHz se divide por 2000
--para obtener una frecuencia de 25 kHz
valor_division<="11111010000";

--se adpata la entrada del filtro a unsigned
--y la salida a logic_vector
dato_in<=UNSIGNED(dato_1);
dato_2<=STD_LOGIC_VECTOR(dato_out);

cp1: spi_adc
   PORT MAP(clk => clk, reset => not_reset, sc => reloj_muestreo,
            sdi => sdi, sck => sck_adc,     cs => cs_adc,
            dout => dato_1);

cp2: divisor_reloj
   GENERIC MAP(n => 11)
   PORT MAP( clkin => clk, en => en, modulo => valor_division,
             clkout => reloj_muestreo);

cp3: spi_dac
   PORT MAP(clk => clk, reset => reset, sc => reloj_muestreo,
            sdo => sdo,sck => sck_dac, cs => cs_dac,
            eoc => eop, din => dato_2);

cp4: filtro_media_movil
   PORT MAP(clk => reloj_muestreo, reset => not_reset,
            dato_in => dato_in, dato_out => dato_out);
END circuito;
```