PRÁCTICA 4

DISEÑO E IMPLEMENTACIÓN DE UNA UNIDAD DE ACOPLAMIENTO SERIE SPI PARA UN CONVERTIDOR A/D

Palabras clave: ADC (*Analog to Digital Converter*), FPGA, PLL (*Phase Locked Loop*), SPI (*Serial Peripheral Interface Bus*), VHDL, AL (Analizador Lógico).

1. INTRODUCCIÓN

Esta es la primera de las prácticas que componen el Proyecto 2 de la asignatura. Este proyecto tiene como objetivo aprender a manejar señales analógicas con una FPGA. Recordar que ya en el Proyecto 1 se capturaba una señal analógica proporcionada por el potenciómetro de la placa Demo del PICkit3 con el ADC del uC (entrada analógica) y, por otro lado, se proporcionaba la señal de actuación para el motor a través del PWM (salida analógica).

Ahora, en este proyecto, se adquiere una señal analógica con un ADC (MCP 3001) y se reconstruye con un DAC (MCP4911), los cuales, como se muestra en la Figura 1, se conectan en la placa de prototipos. En ambos casos la interfaz entre el convertidor y el procesador digital (la FPGA) es del tipo serie SPI. Por tanto, también es otro objetivo importante en el Proyecto 2 aprender cómo funciona esta forma de conexión y desarrollar los controladores necesarios para manejar los convertidores. Tanto el procesado digital que se haga con la señal muestreada como los propios controladores se implementarán en la FPGA.

Dentro de lo que es el Proyecto 2, en esta práctica se acopla con un ADC de 10 bits de resolución una señal analógica de tensión obtenida con un potenciómetro a la FPGA y la combinación digital resultante se visualiza en los LEDs de la placa DE0. Cada vez que se presiona el pulsador de inicio de conversión se debe realizar una adquisición.

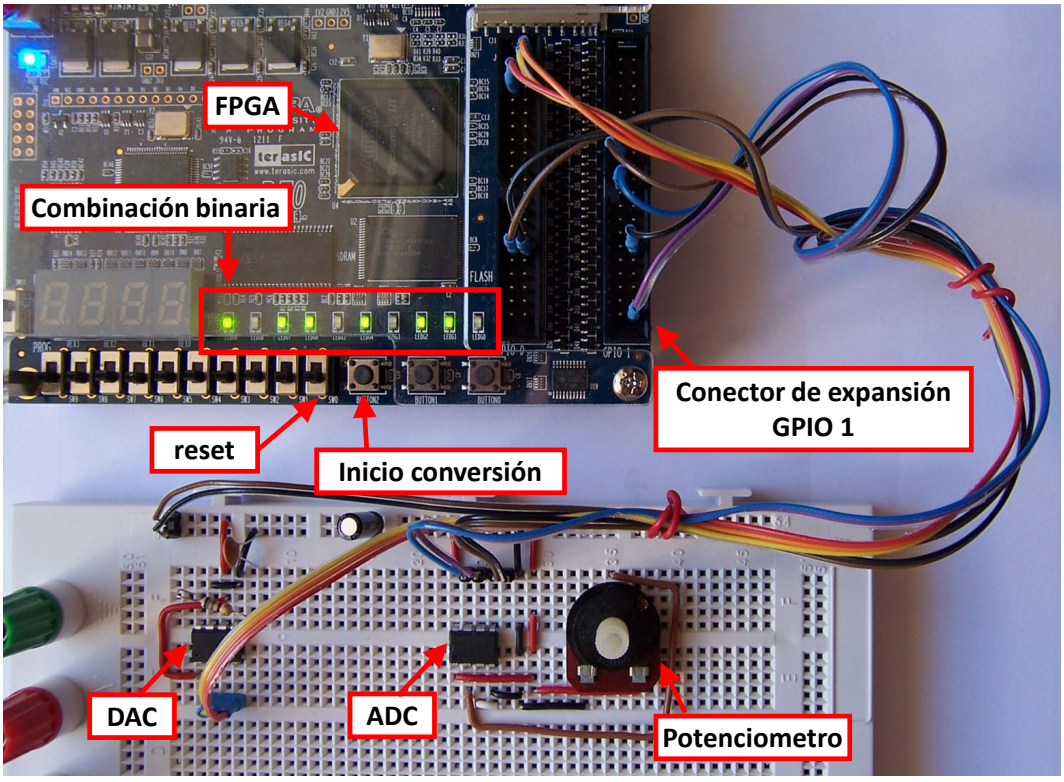


Figura 1. Fotografía del montaje de los convertidores ADC y DAC en la placa del prototipos y su interconexión con la placa DE0.

Para conseguir estos objetivos, se debe realizar un proyecto en el entorno de diseño de la FPGA (Quartus II) que implemente un controlador SPI, o driver, para el circuito ADC y, además, como elemento complementario, se utiliza un sintetizador de frecuencias (PLL) de los que tiene incorporados la FPGA para obtener el reloj de sincronismo de la interfaz SPI. Esto implica además, utilizar el asistente del Quartus II para configurar el PLL.

2. OBJETIVOS DE APRENDIZAJE

1. Repasar el proceso de implementación de sistemas digitales en una FPGA: Diseño, simulación, compilación, síntesis, configuración y prueba.
2. Saber extraer información útil de las hojas de características de un circuito para poder integrarlo en un sistema.
3. Comprender el funcionamiento de un convertidor analógico-digital o ADC y conocer los parámetros más importantes en su funcionamiento.
4. Comprender el funcionamiento de un controlador de un periférico conectado a un procesador digital con interfaz serie síncrona SPI.
5. Saber conectar un circuito ADC a un procesador digital en un prototipo.

3. TAREAS PREVIAS

Para una preparación adecuada de la práctica, además de la lectura detallada de este enunciado, el alumno debe realizar las siguientes tareas previas:

1. Estudiar los temas de teoría sobre el funcionamiento del bus SPI.
2. Leer y entender las hojas características del ADC (MCP3001) y de la FPGA utilizada (EP3C16F484C6) de la familia CYCLONEIII (especialmente en lo referido a los PLLs).
3. Realizar el diagrama de flujo de la descripción algorítmica del controlador **spi_adc.vhd** de la Tarea 2.
4. Realizar un proyecto con el módulo fuente **spi_adc.vhd** del controlador disponible en el curso virtual y el componente **clkout** siguiendo los pasos indicados en la Tarea 2. Es aconsejable repasar las fases del proceso de diseño en el entorno Quartus II como se estudió en cursos anteriores.
5. Identificar en la placa que contiene a la FPGA (DE0) los recursos adecuados para la prueba del controlador diseñado para el ADC y los terminales necesarios. Se debe realizar un esquema eléctrico del circuito completo a entregar como requisito previo para trabajar en el laboratorio y, a ser posible, el montaje del sistema (Figura 1).
6. Se recomienda revisar el funcionamiento del analizador lógico (AL) con el que se ha trabajado en el curso anterior.

4. DESARROLLO DE LA PRÁCTICA

Tarea 1: Estudio de un módulo de control de la comunicación serie y del formato de datos.

El dispositivo fundamental para la comunicación de dispositivos electrónicos digitales con el entorno analógico es el convertidor A/D (se lee analógico a digital) o ADC. Para su estudio se ha seleccionado un dispositivo comercial (MCP3001). En base a sus características, en esta práctica se implementan las funciones básicas necesarias para la comunicación entre el dispositivo y un procesador de datos. Para ello se modela dicho circuito de comunicación en VHDL sintetizable y se implementa en una FPGA de la familia CYCLONEIII de Altera integrada en una placa de prueba (DE0). La documentación proporcionada por los fabricantes de cada elemento se encuentra disponible en el curso virtual de Moovi.

La Figura 2 muestra el aspecto externo y los datos de conectividad con el exterior del ADC objeto de estudio.

En las hojas de características el alumno debe estudiar con detalle los valores que toman VDD, VSS y VREF, y como varía la salida DOUT en función de las entradas IN-/IN+, CLK y CS.

La Figura 2 muestra el protocolo de transmisión serie de la trama que genera el ADC tras la conversión de un valor analógico a un valor digital de 10 bits.

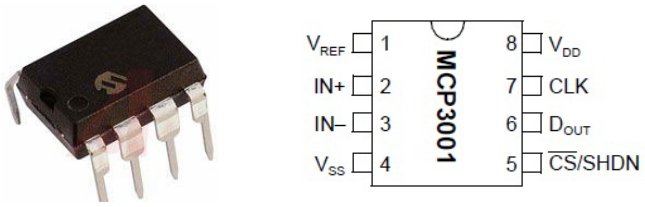
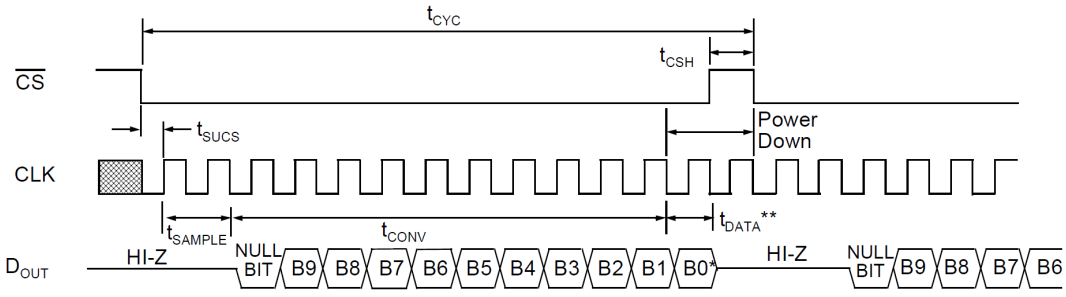


Figura 2. Imagen del DAC MCP3001 y su esquema de conexiones.



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the ADC will output LSB first data, followed by zeros indefinitely. See Figure below.

** t_{DATA}: during this time, the bias current and the comparator powers down and the reference input becomes a high impedance node.

Figura 3. Protocolo de transmisión serie de un dato [B9…B0] en la salida DOUT con respecto a las señales CS y CLK.

En base a esta secuencia proporcionada por el ADC, el controlador que se diseña en esta práctica debe almacenar los bits que definen el dato en formato digital (B9, B8, …, B0) y proporcionar una salida paralelo DATO(9 downto 0). Para ello se tendrán en cuenta aspectos importantes como:

- La frecuencia de la señal de reloj de conversión a la que funciona correctamente el ADC (CLK) debe estar por debajo de la señal del oscilador de la placa conectada a la FPGA.
- El ADC transmite la información en cada flanco negativo del reloj.
- CS es la señal que habilita el comienzo del proceso de conversión.
- Antes de comenzar a almacenar los bits de información hay que tener en cuenta los ciclos de espera de conversión (ciclos 1 y 2 de CLK) y el bit de comienzo de trama (NULL BIT).

Tarea 2: Diseño e implementación de un módulo de control SPI para conexión a un convertidor A/D.

Crear un proyecto nuevo en el entorno de desarrollo Quartus II de Altera para el diseño del circuito controlador del ADC. Añadir al proyecto el fichero **spi_adc.vhd** disponible en el curso virtual (Faitic), que describe la funcionalidad requerida.

Un controlador debe proporcionar todas las señales de entrada necesarias al dispositivo periférico y recibir las salidas para procesarlas o almacenarlas. La interfaz de entrada-salida del controlador queda definida en la entidad (**ENTITY**):

```
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY spi_adc IS
GENERIC(n: INTEGER:=10);  --número de bits del dato
PORT
  ( clk, reset, sc, sdi: IN STD_LOGIC;
    sck,cs: out STD_LOGIC;
    dout: out STD_LOGIC_VECTOR(n-1 downto 0) );
END;
```

Para comprender la arquitectura interna es necesario tener en cuenta la función que realizan las señales de entrada y salida del módulo controlador spi_adc:

- **clk** es la señal de reloj que la FPGA recibe del oscilador de 50 MHz de la placa DE0.
- **sck** es la señal de reloj proporcionada al ADC por el controlador spi_adc, que se debe generar configurando uno de los PLLs internos de la FPGA de forma que proporcione una señal de salida c0 de 1 MHz.
- **sc** (*Start Conversion*) es la señal de muestreo externa proporcionada por el usuario (se debe conectar al pulsador **Button2** de la placa DE0).
- **cs** (*Chip Select*) es la señal de muestreo conectada al ADC, que debe generar el controlador a partir de la señal de inicio de conversión **sc**.
- **reset** es la señal de inicialización, que se obtiene a partir del interruptor **SW0** de la placa DE0.
- **dout** es el bus de datos paralelo de salida, se debe conectar a los **LEDs** de la placa DE0.

Como se puede observar, la descripción completa de la arquitectura incluye un proceso que implementa el protocolo de transmisión serie y además un componente, **clkout**, que tiene como objeto sintetizar la señal de reloj **sck** de 1 MHz. La instanciación de este componente es la siguiente:

```
clkadc: clkout port map(areset=> reset, inclk0=> clk, c0 => ck);
```

A continuación se describe cómo se obtiene el módulo **clkout** utilizando al asistente del software de desarrollo Quartus II.

El entorno de diseño de la FPGA permite al usuario incluir en sus modelos *IP cores* (*Intellectual Property cores*), componentes prediseñados que ya han sido precompilados y probados. Estos componentes cubren un amplio espectro de campos (aritmética, almacenamiento, temporización, etc.) y están definidos en función de ciertos parámetros que el diseñador puede modificar.

Un PLL (*Phase-Locked Loop*) es un sistema basado en un oscilador controlado por tensión (VCO) con una realimentación entre las fases de las señales de entrada y de salida. Permite la implementación de señales de reloj estables facilitando la tarea de generación y distribución de señales de temporización en la FPGA. A continuación se describe el proceso para la definición del componente **clkout** a partir de la interfaz ALTPLL. A través de esta interfaz será posible configurar uno de los PLLs de la FPGA para obtener una señal de reloj de 1 MHz a partir de la señal de temporización genérica proporcionada por el oscilador de la placa (50 MHz). Esta interfaz está incluida en la herramienta del entorno (*MegaWizard Plug-in Manager*) destinada a configurar para el usuario cualquiera de los *IP cores* disponibles. Para generar el módulo **clkout** seguir los siguientes pasos:

1.- Desde el menú principal ejecutar la orden **Tools>MegaWizard Plug-In Manager** y seleccionar la opción **Create** en el cuadro de diálogo que se abre (Figura 4).

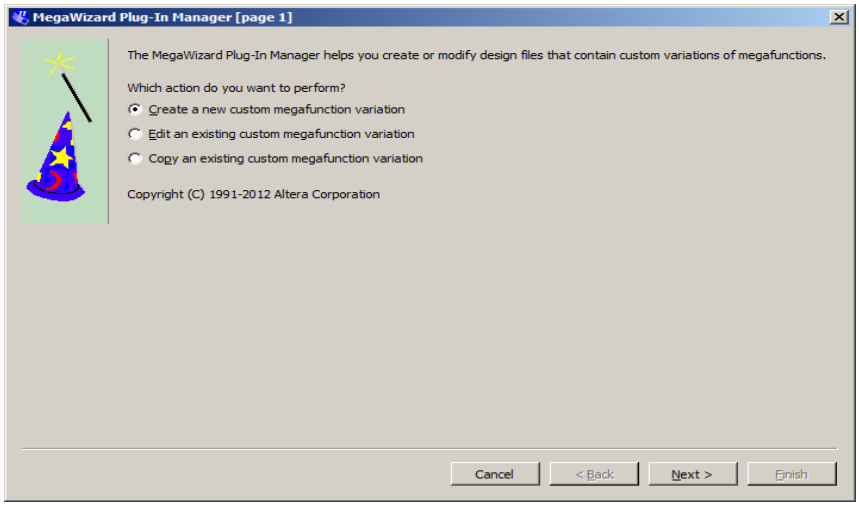


Figura 4. Asistente para la configuración de componentes prediseñados.

2.- A continuación se abre el recuadro de la Figura 5 en el que se debe seleccionar el componente **ALTPLL**, que está dentro de la carpeta **I/O,** elegir la opción de crear un fichero de VHDL y, finalmente, escribir el nombre de dicho fichero que contendrá el diseño (en este caso se ha puesto el nombre **clkout**).

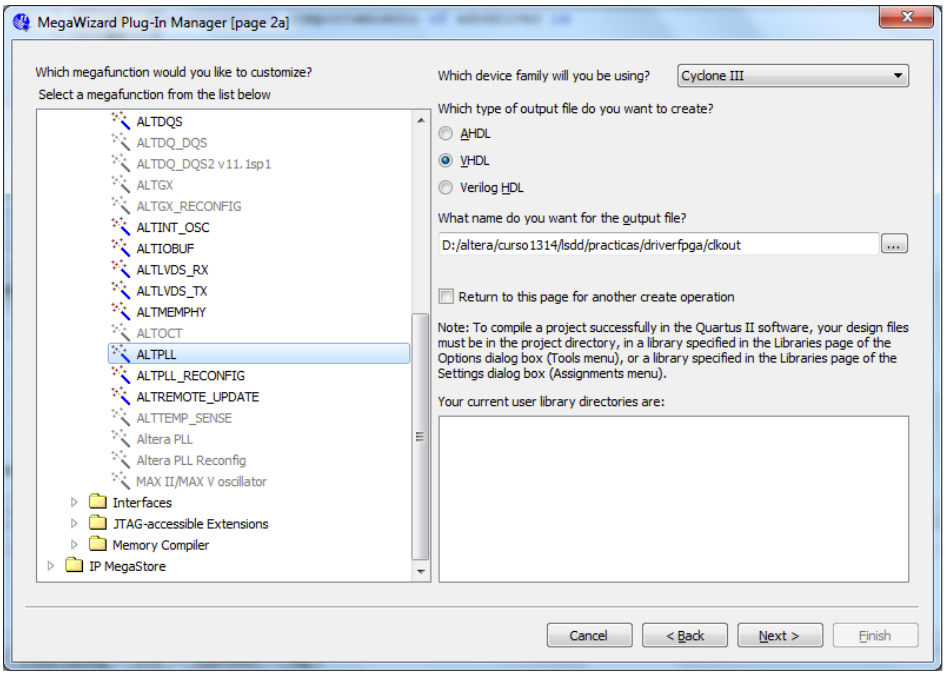


Figura 5. Elección del módulo PLL.

3.- En el siguiente paso de configuración se debe especificar el valor de la frecuencia de entrada al PLL, que es de 50 MHz.

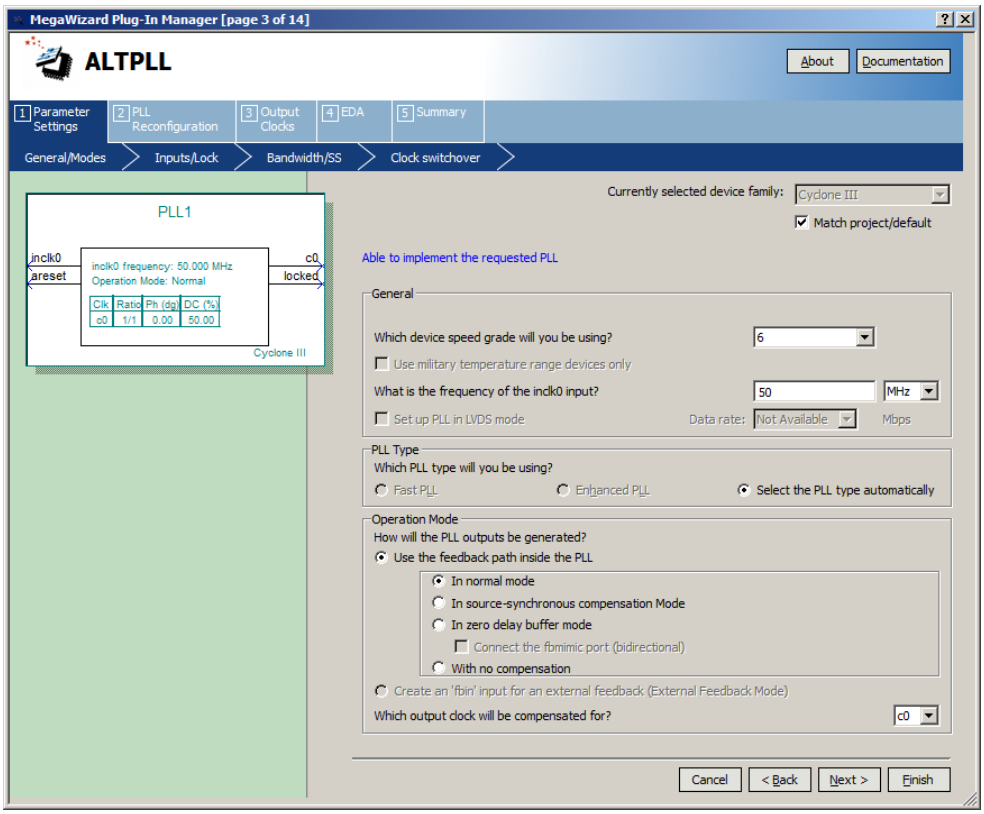


Figura 6. Elección del tipo de PLL y de la frecuencia de entrada.

4.- En la siguiente pantalla dejar las opciones por defecto, excepto la salida Lock Output, que no se utilizará en el diseño, por lo que se puede eliminar.

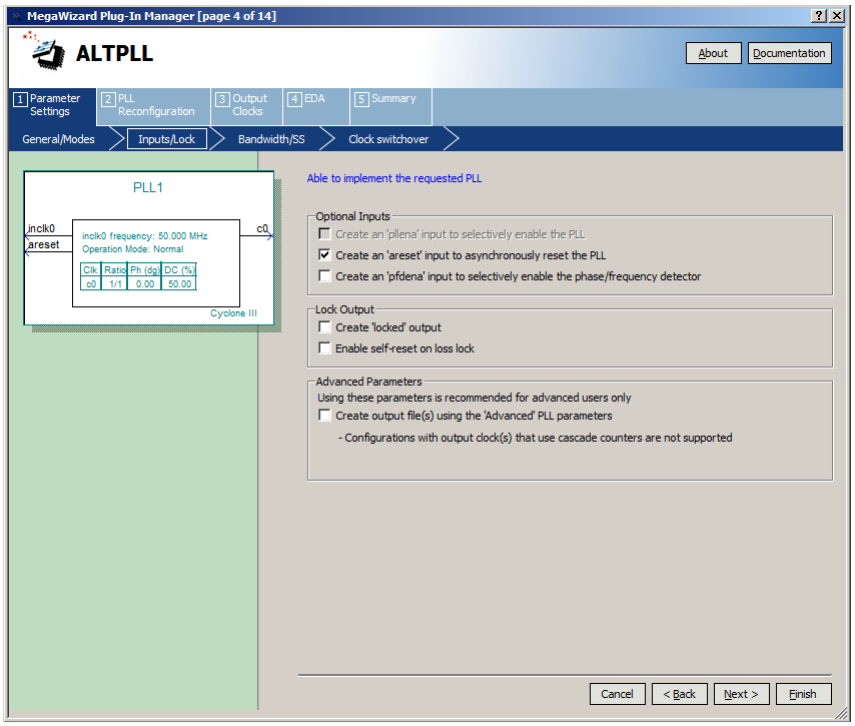


Figura 7. Configuración del PLL.

5.- En el siguiente paso se debe configurar la frecuencia de salida que se desea y el ciclo de trabajo del reloj, que será del 50%. Solamente se configurará el reloj c0 (Figura 8). Las siguientes pantallas se deben aceptar con los valores con los que aparecen por defecto, hasta llegar a la última pantalla, que debe quedar como la de la Figura 9.

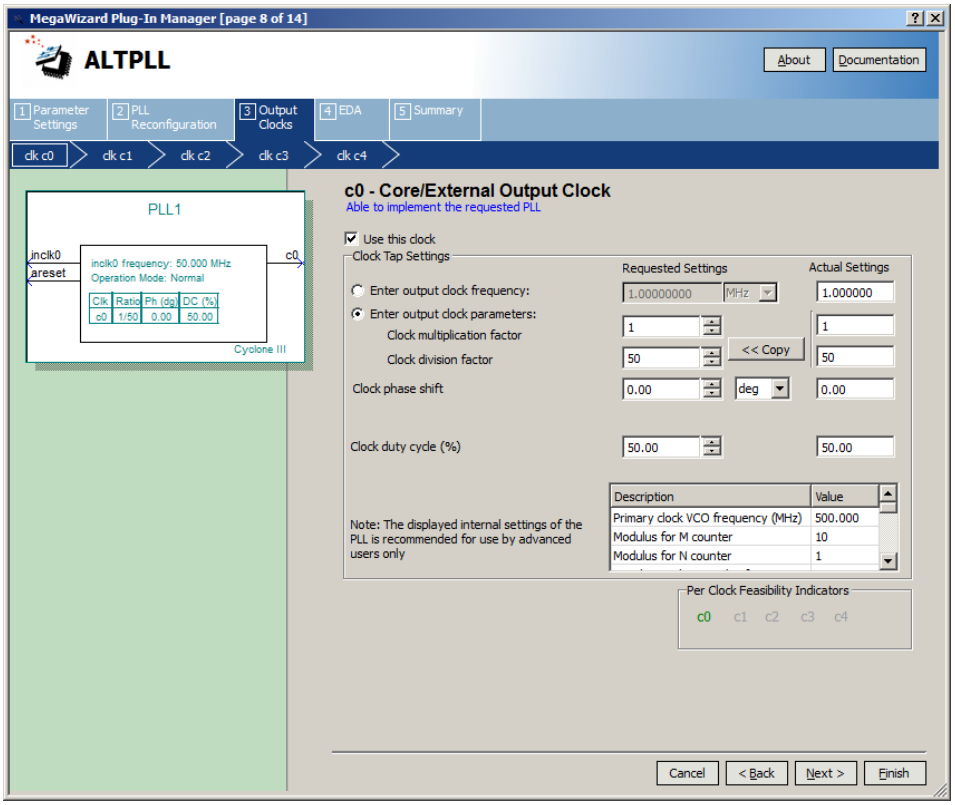


Figura 8. Configuración de la frecuencia de salida del PLL.

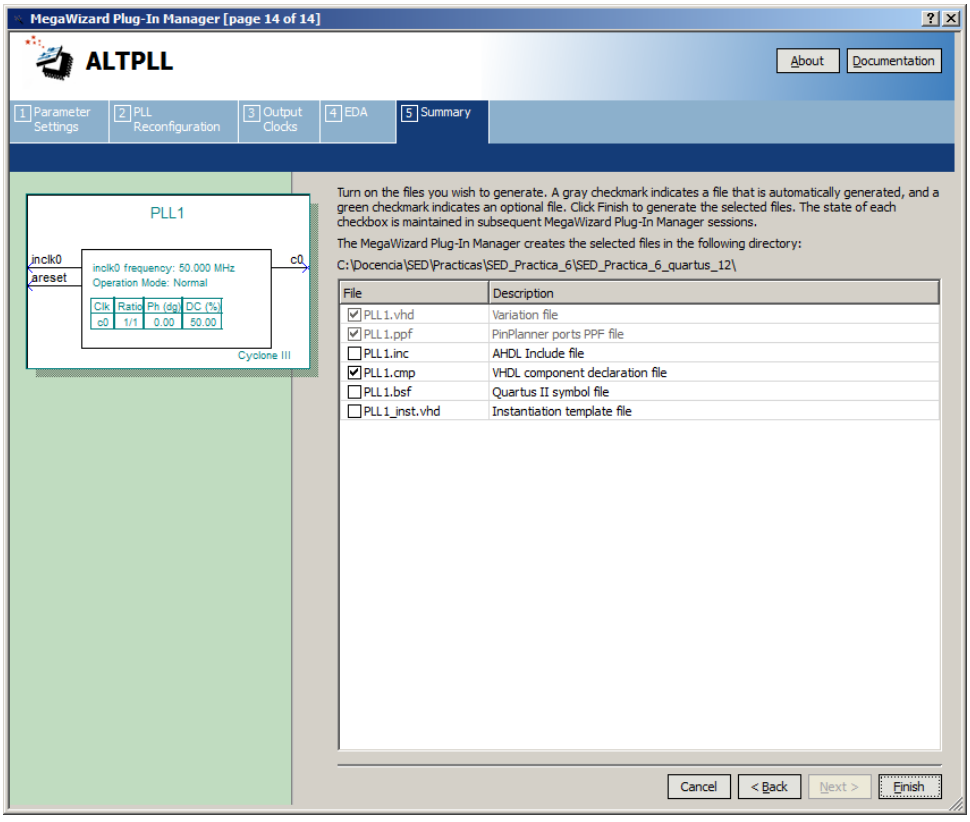


Figura 9. Último cuadro del asistente de configuración del PLL.

Al terminar todos los pasos del asistente, el programa añade los ficheros del diseño del módulo IP al proyecto.

Una vez preparado el proyecto se deben asignar los terminales de la FPGA, compilarlo de nuevo y realizar una simulación para comprobar el funcionamiento correcto del sistema. Para la simulación se debe generar un fichero de simulación (test_bench) con extensión .vht que incorpore los vectores de prueba (recordar que para generar la plantilla desde el Quartus se usa la orden **Processing>Start>Start Test Bench Template Writer**). Para este caso se proponen los siguientes valores de prueba:

```
48    reset <= '1', '0'  after 1000 ns;
49    sc    <= '1', '0'  after 3000 ns;
50
51    dato: PROCESS
52    BEGIN
53      sdi <='0'; wait for 6500 ns;
54      sdi <='1'; wait for 1000 ns;
55      sdi <='0'; wait for 1000 ns;
56      sdi <='1'; wait for 2000 ns;
57      sdi <='0'; wait for 2000 ns;
58      sdi <='1'; wait for 7500 ns;
59
60    END PROCESS dato;
61
62    reloj: PROCESS
63    BEGIN
64       clk <= '0'; wait for 10 ns;
65       clk <= '1'; wait for 10 ns;
66    END PROCESS reloj;
```

Figura 10. Vectores de prueba del controlador **spi_adc**.

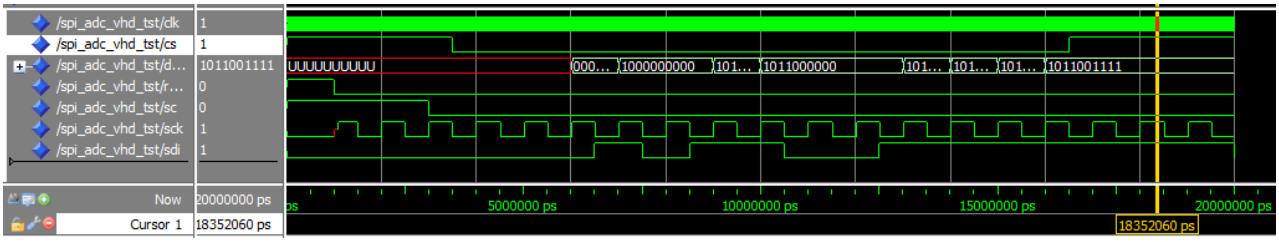


Figura 11. Resultado de la simulación del controlador **spi_adc**.

Tarea 3: Captura de una entrada analógica con un circuito convertidor A/D con interfaz serie SPI.

Conectar en la placa de prototipos el ADC y el potenciómetro y realizar las interconexiones con la placa DE0. Proporcionar al dispositivo las señales de alimentación (VDD) y la tensión de referencia (VREF) correspondientes. Elegir los puertos de conexión adecuados en la FPGA para la entidad **spi_adc**. La señal de salida **dout** se debe visualizar en los LEDs de la placa (LED0 – LED9). Se debe tener en cuenta que, después de la asignación de los terminales de la FPGA, es necesario compilar el proyecto de nuevo. En la siguiente tabla se resumen los terminales de los componentes. Utilizar el esquema realizado en las tareas previas.

Tabla 1. Interconexiones entre la FPGA y el ADC.

Patilla ADC	Denominación de la patilla en el MCP3001	Conector Expansión DE0	Patilla Conector DE0 (J5)	Pin FPGA
1 y 8	VREF y VDD Alimentación y referencia	GPIO1	29	
2	Entrada analógica			
3 y 4	Referencia 0 V	GPIO1	30	
5	CS	GPIO1	15	PIN_U15

6	DOUT	GPIO1	40	PIN_V7
7	CLK	GPIO1	19	PIN_R16

Para generar la señal analógica variable de entrada se utiliza un potenciómetro o una resistencia ajustable en modo divisor de tensión, es decir, se conectan sus extremos al positivo de la alimentación (3.3 V) y a la de referencia (0 V), y la toma media a la patilla 2 del ADC.

Una vez comprobado el correcto funcionamiento del sistema completo, se deben tomar distintos valores de la tensión continua de entrada (toma media del potenciómetro) utilizando el multímetro o el osciloscopio y anotar su valor frente a la combinación binaria obtenida en los LEDs. Comprobar si coincide con el valor teórico esperado (utilizar la ecuación del Apartado 4.2 de la hoja de características del ADC).

Para una mayor exactitud, no contar con el valor nominal de 3.3 V para la tensión de referencia, si no medirlo también con la instrumentación.

Tabla 2. Valor de la tensión de referencia (tensión en la patilla 1 del ADC).

	V

Tabla 3. Tabla de resultados.

Muestra nº	Tensión de entrada Vdc	Combinación binaria obtenida	Valor teórico de la tensión de entrada	Error cometido
1				
2				
3				
4				

Tarea 4: Utilización del AL para monitorizar el puerto SPI.

Por último, monitorizar las señales del bus SPI con el analizador lógico y capturar una trama correspondiente a una conversión, como se muestra en el siguiente ejemplo:

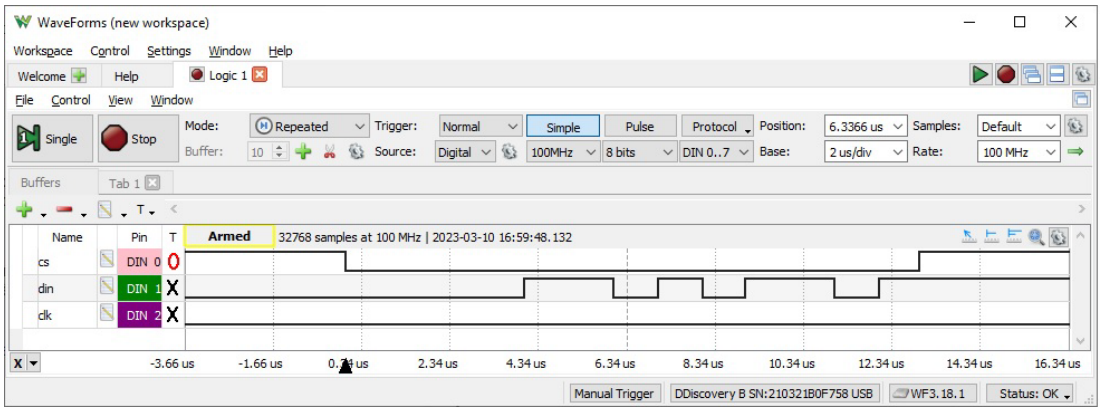


Figura 12. Captura de una trama del bus mediante el analizador lógico.

5. EVALUACIÓN

	calificación
1.- Tareas Previas.	
• Diagrama de flujo.	
• Esquema eléctrico.	
• Proyecto Quartus II (Tarea 2).	
• Montaje (Tarea 3).	
3.- Simulación (Tarea 2).	
4.- Tabla de valores (Tarea 3).	
5.- Captura de valores digitalizados en AL.	

ANEXO I:
Módulo **spi_adc.vhd**

```
--Laboratorio de Electrónica Digital. Universidad de Vigo
--Módulo de control del circuito ADC MCP3001
--conectado a la placa DE0
--Conexión de las señales de entrada:
--    clk: se conecta al reloj de 50 MHz de la placa
--    reset: se conecta al interruptor SW0
--    sc: Start Conversión, se conecta al pulsador BUTTON2
--    din: Se conecta al terminal 6 del ADC
--Conexión de las señales de salida:
--    sck: es el reloj SPI de 1 MHz
--    cs: es la señal SPI de selección del periférico
--    dout: es la palabra de datos
 LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;

ENTITY spi_adc IS
 GENERIC(n: INTEGER:=10);  --número de bits del dato
 PORT
   ( clk, reset, sc, sdi: IN STD_LOGIC;
     sck,cs: out STD_LOGIC;
     dout: out STD_LOGIC_VECTOR(n-1 downto 0) );
 END;

ARCHITECTURE comportamiento OF spi_adc IS

 SIGNAL ck,sc_prev: STD_LOGIC;

COMPONENT clkout port
    ( areset : IN STD_LOGIC:= '0';
      inclk0: IN STD_LOGIC:= '0';
      c0    : OUT STD_LOGIC );
 END COMPONENT;

 BEGIN

 clkadc: clkout PORT MAP(areset=>reset, inclk0=>clk, c0=>ck);

 sck <= ck;

datoin: PROCESS(reset,ck)
   VARIABLE start, scint: std_logic;
   VARIABLE dato: std_logic_vector(9 downto 0);
   VARIABLE indice : integer range 0 to 11;
   VARIABLE altaimp : integer range 0 to 15;

 BEGIN                    -- incialización asíncrona
   IF reset='1' THEN
      dout <= (OTHERS => '0');
      scint:='0';
      indice:=0;
      altaimp:=0;           -- contador de los dos ciclos inciales
      sc_prev <= '1';
   ELSE                     -- activo a flanco positivo
      IF (ck'EVENT and ck='1') THEN
          IF (sc='0' and scint='0') THEN
             sc_prev<= '0';
             scint:='0';
             altaimp:=altaimp+1;
             IF (sdi= '0' and altaimp>=4 and indice=0) THEN  -- ciclos iniciales
                indice:=1;
                sc_prev<= '0';                                -- habilitación
                dato:= (OTHERS => '0');
             ELSIF (indice>=1 and indice <11) THEN  -- estados de transmision de dato
                dato(10-indice):= sdi;              -- mas desabilitación
                IF (indice <10) THEN    -- transmision de dato
                   sc_prev<= '0';
                ELSE dout <= dato;      -- desabilitación
                   scint:='1';
                   sc_prev<= '1';
                END IF;
                indice:=indice+1;
             ELSE NULL;
             END IF;
          ELSIF sc='1' THEN --requiere flanco positivo en habilitación externa
            scint:='0';     --para nueva transmisión
            indice:=0;
            altaimp:=0;
          ELSE NULL;
          END IF;
      ELSE NULL; -- clk
      END IF;
   END IF;
  dout<= dato;

 END PROCESS;

habil: PROCESS(reset, ck)
 BEGIN
    IF (ck'EVENT AND ck='0') THEN
      cs <= sc_prev;
      ELSE NULL;
    END IF;
END PROCESS;
END ARCHITECTURE;
```