Escuela Politécnica Superior,
Grado en Informática

Asignatura: Diseño Automático de Sistemas

Práctica 2

22 de febrero de 2022

UNIVERSIDAD NEBRIJA

UNIVERSIDAD
NEBRIJA

Pablo Sarabia Ortiz

Índice/Tabla de contenidos

1. Introducción

1.1. Presentación

Esta guía va a explicar el diseño completo de la interfaz para adquirir los datos del micrófono incluido en la placa Nexys 4 DDR. Parte de la implementación de la adquisición de los datos se continuará realizando en la práctica 3.

1.2. Bibliografía recomendada

Siempre que se va a utilizar una interfaz de hardware debéis tener el datasheet y el manual de referencia del HW que vais a utilizar en nuestro caso el manual de referencia para la Nexys 4DDR lo encontramos aquí. El datasheet del micrófono ADMP421 está disponible aquí.

1.3. Objetivo

Elaborar los bloques necesarios para poder registrar las medidas del micrófono ADMP421. En está práctica se va entender el funcionamiento de la interfaz PDM (Pulse Density Modulation) cómo almacenarla y después reproducirla. Para ello se necesitan de una serie de bloques funcionales: un registro de desplazamiento, un divisor de reloj, un controlador de memoria y un generador de PWM.

En la práctica 2 se va a implementar el registro de desplazamiento y el generador de reloj.

2. Micrófono en Nexys4 DDR

2.1. Interfaz física

La placa Nexys4 DDR tiene un micrófono MEMS (MicroElectroMechanical System) que permite grabar el audio de forma omnidireccional. La interfaz es la siguiente:

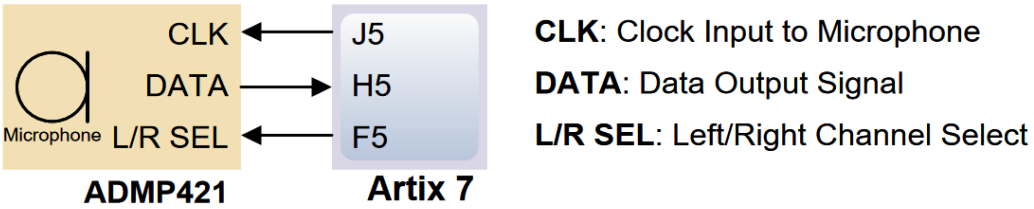


Figure 24. Microphone block diagram.

Esta interfaz consta de CLK (en el puerto J5 de la FPGA), que debemos generar en nuestro diseño. El puerto DATA (en el puerto H5 de la FPGA) de los datos que vienen del micrófono que es un stream de datos de un bit de ancho con modulación PDM (Pulse Density Modulation). Por último, tenemos el puerto de L/R SEL(en el puerto F5 de la FPGA) que nos permite elegir el canal (izquierdo o derecho).

2.2. Interfaz digital

El micrófono acepta en la señal de reloj frecuencias desde 1 MHz a 3.3 MHz a mayor frecuencia mayor calidad de audio, también al utilizar una frecuencia mayor vamos a necesitar más espacio para almacenar cada muestra. En nuestro diseño usaremos la **frecuencia de 1.5 MHz.** Por último, la señal L/R Sel decide si muestreamos en el flanco de subida o de bajada y nos serviría para tener audio estéreo, en el proyecto vamos a

usar mono así que nos basta con poner la señal a 0 y así obtendremos un canal en el flanco ascendente del reloj, se corresponde con el DATA1 de la figura inferior. El protocolo que utiliza el micrófono es muy sencillo y se muestra en la siguiente figura, el dato nos llega con el flanco de subida del reloj, en nuestro caso DATA2 no es necesario ya qye hemos puesto la señal L/R a 0:

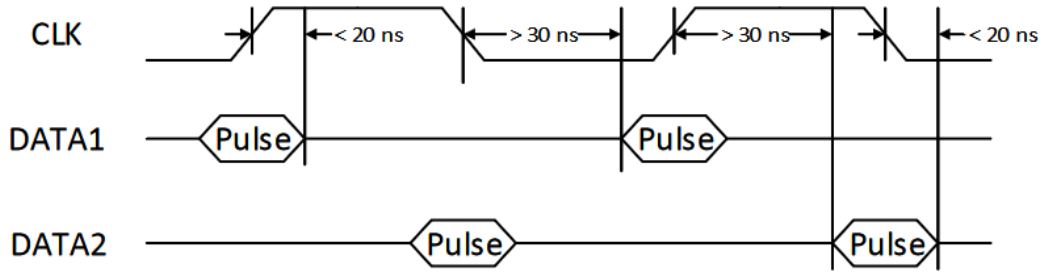
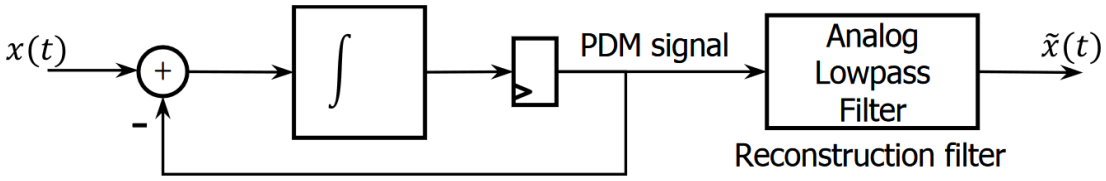


Figure 27. PDM Timing Diagram.

2.3. Pulse Density Modulation (PDM)

Habitualmente los micrófonos MEMS utilizan una modulación PDM que es el resultado de usar un ADC por aproximación (sigma-delta). Es muy popular porque requiere solo de un puerto para comunicar los datos. Se basa en realizar un sobremuestreo.



La amplitud de la señal original (audio) es proporcional a la densidad de los pulsos. A diferencia de un PWM la señal no tiene una frecuencia fija. Se puede recuperar la señal original usando un filtro de paso bajo analógico. A continuación, se muestra una figura con una onda sinusoidal modulada en PDM.

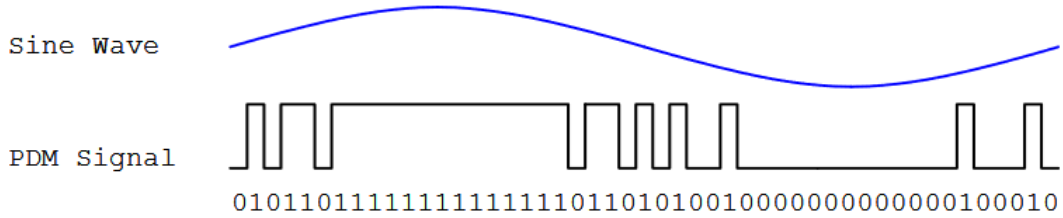


Figure 25. PDM representation of a sine wave.

Para nuestro diseño final necesitaremos convertir esta señal a una señal PCM (Pulse Code Modulation) PCM consiste en almacenar en una palabra binaria el valor de la señal mediante el uso de filtros, lo veremos en las siguientes prácticas.

3. Generador de reloj síncrono

3.1. Introducción

Debido a que debemos generar un reloj a una frecuencia inferior (1.5 MHz) al del sistema (100 MHz) se va necesitar generar un reloj síncrono con el principal (SCLK). Además, debido a que nuestro sistema va a funcionar a 100 MHz vamos a usar una estrategia, clock enable, para evitar los diferentes dominios de reloj (y los problemas de metaestabilidad que pueden derivar), consiste en que el bloque de generación de reloj nos genere también un pulso de un ciclo (a la frecuencia del sistema 100 MHz) que sirva para indicar que es el pulso de subida del reloj a menor frecuencia.

Hasta ahora en nuestros sistemas había un único dominio de reloj y por tanto nuestras señales registradas seguían la siguiente estructura:

```
process (clk, n_rst)
begin
      if n_rst = '1' then
            state    <= CLEAR;
            r_BCD0   <= ( others=> '0');
            r_BCD1   <= ( others=> '0');
            r_rstcnt <= ( others=> '0');
      elsif rising_edge(clk)then
            state    <= stateNext;
            r_BCD0   <= BCD0;
            r_BCD1   <= BCD1;
            r_rstcnt <= rstcnt;
      end if;
end process;
```

Clock Enable consiste en que en el dominio de reloj de menor frecuencia las señales registradas van a tener dos condiciones (la habitual rising_edge(clk)) y la que sea el flanco del reloj (SCLK_rise) síncrono (SCLK) que hemos generado, también generaremos la señal de SCLK_fall que indica el flanco de bajada. Quedando de la

```
process (clk, n_rst)
begin
      if n_rst = '1' then
            state    <= CLEAR;
            r_BCD0   <= ( others=> '0');
            r_BCD1   <= ( others=> '0');
            r_rstcnt <= ( others=> '0');
      elsif rising_edge(clk)and(SCLK_rise = '1')then
            state    <= stateNext;
            r_BCD0   <= BCD0;
            r_BCD1   <= BCD1;
            r_rstcnt <= rstcnt;
      end if;
end process;
```

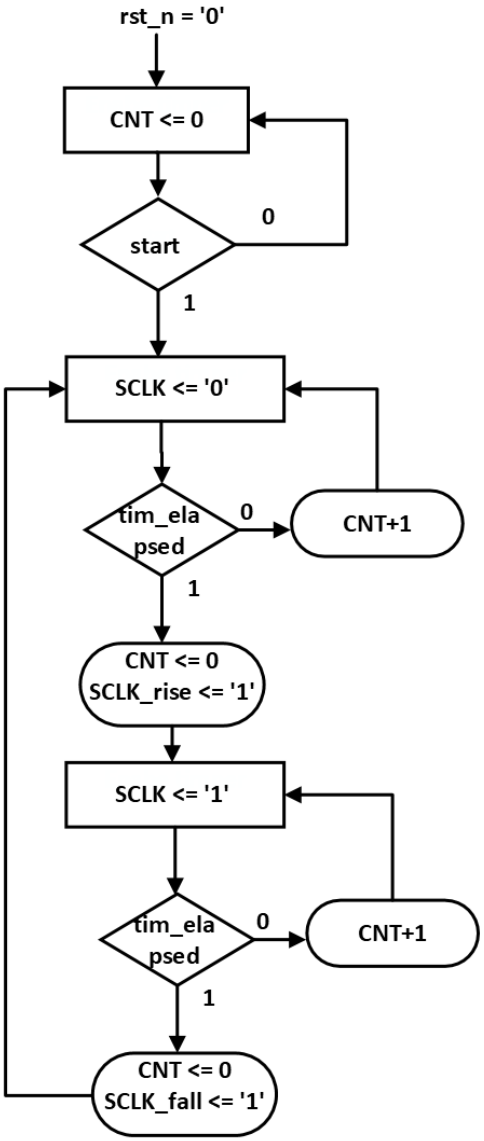
siguiente manera.

3.2. Interfaz del módulo

El módulo del generador del reloj va a tener la siguiente interfaz como entradas start, clk y rst_n. Como salidas SCLK, SCLK_rise, SCLK_fall. Cómo genéricos la frecuencia del reloj del sistema y la frecuencia de reloj deseada.

3.3. Funcionamiento (FSM)

En la siguiente imagen se muestra el diagrama ASM del generador de reloj CNT es una señal de longitud c_counter_width que almacena el valor del contador. También es necesario calcular otra constante que es c_half_T_SCLK que contiene el numero de ciclos para comparar con el contador CNT. time_elapsed es la comparación (CNT = c_half_T_SCLK).



C_half_T_SCLK es la mitad del periodo del reloj SCLK, debéis usar los genéricos. Se va a redondear hacia abajo. Es decir si la frecuencia deseada no es un divisor de la frecuencia del sistema se escoge una un poco superior.

Las señales SCLK_rise y SCLK_fall sólo duran un ciclo de reloj.

Se pide implementar la máquina de estados en dos procesos, tal y como se ha visto en clase

3.4. Fichero proporcionado

Se proporciona el fichero fsm_SCLK.vhd con la interfaz del módulo.

3.5. Simulación

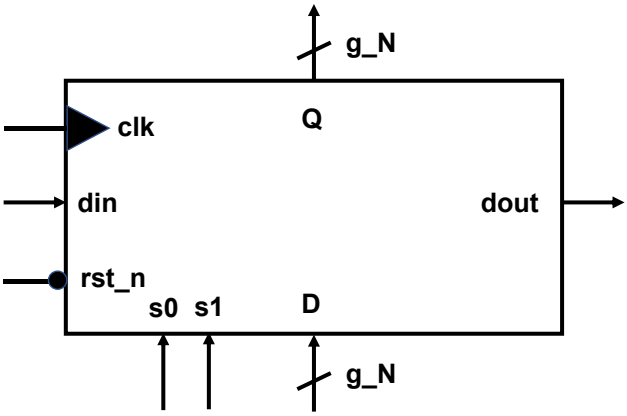
Se debe implementar un testbench que compruebe que el funcionamiento de la máquina de estados es correcto. Para reducir el tiempo de simulación se recomienda subir la frecuencia deseada a un valor de 20 MHz.

3.6. Mejora opcional

Mejora opcional usar el comando *assert report* para mostrar la frecuencia real a la que va a funcionar, este comando no se sintetiza, pero permite que la labor de debugging sea más sencilla. Preguntad al profesor si lo queréis implementar.

4. Registro de desplazamiento con carga en paralelo

Tal y como hemos mencionado en el apartado 2 el micrófono utiliza una interfaz de un único hilo para comunicar los datos con la FPGA. Sin embargo, para procesar y almacenar los valores es mucho más cómodo utilizar palabras con un tamaño de bits. Para poder pasar de serie a paralelo es necesario usar un registro de desplazamiento (shift register). En esta práctica vamos a implementar un shift register parametrizable con carga en paralelo. Este registro nos va a permitir tanto pasar de serie a paralelo como viceversa, con tamaño del registro variable. También se podrá hacer rotar los bits en ambos sentidos.



4.1. Diseño y funcionamiento

El registro va a permitir cargar datos en paralelo por ello necesitamos dos señales de control que nos permitan bien desplazar el dato o bien cargar en paralelo el registro. El registro va a tener dos entradas de datos y dos salidas, dos en paralelo y dos en serie.

Las señales s0 y s1 funcionan de la siguiente manera (se entiende que rotar hacia la derecha es del MSB en dirección al LSB):

S0	S1	Acción
0	0	Sin cambios
0	1	Rotar hacia la derecha
1	0	Rotar hacia la izquierda
1	1	Carga paralelo

En la salida Q se corresponde **siempre** con los bits que hay actualmente en el registro. Por din los datos entrarán con la frecuencia correspondiente y salen por dout a la misma frecuencia. El reset asíncrono debe cargar el registro con 0s. D y Q dependen su tamaño del genérico g_N que determina el tamaño del registro de desplazamiento.

4.2. Implementación y simulación

Para este modulo el alumno implementará el modulo completo incluyendo interfaz (usando los nombres provistos en la figura en la que se describe el bloque) y se deberá realizar una simulación en la que se verifique que las rotaciones de los datos son correctas en ambos sentidos. Se recomienda elegir un g_N pequeño para reducir el número de ciclos que tarda el dato en salir por la salida dout.

En la implementación va a ser necesario usar bucles for repase la teoría

5. Evaluación y entrega

5.1. Grupos

Se utilizarán los mismos grupos que en la anterior práctica.

5.2. Puntuación

Este hito se corresponde con un **30%** del valor del proyecto final. La entrega se realizará tras el último día de prácticas de la asignatura donde se evaluará el proyecto completo, para esta entrega final será necesario adjuntar un pdf documentando el proyecto. El código será revisado por el profesor en la siguiente sesión de prácticas. Para la revisión es necesario llevar el proyecto funcionando y con una visualización de las señales de interés clara que permita ver el funcionamiento del sistema.