Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica

IEE2433 - Diseño de Circuitos Integrados Digitales - 2018

Tarea 1: Filtro FIR

Juan Francisco Troncoso Z.

Resultados de la Simulación

El filtro diseñado de 16 coeficientes cumplió satisfactoriamente las especificaciones de diseño tales como respetar la frecuencia de entrada y salida de los datos de audio, aun cuando nuestra tasa de procesamiento es mas elevada, específicamente de $384kHz$, trabajar con coeficientes fraccionarios, lograr una sincronizacion de todos los módulos con un reset, por lo que no fue necesario retardos y lograr una correcta respuesta al impulso.

El testbench consistió en generar un clock arbitrario ($10ns$) dado que no se pudo simular en la plataforma de CloudV uno con la tasa de procesamiento especificada anteriormente y con la cantidad de flancos necesaria para lograr una respuesta rápida. La variable *nuevo_clk* corresponde al clock de procesamiento, y *clk_lento* corresponde a la tasa de llegada y salida de los datos de audio. La entrada al filtro corresponde a un impulso de una duración especifica para que solo sea muestreado una vez por el filtro. Finalmente tenemos *salida_filtro* que representa la salida del sistema con un tamaño de los 20 bits mas significativos que entrega todo el proceso. El sistema se inicia y se sincroniza con un reset como se observa en la siguiente figura.

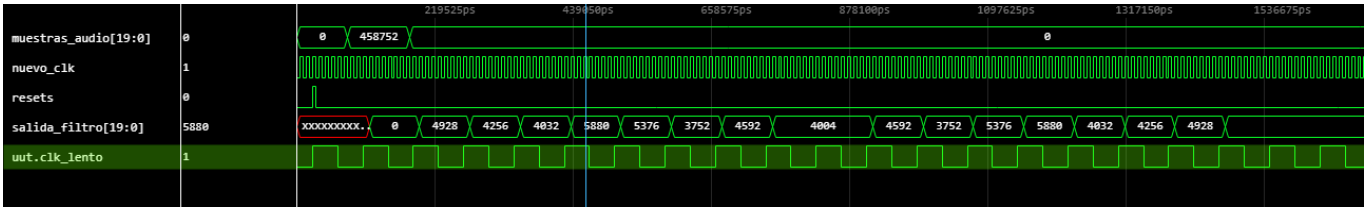

Figura 1: Resultados de la simulacion del filtro FIR.

Como hemos desarrollado una función de transferencia discreta simétrica, es de esperar que al introducir un impulso, se observen los coeficientes ponderados con un simétrico tal como muestra *salida_filtro*. De todas formas, en el anexo de este documento esta el testbench utilizado en la Figura 1.

Diagrama de bloques

El diagrama de bloques que representa la conexión entre los modulos diseñados se presenta en la Figura 2. Cada modulo presenta las entradas y salidas que se han declarado en el codigo de Verilog. En síntesis, los datos de audio contenidos en 20 bits, son muestreados a $48kHz$ comandados por el clock creado con el modulo *clk_lento*. Cada muestra se guarda en uno de los 16 registro de los *ASR*, avanzando uno por uno en cada flanco de subida de este clock. Por otro lado, los módulos *up_counter* y *down_counter* recorren todos los registros en solo un ciclo del *clk_lento* para sacar sus valores en forma sincronizada

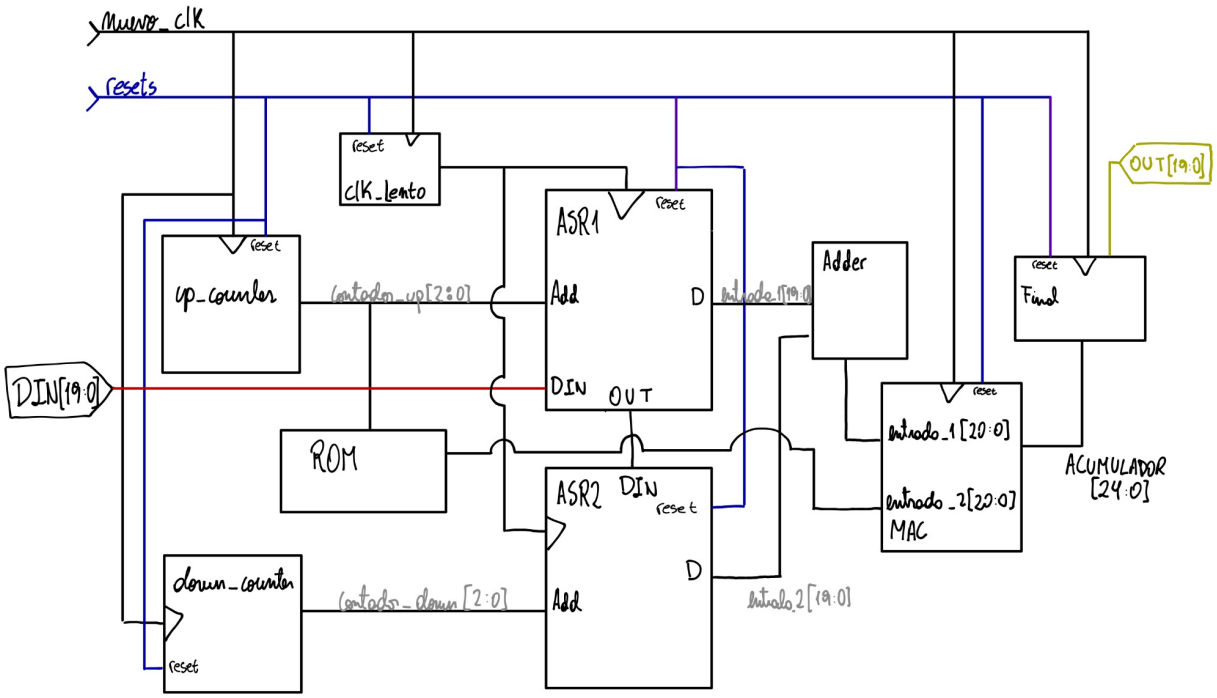

Figura 2: Diagrama de Bode.

(primero registro, con el ultimo registro, segundo registro con el penúltimo registro, etc.) y sean sumados en el modulo *Adder*. Este resultado sera multiplicado por uno de los coeficientes guardada en la *ROM* según corresponda, y se iran sumando hasta el ultimo registro de los *ASR*. Tambien, se considerarán los bits mas significativos en este proceso para que el modulo final entregue el resultado a la misma frecuencia con la cual entran los datos, es decir $48kHz$.

Anexo

```
`timescale 1ns/1ns
module test_filtro_fir;
        //Inputs
        reg [0: 0] nuevo_clk;
        reg [0: 0] resets;
        reg [19: 0] muestras_audio;

        //Outputs
        wire [19: 0] salida_filtro;

        //Instantiation of Unit Under Test
        filtro_fir uut (
                .nuevo_clk(nuevo_clk),
                .resets(resets),
                .muestras_audio(muestras_audio),
                .salida_filtro(salida_filtro)
        );

        initial begin
        //Inputs initialization
                nuevo_clk = 0;
                resets = 0;
                muestras_audio = 0;
        end

        initial begin
        repeat(4) #5 nuevo_clk = ~nuevo_clk;
        #5 nuevo_clk = ~nuevo_clk;
        resets = 1'b1;
        #5 nuevo_clk = ~nuevo_clk;
        resets = 1'b0;
        forever #5 nuevo_clk = ~nuevo_clk;
        end
        initial begin
        muestras_audio = 000_0000_0000_0000_0000; #80;
        muestras_audio = 1111_0000_0000_0000_0000; #100;
        muestras_audio = 0000_0000_0000_0000_0000; #150;
        end

endmodule
```