

DSED PROYECTO FINAL

Carlos Gómez Rodríguez y Alejandro Ramos Martín

9 de junio de 2020

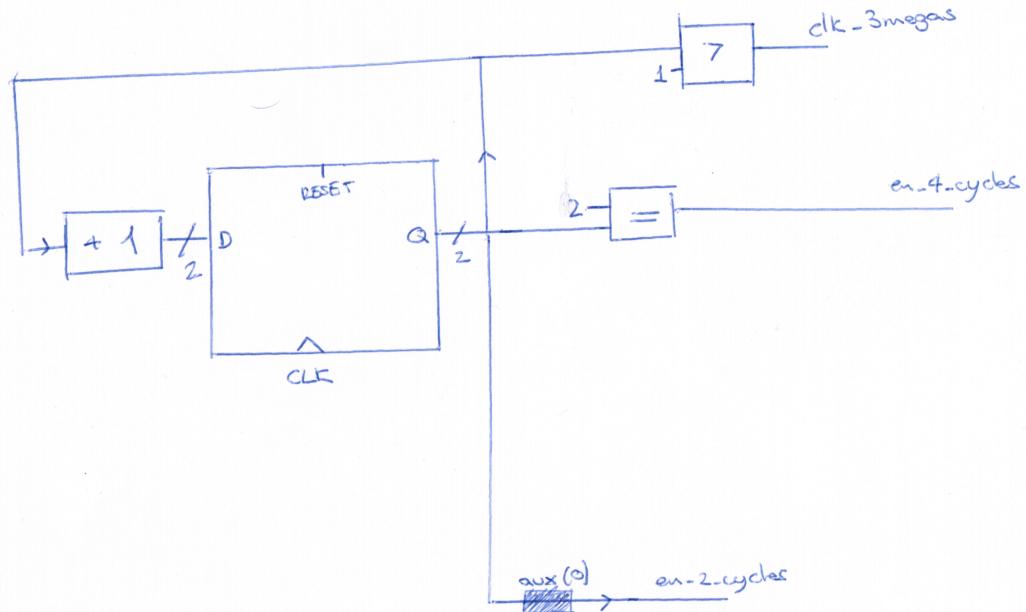
Índice

1. Bloque 1: Interfaz de audio	2
1.1. Tarea 1.1:	2
1.2. Tarea 1.2:	2
1.3. Tarea 1.3:	2
1.4. Tarea 1.4:	3
1.5. Tarea 1.5:	4
1.6. Tarea 1.6:	4
1.7. Tarea 1.7:	4
1.8. Tarea 1.8:	5
1.9. Tarea 1.9:	5
1.10. Tarea 1.10:	5
1.11. Tarea 1.11:	6
1.12. Tarea 1.12:	8
1.13. Tarea 1.13:	8
1.14. Tarea 1.14:	8
1.15. Tarea 1.15:	8
1.16. Tarea 1.16:	8
2. Bloque 2: Filtro FIR	9
2.1. Tarea 2.1:	9
2.2. Tarea 2.2:	10
2.3. Tarea 2.3:	13
2.4. Tarea 2.4:	14
2.5. Tarea 2.5:	14
2.6. Tarea 2.6:	14
2.7. Tarea 2.7:	15
2.8. Tarea 2.8:	15
2.9. Tarea 2.9:	15
2.10. Tarea 2.10:	16
2.11. Tarea 2.11:	17
2.12. Tarea 2.12:	17
2.13. Tarea 2.13:	17
2.14. Tarea 2.14:	17
3. Bloque 3: Controlador y Memoria	18
3.1. Tarea 3.1:	18
3.2. Tarea 3.2:	19
3.3. Tarea 3.3:	19
3.4. Tarea 3.4:	19
3.5. Tarea 3.5:	22
3.6. Tarea 3.6:	22
3.7. Mejora Display:	22
3.8. Mejora Volumen:	22

1. Bloque 1: Interfaz de audio

1.1. Tarea 1.1:

Diseña un circuito que sea capaz de generar las señales especificadas. Dibuja el esquemático correspondiente a tu diseño en el espacio inferior.

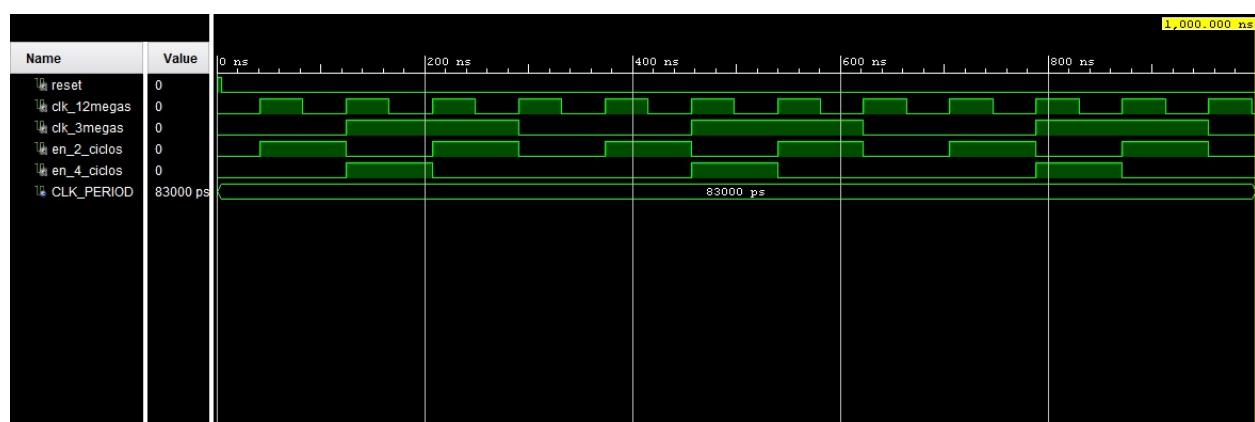


1.2. Tarea 1.2:

Implementa tu diseño en VHDL.

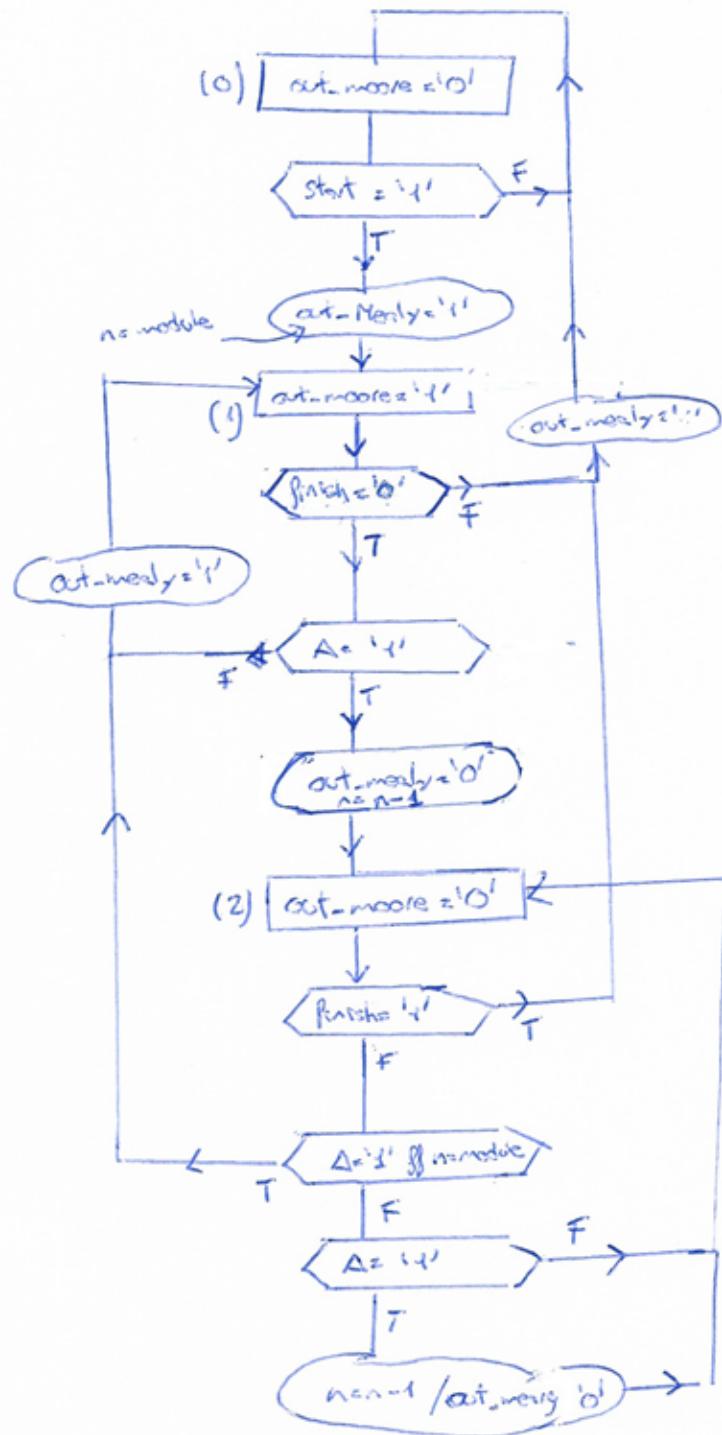
1.3. Tarea 1.3:

Genera un testbench que verifique la funcionalidad de tu diseño.



1.4. Tarea 1.4:

Realiza el diagrama ASMD que describa la misma funcionalidad que el pseudo-código anterior.



1.5. Tarea 1.5:

Implementa el diagrama anterior en un fichero VHDL.

1.6. Tarea 1.6:

Realiza un testbench que tenga la señal micro data fija a '1' para comprobar que las transiciones de estados, la selección de los datos de salida y la activación de sample_out_ready se produce correctamente.



1.7. Tarea 1.7:

Realiza un testbench que introduzca una señal pseudo-aleatoria en micro data y comprueba que la digitalización se realiza correctamente.

La señal pseudoaleatoria introducida es la siguiente:

```
wait for 4000 ns;
micro_data <= '1';
wait for 1000 ns;
micro_data <= '0';
```

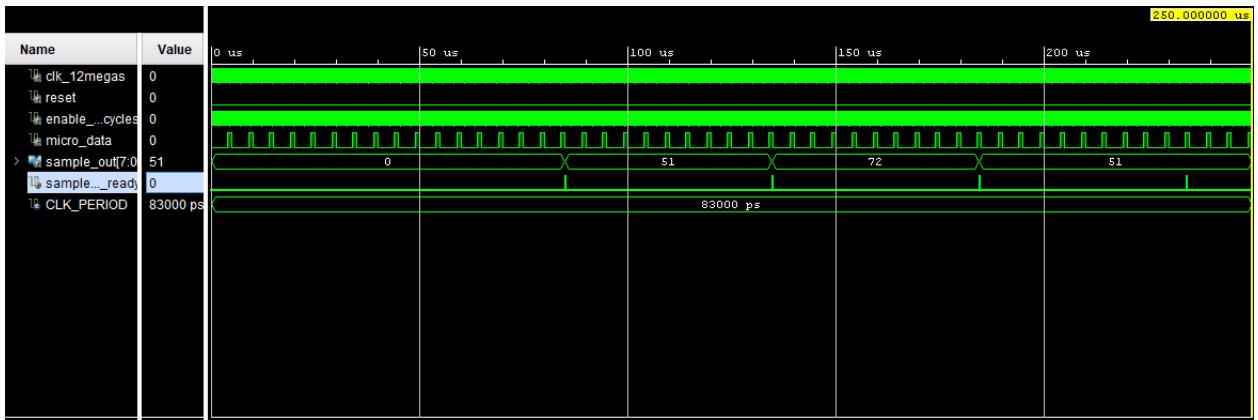
Esta secuencia se repite cada 5 us. La primera muestra se obtiene tras 255 ciclos del reloj de 3 MHz, esto es, tras 85 us. Por lo tanto, estos 255 ciclos equivaldrán a 17 repeticiones de la secuencia pseudoaleatoria de la señal de entrada. Cada repetición contiene 3 '1's, por lo que esta primera muestra de salida del micrófono tendrá un valor de:

$$sample_out = 3 \cdot 17 = 51$$

El siguiente dato será la cuenta almacenada en dato2 desde el inicio. Puesto que la primera etapa no se saca el dato, esto ocurrirá tras $105+150+105=360$ ciclos (120 us). Estos ciclos equivaldrán a 24 repeticiones de la secuencia de datos. En consecuencia, el valor de la segunda muestra de salida del micrófono será:

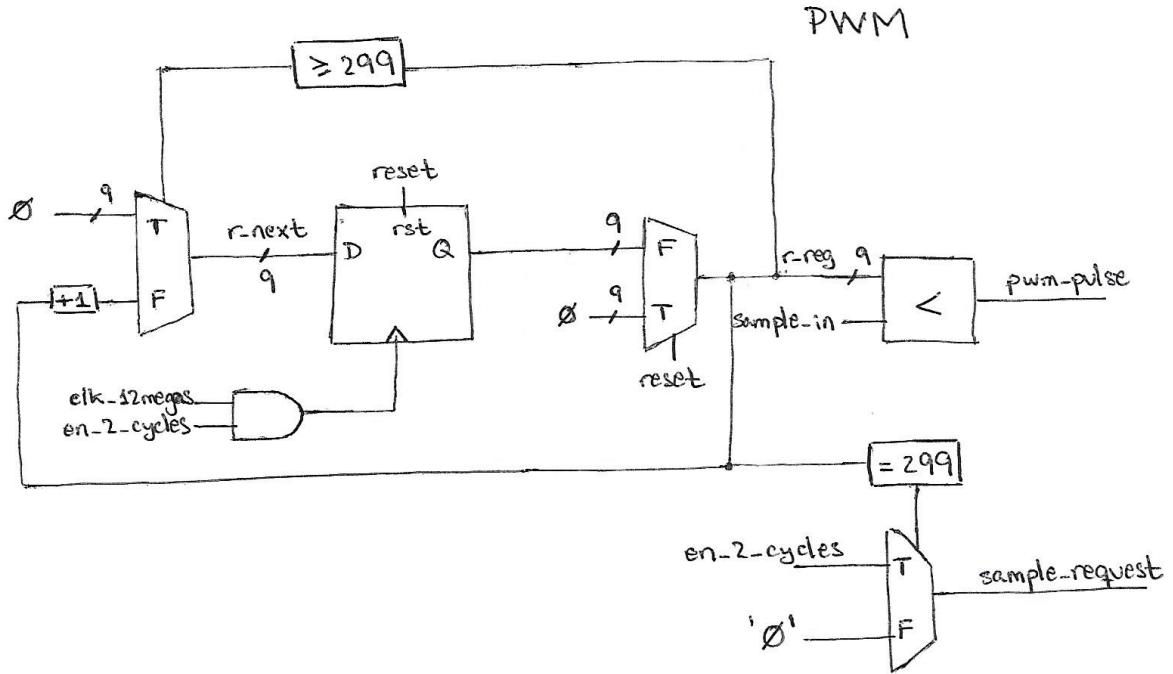
$$sample_out = 3 \cdot 24 = 72$$

El siguiente dato será 51, el siguiente 72... y así recursivamente.



1.8. Tarea 1.8:

Modifica el diseño propuesto en el libro del Dr. Chu para que sea compatible con las especificaciones de nuestro sistema. Es decir, que el contador cuente de 0 a 299 e incluya reset y enable y que el circuito produzca la salida sample request en el momento apropiado y con la duración apropiada. Dibuja el esquemático de tu diseño en la parte inferior.

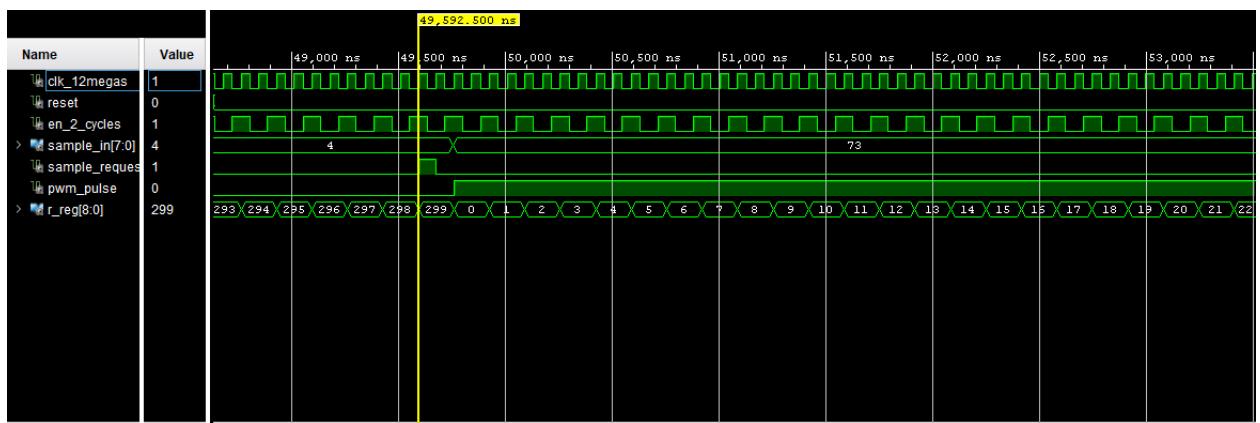
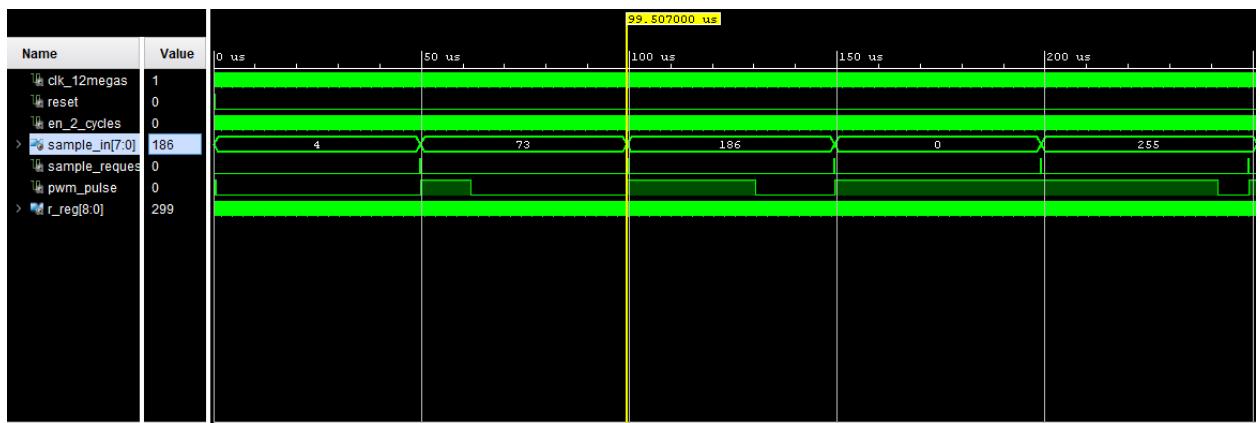
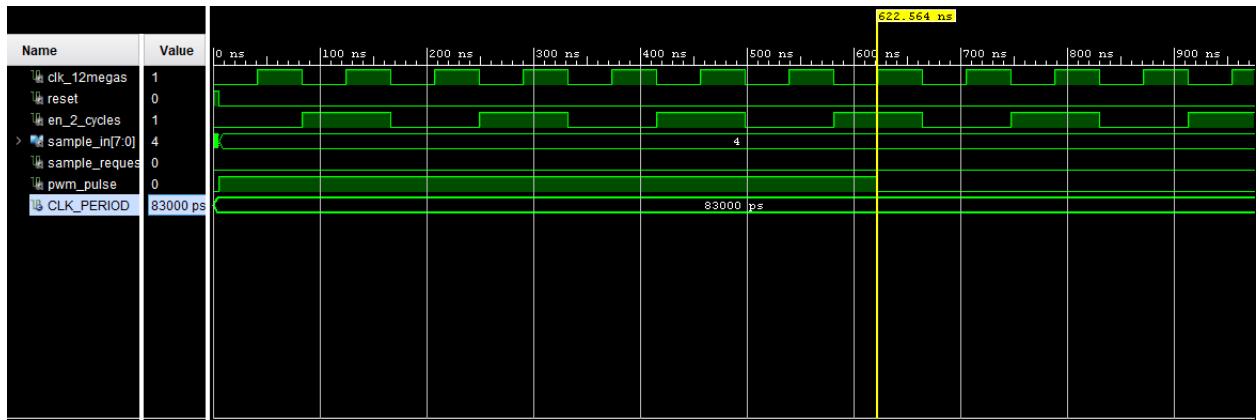


1.9. Tarea 1.9:

Implementa el esquemático anterior en un fichero VHDL.

1.10. Tarea 1.10:

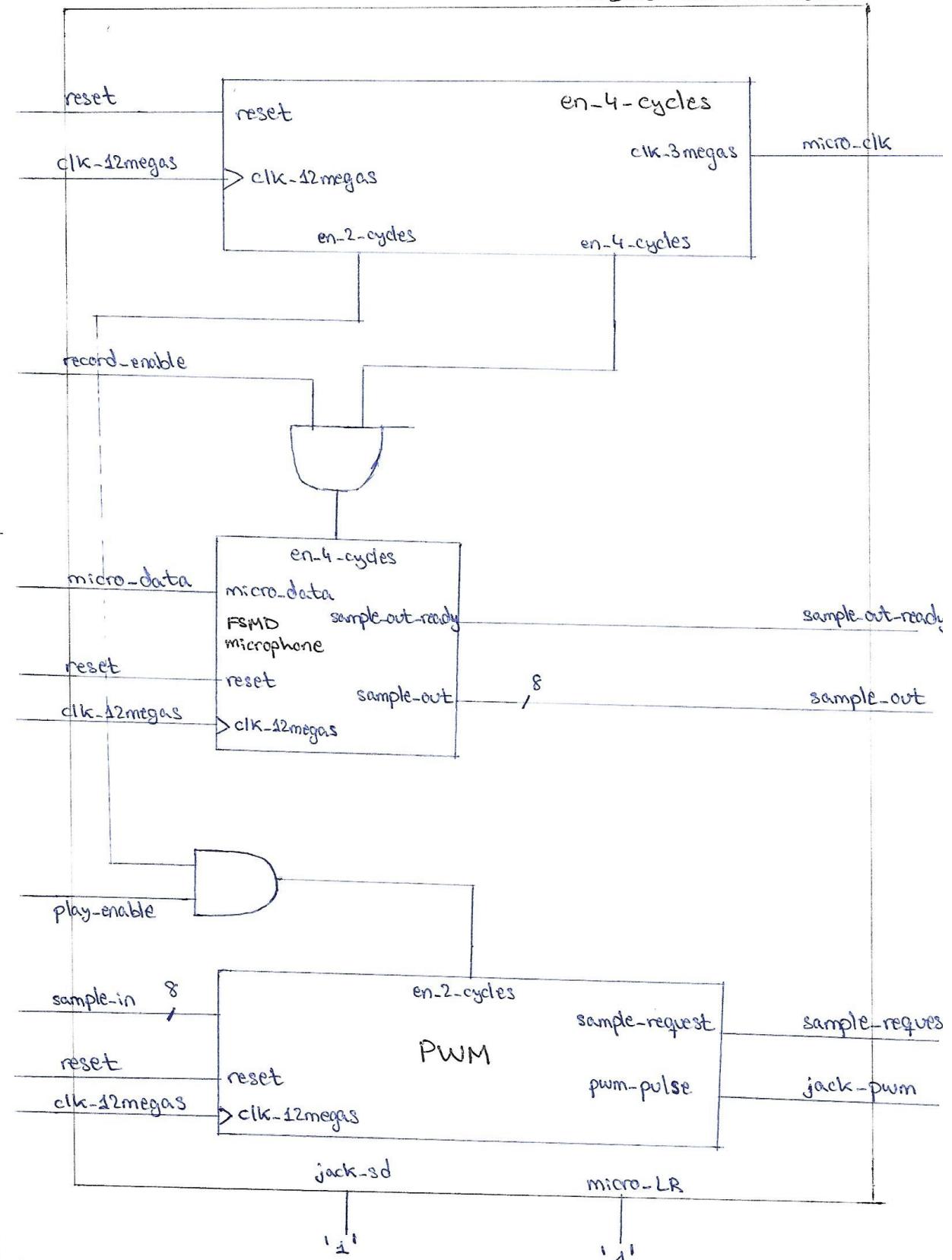
Realiza un testbench que verifique el funcionamiento de la interfaz de la salida de audio.



1.11. Tarea 1.11:

Dibuja el esquemático a nivel bloque de la interfaz de audio completa. Ten en cuenta que record enable y play enable especifican cuándo están activas las interfaces del micrófono y de la salida de audio, respectivamente. Del mismo modo, tienes que asignar un '1' tanto a micro LR, como a jack_sd.

AUDIO INTERFACE



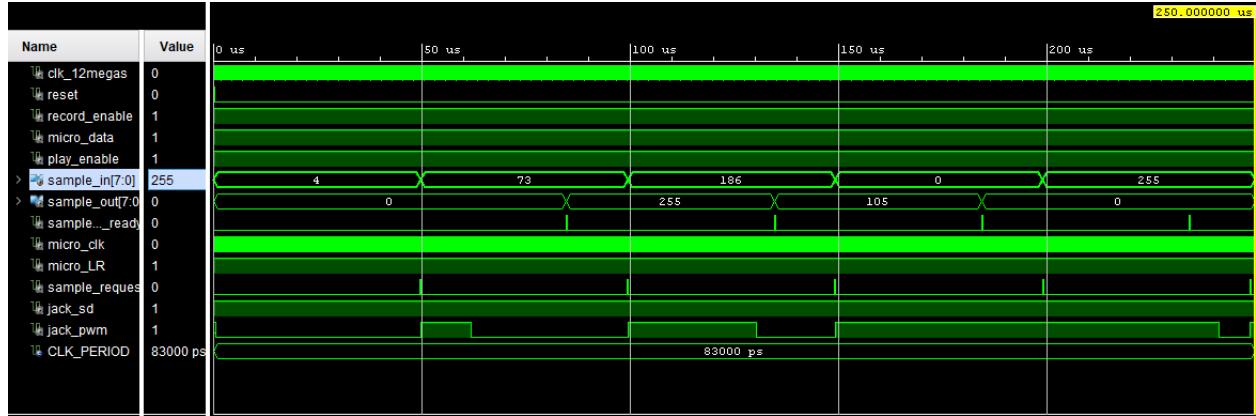
1.12. Tarea 1.12:

Implementa en VHDL la interfaz de audio completa. Para ello utiliza un estilo de código estructural que instancia cada uno de los tres bloques desarrollados anteriormente.

1.13. Tarea 1.13:

Diseña un testbench que verifique la funcionalidad de la interfaz de audio completa.

Con el fin de simplificar la simulación se ha introducido una señal de datos continua: micro_data <= '1', y los resultados son los esperados si se comparan con las simulaciones independientes de los bloques.

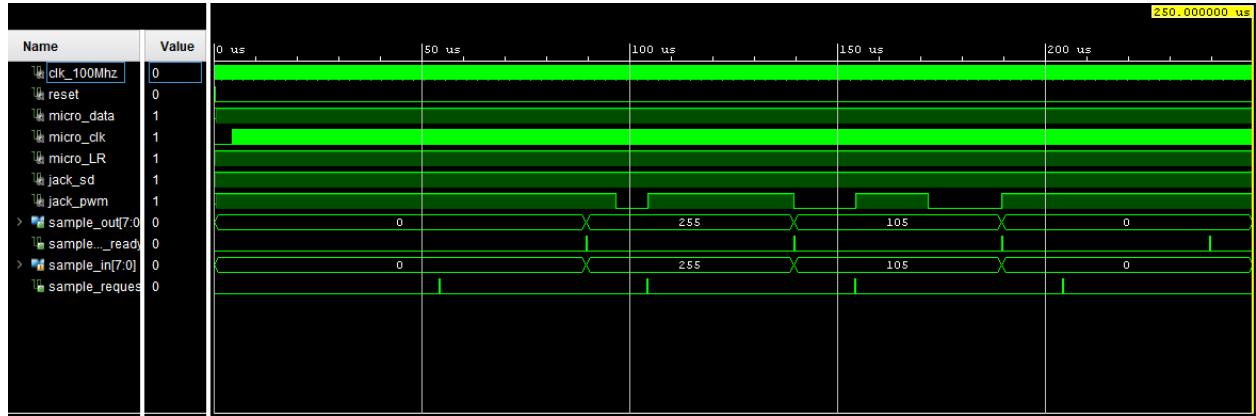


1.14. Tarea 1.14:

Implementa en VHDL el controlador. Utiliza estilo estructural. Necesitas emplear el asistente arquitectural de Vivado para generar el reloj de 12 MHz a partir del reloj de 100 MHz.

1.15. Tarea 1.15:

Diseña un testbench que verifique la funcionalidad del controlador. Puedes utilizar los estímulos empleados en testbenches anteriores.



1.16. Tarea 1.16:

Escribe el fichero de restricciones .xdc. Sintetiza y realiza la implementación física del diseño. Genera el fichero .bit y programa la FPGA. Comprueba el funcionamiento correcto conectando unos auriculares a la salida de audio de la placa. Si la salida de audio es muy ruidosa, puedes jugar con el valor de micro_LR y ponerlo a '0' en lugar de a '1'. Guarda en un fichero de texto todos los mensajes de warning que hayan aparecido en el proceso anterior. Avisa al profesor para comprobar el funcionamiento.

2. Bloque 2: Filtro FIR

2.1. Tarea 2.1:

¿Cuál es el rango de un número de 8 bits en punto fijo <1,7> en complemento a dos? ¿Cuál es su precisión?

Tarea 2.1

S	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
ds	0.25	0.125	0.0625	0.03125	0.015625	0.0078125	

Precisión: 10'0078125

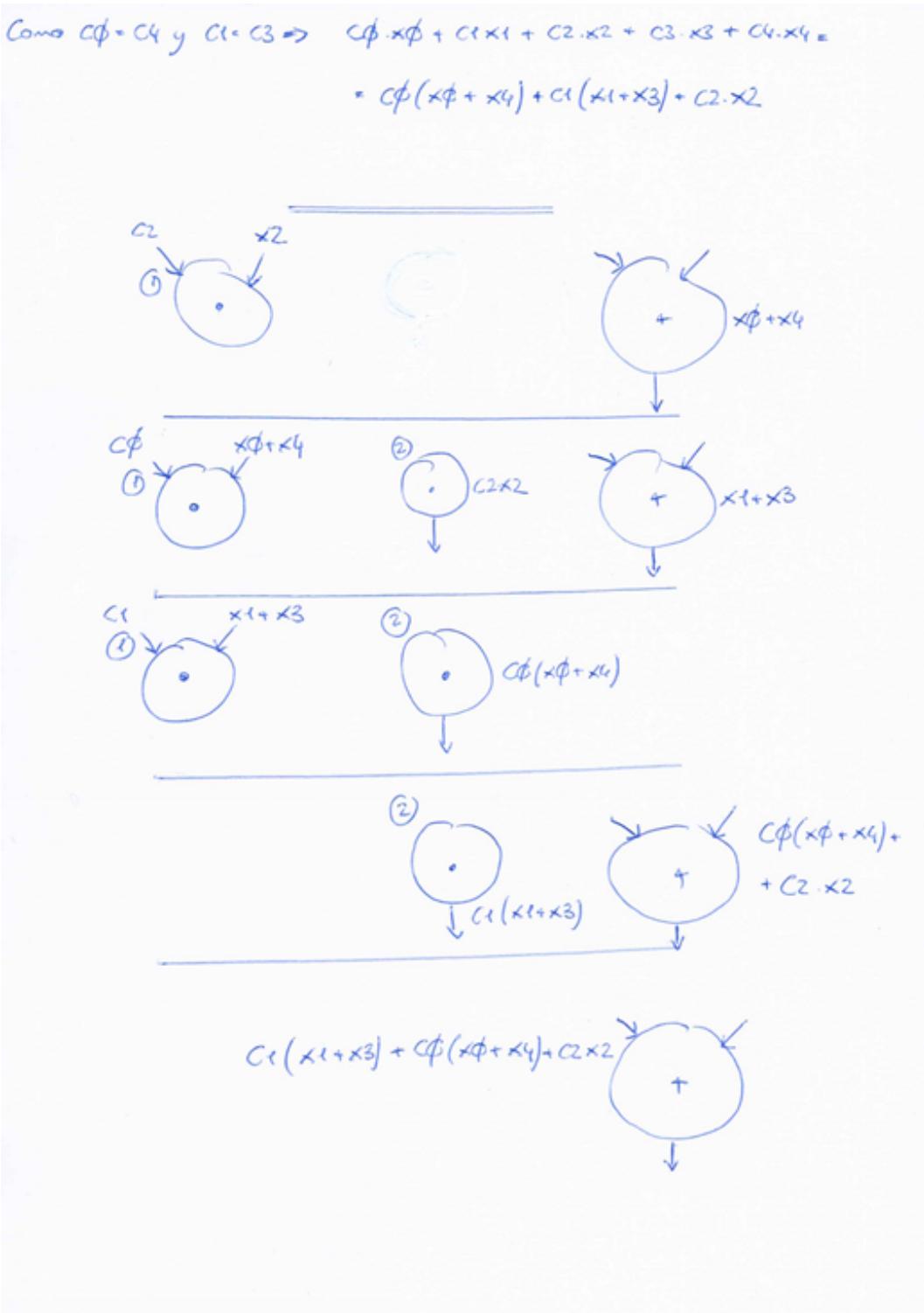
$$\text{Máx: } "0111111" = 0.9921875_{10}$$

$$\text{Min: } "10000000" = -1_{10}$$

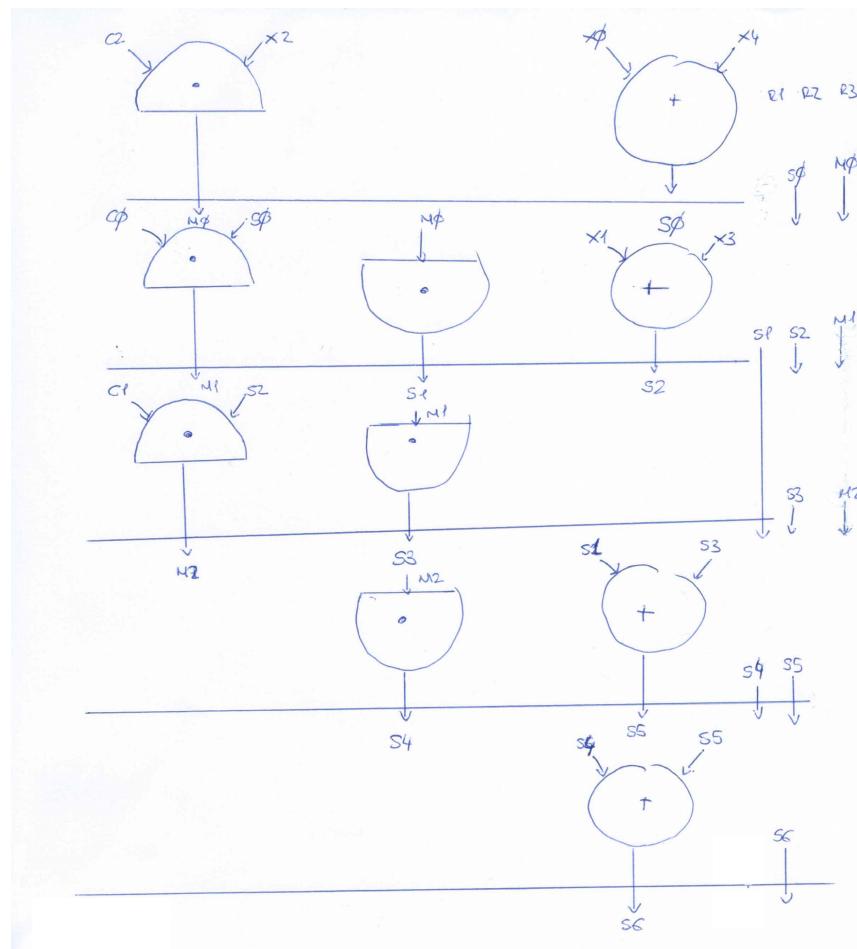
2.2. Tarea 2.2:

Realiza sobre papel la implementación de un filtro FIR de 5 etapas con la siguiente asignación: Dos medios multiplicadores y un sumador. Realiza todos los pasos descritos en la sección 6.4.

Planificación temporal

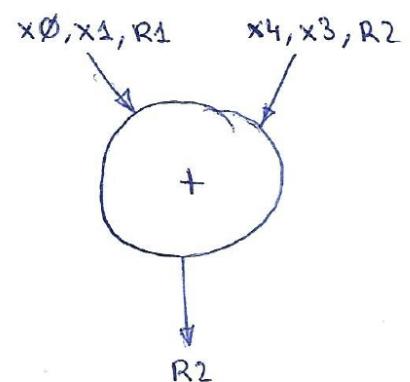
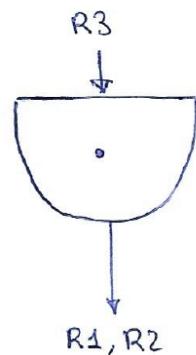
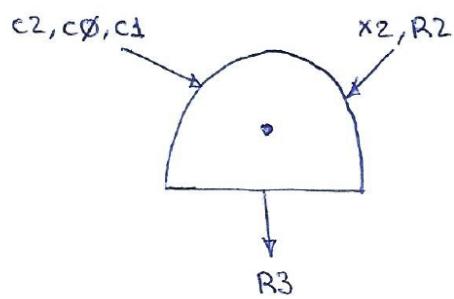


Vinculación, análisis de tiempo de vida de variables y asignación de registros:



Análisis de conexiones para la extracción de multiplexores:

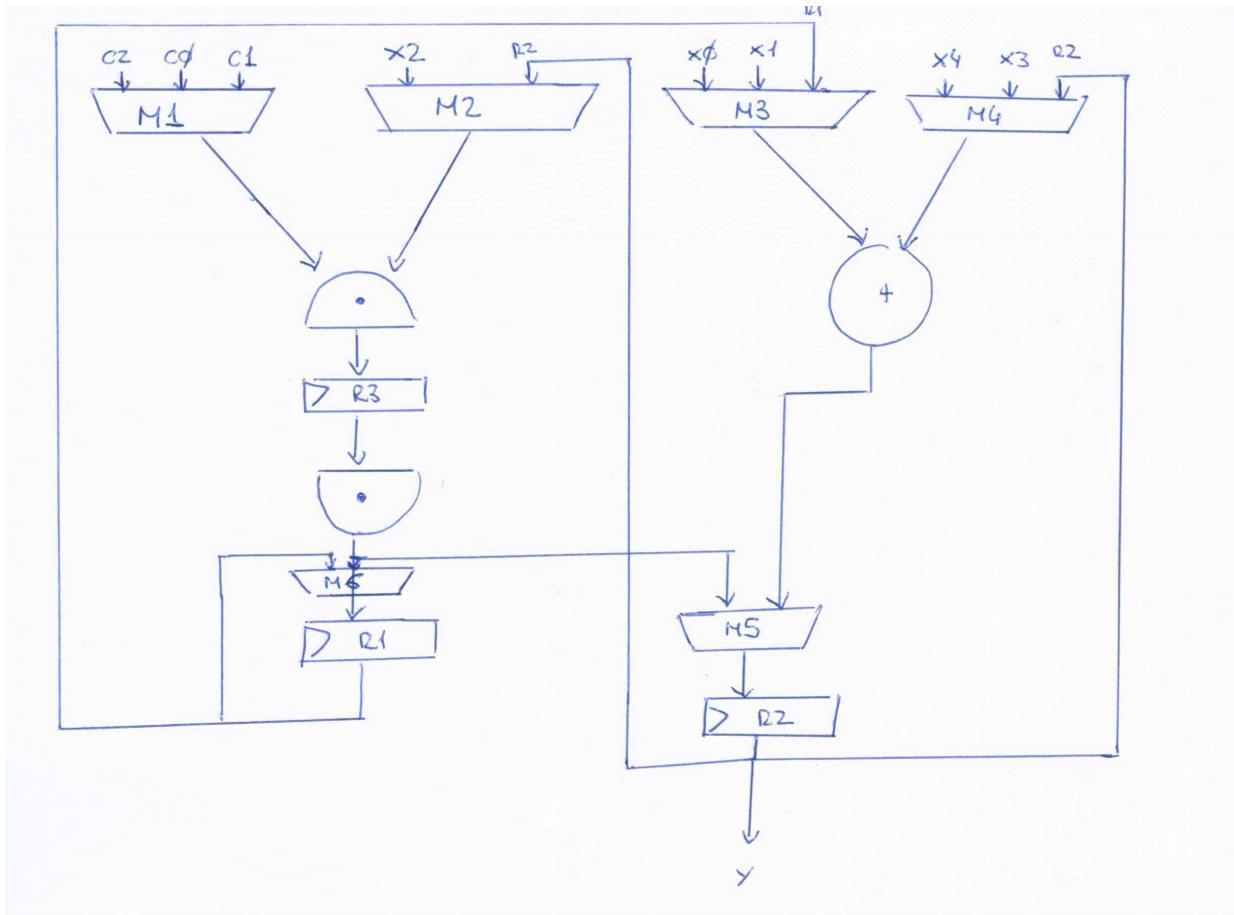
Conexiones:



Cronograma:

	T1	T2	T3	T4	T5	T6
M1	C2 (00)	C0 (01)	C1 (10)			
M2	X2 (0)	S0 (1)	S2 (1)			
M3	X0 (00)	X1 (01)		S1(10)	S4 (10)	
M4	X4 (00)	X3 (01)		S3 (10)	S5 (10)	
M5	S0 (1)	S2 (1)	S3 (0)	S5 (1)	S6 (1)	
M6		S1 (1)	S1 (0)	S4 (1)		
R1			S1	S1	S4	
R2		S0	S2	S3	S5	S6
R3		M0	M1	M2		

	T1 000	T2 001	T3 010	T4 011	T5 100
M1	C2	C0	C1		
M2	X2	R2	R2		
M3	X0	X1		R1	R1
M4	X4	X3		R2	R2
M5	sum_out	sum_out	mul_out	sum_out	sum_out
M6		mul_out	R1	mul_out	



Implementación:

2.3. Tarea 2.3:

Realiza un análisis de cuantificación de las señales, de manera que especifiques cuántos bits vas a emplear en cada señal y en qué posición va a estar el punto decimal. Emplea la notación y las metodologías presentadas en clase.

Partimos de que las muestras de entrada sample_in y las de salida sample_out tienen 8 bits cada una y utilizan una notación <1,7>.

El menor de los coeficientes positivos del filtro tiene un valor:

$$c_0 = c_4 = +0,039$$

que necesitará una precisión de 7 bits decimales. El menor de los coeficientes negativos del filtro tiene un valor:

$$c_0 = c_4 = -0,0078$$

que necesitará también una precisión de 7 bits decimales. Añadiendo el bit de signo, representaremos todos los coeficientes con 8 bits y utilizando una notación <1,7>.

Las señales de entrada al medio multiplicador mul_in0 y mul_in1 utilizarán una precisión de 8 bits y una notación <1,7>. La salida del multiplicador mul_out será en consecuencia de la forma <2,14>.

Las señales de entrada al sumador sum_in0 y sum_in1 utilizarán una precisión de 16 bits, con una representación <2,14>, ya que se introduce a su entrada la señal de salida del multiplicador en varios ciclos. Se ha elegido la misma

representación <2,14> para la salida sum_out.

Los registros R1, R2 y R3 almacenarán temporalmente las salidas del sumador y los medios multiplicadores, por lo que requieren una precisión de 16 bits, y se ha elegido para ello una notación <2,14>.

2.4. Tarea 2.4:

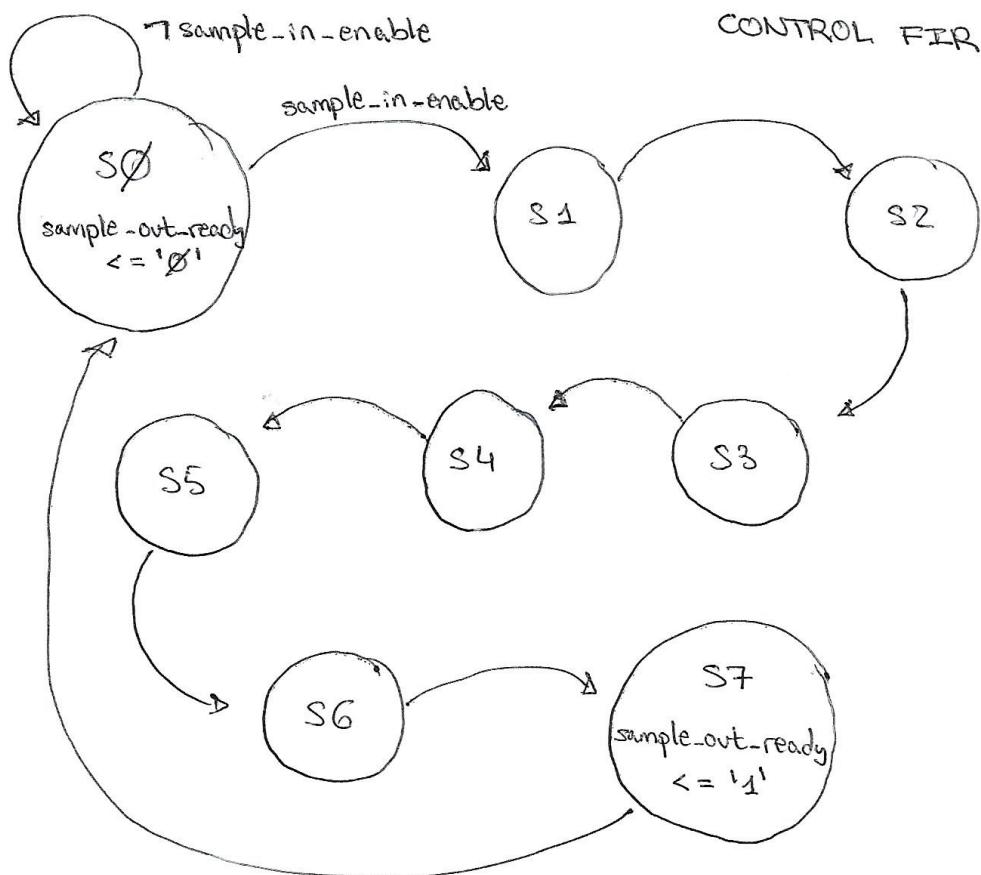
Escribe modelos VHDL para todos los componentes que se emplean en tu ruta de datos. Los operadores aritméticos (suma y multiplicación) no necesitan describirse en un estilo estructural. Para conseguir el medio multiplicador, añade un registro extra a la salida de un multiplicador completo, de esta forma se comportará como un multiplicador perfectamente segmentado. La herramienta de síntesis se encargará de mover este registro al interior del multiplicador de manera óptima. Comprueba el funcionamiento correcto de tus componentes creando los correspondientes testbenches.

2.5. Tarea 2.5:

Escribe un modelo estructural de la ruta de datos completa en VHDL. Crea un testbench para la ruta de datos.

2.6. Tarea 2.6:

Dibuja el diagrama de estados que describe el controlador de tu filtro FIR. Sigue la metodología de implementación de máquinas de estados finitos vista en clase.



2.7. Tarea 2.7:

Escribe el código VHDL correspondiente a tu controlador.

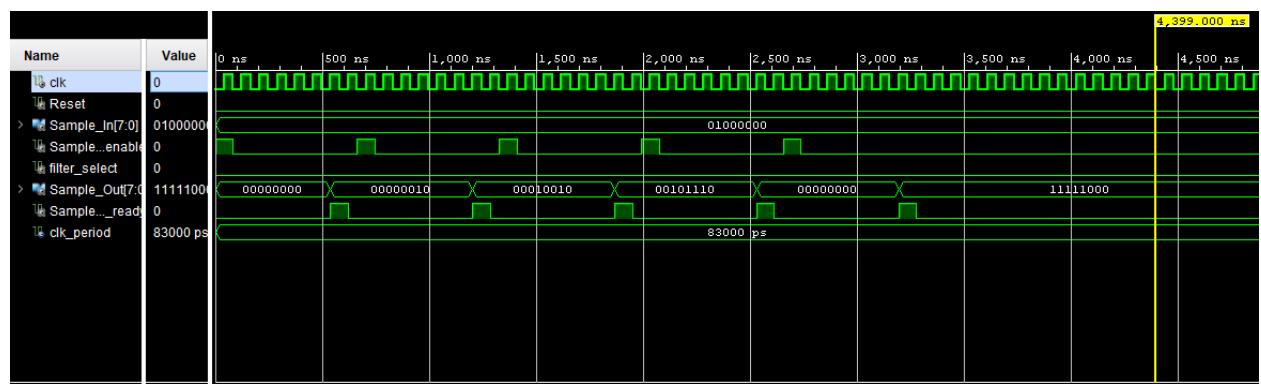
2.8. Tarea 2.8:

Escribe el código VHDL estructural correspondiente a tu filtro completo.

2.9. Tarea 2.9:

Escribe el código VHDL correspondiente a un testbench que introduzca un único impulso a la entrada de valor +0.5. ¿Qué secuencia esperas en Sample_Out si en Sample_In la secuencia es (0, 0, 0, 0, X, 0, 0, 0, 0)? Considera que X es el mayor/menor número positivo/negativo (cuatro casos) que se puede representar. Ten en cuenta la cuantificación de tus señales. Asegúrate de que tu diseño proporciona los valores esperados.

Si la simulación se realiza para un filtro FIR paso bajo, el resultado obtenido, para una muestra de entrada de valor +0.5, es el mostrado:



En el caso planteado, la secuencia a la salida del filtro quedaría como:

1)

$$sample_in = 0 : sample_out = 0$$

2)

$$sample_in = 0 : sample_out = 0$$

3)

$$sample_in = 0 : sample_out = 0$$

4)

$$sample_in = 0 : sample_out = 0$$

5)

$$sample_in = X : sample_out = c_0 \cdot X$$

6)

$$sample_in = 0 : sample_out = c_1 \cdot X$$

7)

$$sample_in = 0 : sample_out = c_2 \cdot X$$

8)

$$sample_in = 0 : sample_out = c_3 \cdot X$$

9)

$$sample_in = 0 : sample_out = c_4 \cdot X$$

Números positivos:

$$Mayor : X = 127/128 = 0,9921875$$

$$Menor : X = 1/128 = 0,0078125$$

Números negativos:

$$Mayor : X = -128/128 = -1$$

$$Menor : X = -1/128 = -0,007812$$

2.10. Tarea 2.10:

Escribe el código VHDL correspondiente a un testbench que introduzca en la entrada la siguiente secuencia (0, 0.5, 0, 0.125, 0, 0, 0, 0, ...). ¿Qué secuencia esperas en Sample_Out para esta secuencia de entrada? Asegúrate de que tu diseño proporciona los valores esperados.

La salida puede escribirse como:

$$sample_out = c_0 \cdot x_0 + c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3 + c_4 \cdot x_4$$

En los 8 ciclos de simulación se obtendrá el resultado mostrado a continuación:

1)

$$x_0 = 0 : sample_out = 0$$

2)

$$x_0 = 0,5, x_1 = 0 : sample_out = 0,5 \cdot c_0$$

3)

$$x_0 = 0, x_1 = 0,5 : sample_out = 0,5 \cdot c_1$$

4)

$$x_0 = 0,125, x_1 = 0, x_2 = 0,5, x_3 = 0 : sample_out = 0,125 \cdot c_0 + 0,5 \cdot c_2$$

5)

$$x_0 = 0, x_1 = 0,125, x_2 = 0, x_3 = 0,5, x_4 = 0 : sample_out = 0,125 \cdot c_1 + 0,5 \cdot c_3$$

6)

$$x_0 = 0, x_1 = 0, x_2 = 0,125, x_3 = 0, x_4 = 0,5 : sample_out = 0,125 \cdot c_2 + 0,5 \cdot c_4$$

7)

$$x_0 = 0, x_1 = 0, x_2 = 0, x_3 = 0,125, x_4 = 0 : sample_out = 0,125 \cdot c_3$$

8)

$$x_0 = 0, x_1 = 0, x_2 = 0, x_3 = 0,5, x_4 = 0,125 : sample_out = 0,125 \cdot c_4$$

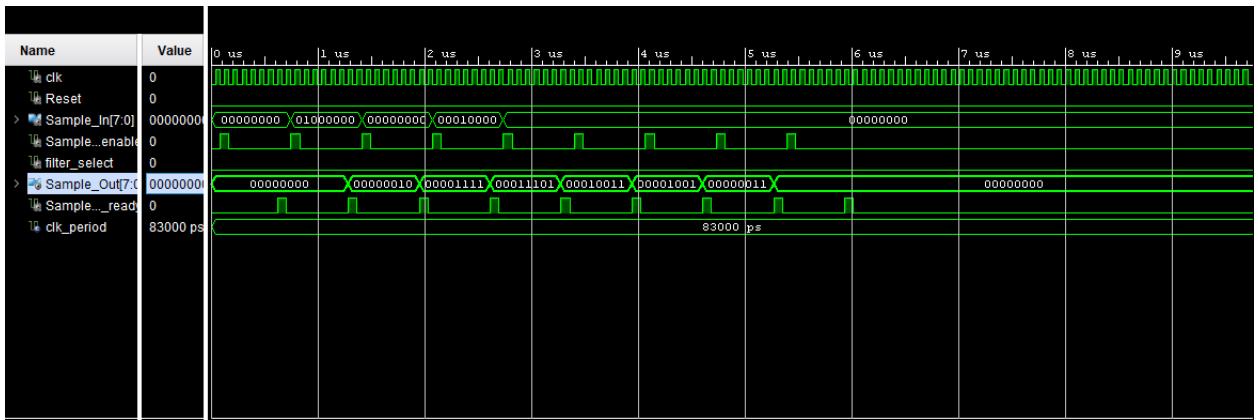
La simulación se ha realizado para un filtro paso bajo. La secuencia obtenida tras sustituir por los valores de dichos coeficientes es la siguiente:

$$sample_out = \{0, 0,0195, 0,1211, 0,2275, 0,1514, 0,0752, 0,0303, 0,0049\}$$

que en formato <1,7>en complemento a 2, y redondeando a lo bajo, resulta ser:

$$sample_out = \{00000000, 00000010, 00001111, 00011101, 00010011, 00001001, 00000011, 00000000\}$$

El resultado de la simulación coincide con el esperado, como se muestra a continuación:



2.11. Tarea 2.11:

Escribe el código VHDL correspondiente a un testbench que lea las muestras de un fichero y escriba los resultados en otro. Llama al fichero que contiene los datos de entrada “sample_in.dat”; llama al fichero de salida “sample_out.dat”. Rellena el fichero de entrada con una secuencia que introduzca un único impulso. Cada línea del fichero se corresponde con un dato.

2.12. Tarea 2.12:

Utiliza el fichero “haha.wav” en Matlab para crear un fichero de entrada para tu testbench. En el apéndice 2 se detallan las secuencias que tienes que emplear en Matlab para: Cargar un fichero de audio en Matlab y crear otro fichero con el formato necesario para tu testbench. Utilizar la función filter de Matlab para obtener la respuesta de un filtro FIR con precisión real. Cargar y escuchar la salida que ha producido tu testbench en Matlab.

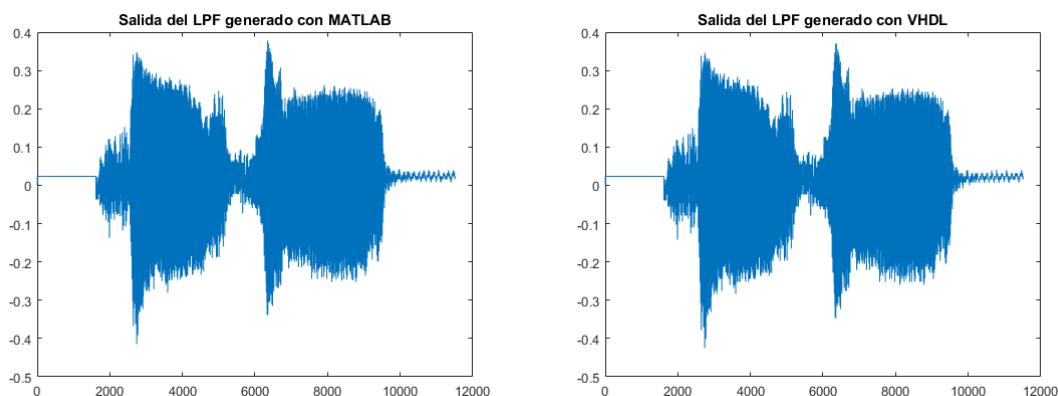
2.13. Tarea 2.13:

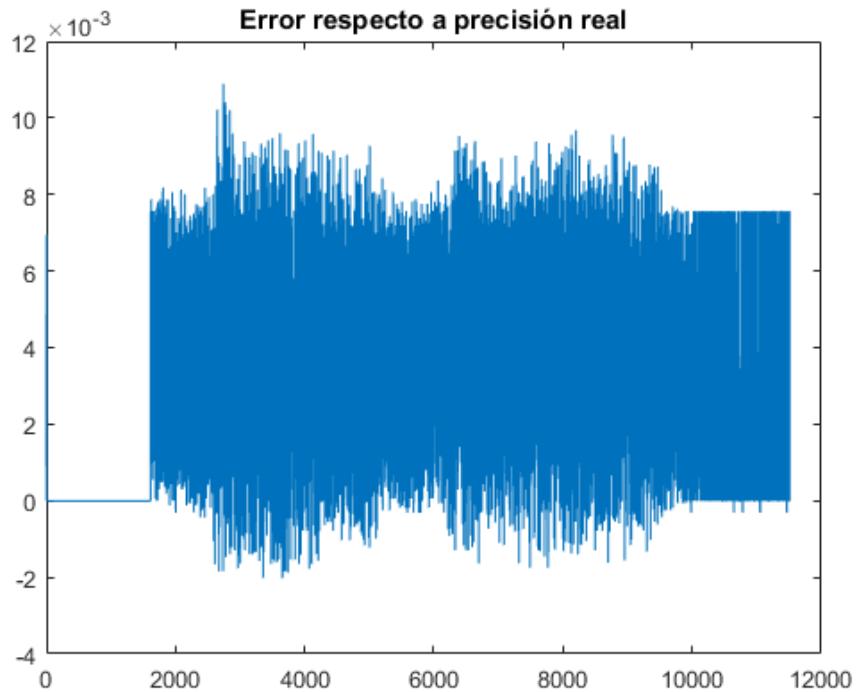
Emplea tu testbench para procesar el fichero de entrada que te ha proporcionado Matlab. Escribe los datos filtrados en el fichero “sample_out.dat”, que importarás posteriormente en Matlab.

2.14. Tarea 2.14:

Importa tu fichero “sample_out.dat” en Matlab. Compara los resultados del testbench con los valores con precisión real proporcionados por la función filter de Matlab. Haz un gráfico del error de tus resultados (resta tus resultados de los datos con precisión real). Utiliza la función sound de Matlab para escuchar la forma de onda original. Compárala después con tu sonido filtrado y observa si hay alguna diferencia entre el filtro con precisión real y tu filtro.

A continuación se muestran las gráficas con la salida del filtro generado con MATLAB y VHDL respectivamente, así como el error entre ambas señales:





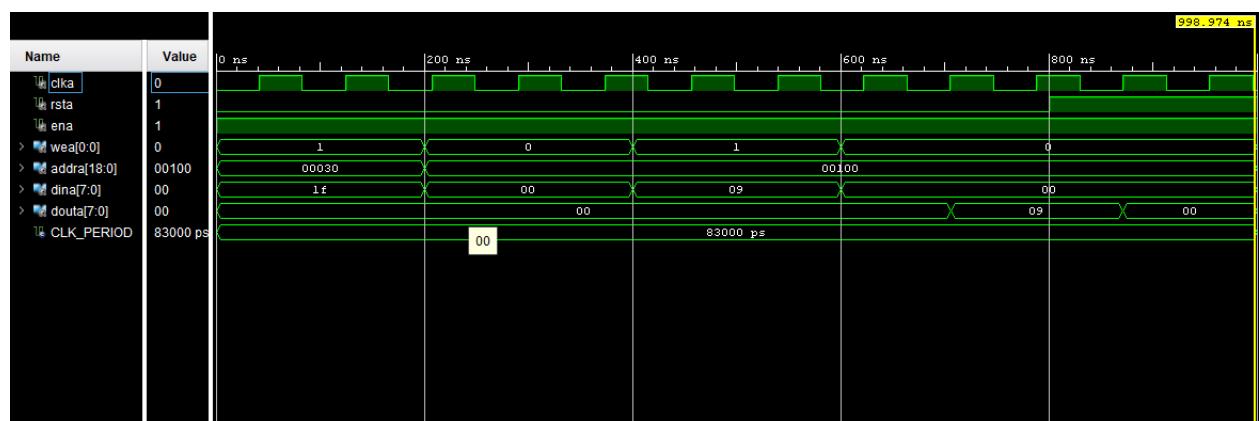
3. Bloque 3: Controlador y Memoria

3.1. Tarea 3.1:

Tarea 3.1: Crea un testbench que trabaje a la frecuencia del sistema para comprobar y entender el funcionamiento de la memoria encapsulada. Anota a continuación la función de cada puerto y la temporización de la memoria (un pequeño cronograma que incluya una escritura y una lectura).

Se ha habilitado el puerto A para el uso de la memoria RAM. El puerto clka es la entrada de la señal de reloj del sistema, rsta será el reset del módulo y ena la señal de enable que activará, si está a nivel alto, el sistema. wea es la entrada de otra señal enable, que activará la escritura del dato de entrada dina en la dirección establecida por addra, si está a nivel alto, o la lectura de este registro si está a nivel bajo. En este último caso el dato leído se obtendrá en douta.

A continuación se incluye una captura de la simulación obtenida para unos datos de entrada de prueba, mostrando un ejemplo de lectura y escritura en la memoria.



3.2. Tarea 3.2:

Determina cuánto tiempo de grabación va a poder estar almacenado en la memoria.

El número de posiciones de la memoria es de 524288. En cada una de estas posiciones se almacena una muestra. Lo que hace que la memoria como máximo pueda almacenar 524288 muestras. En el apartado 1.2, se puede ver que se almacena una muestra cada 150 ciclos de reloj con un reloj de 3 MHz.

$$Tiempo\ Total = (Posiciones\ en\ memoria) \cdot 150 \frac{\text{ciclos}}{\text{muestra}} \cdot \frac{1}{f_{\text{sistema}}} \frac{\text{s}}{\text{ciclo}} \quad Tiempo\ Total = 26,2\ s$$

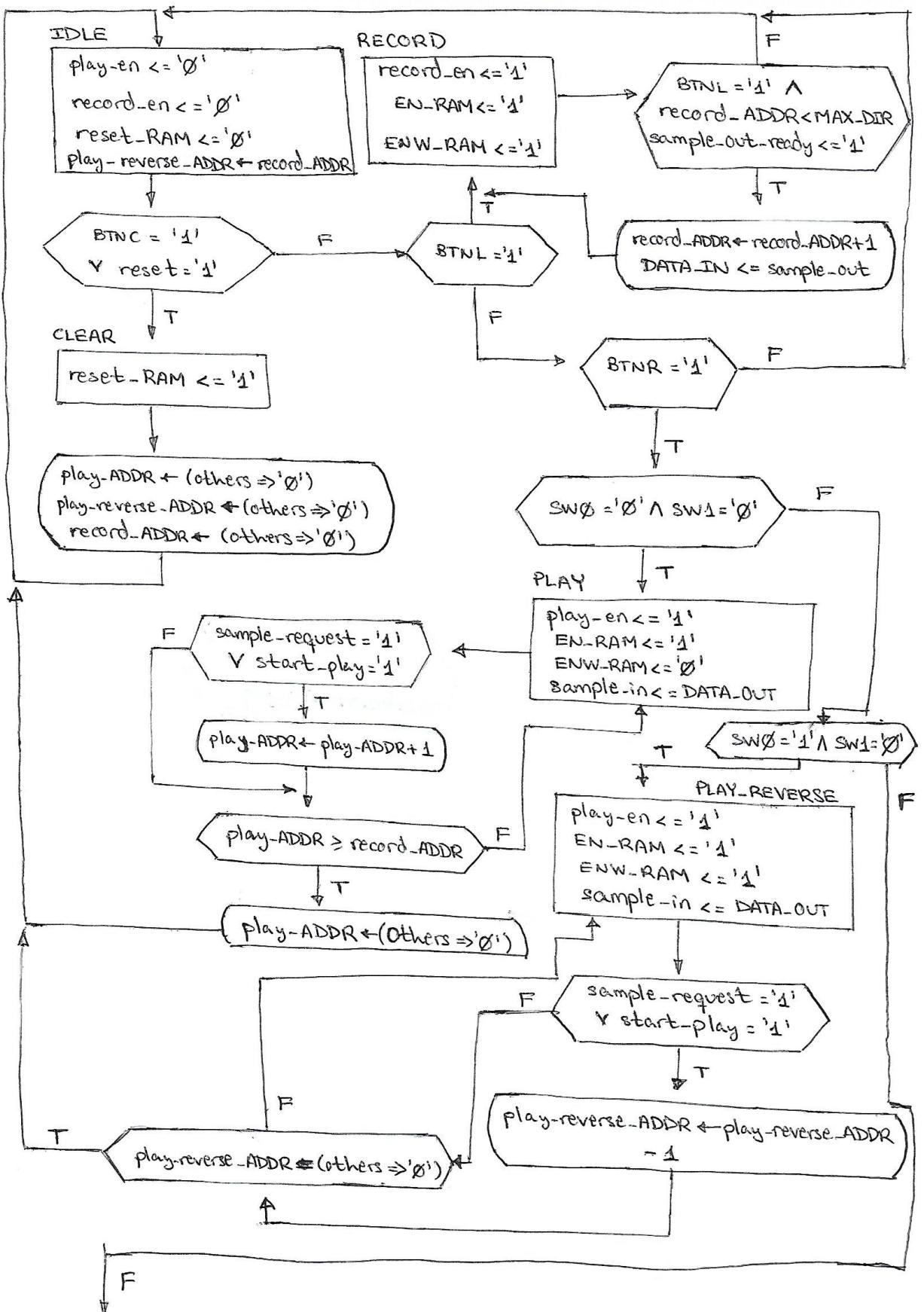
3.3. Tarea 3.3:

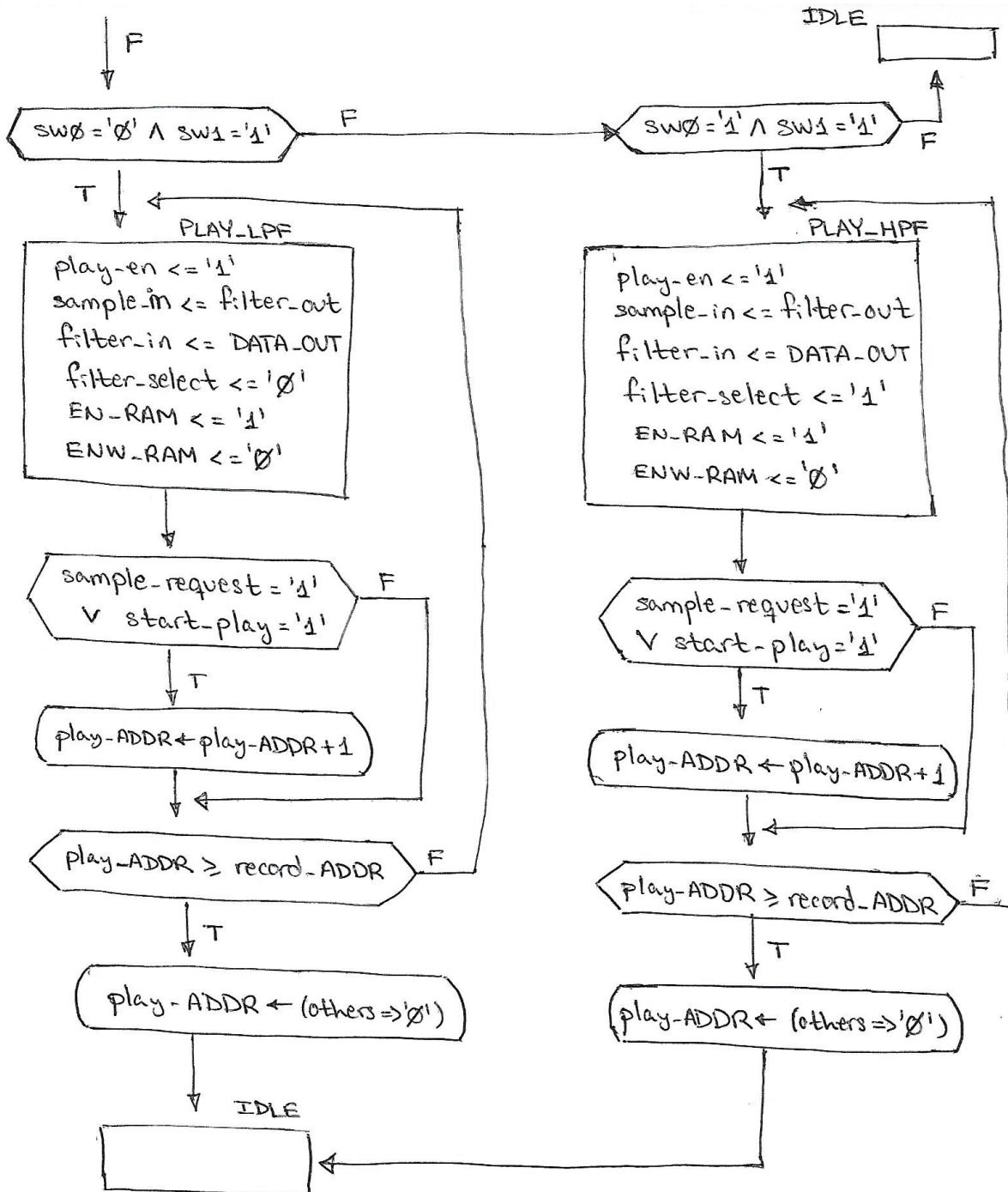
Determina qué operación es necesaria para hacer la transformación de las muestras de binaria a complemento a dos y de complemento a dos a binaria.

La operación a llevar a cabo para la conversión (tanto en complemento a dos → muestra como en muestra → complemento a dos) es cambiar el bit más significativo, $a(7) = \text{not}(a(7))$ y dejar el resto de bits (6 downto 0) igual.

3.4. Tarea 3.4:

Añade a continuación todas las hojas necesarias para describir tu diseño y la planificación para llevarlo a cabo. Puedes incluir diagramas esquemáticos, cronogramas explicativos, un plan de pruebas, una planificación temporal y todo lo que consideres necesario para explicar las decisiones que has tomado. Avisa al profesor antes de comenzar con el diseño para obtener el visto bueno.





3.5. Tarea 3.5:

Avisa al profesor cuando tengas todas las especificaciones mínimas del sistema global cubiertas.

3.6. Tarea 3.6:

Sube al Moodle un fichero .vhd con todas las fuentes de tu proyecto.

3.7. Mejora Display:

En esta mejora se ha usado el display de 7 segmentos para mostrar los segundos que quedan por reproducir o que se puede seguir grabando hasta que la memoria se llene: Para llevar a cabo la mejora se ha dividido en 2 partes:

Por una parte, se tiene que calcular el número de posiciones que quedan en memoria. Esto se hace mediante una resta de punteros:

- Grabación: Tamaño de la memoria - Posición actual de grabación.
- Reproducción: Mayor posición de la grabación - Posición actual de reproducción.

Finalmente, se multiplica el resultado anterior por el periodo de reproducción: $20 \mu s$ Con lo cual tenemos ya en un registro el número de segundos que tenemos que mostrar en el display. Para ello existen dos opciones: Dividir el número en decenas y unidades y cada uno mostrarlo en un display de 7 segmentos diferente. La segunda opción, que supone un ahorro de operaciones y por tanto, de área, consiste en transformar el registro de segundos, en dos registros con unidades y decenas con dos ROMs, esta opción es viable ya que el valor máximo de segundos a mostrar es un número acotado (Tarea 3.2). Finalmente, usando un MUX transformamos el valor numérico de unidades y decenas en su codificación en el display de 7 segmentos. Este mismo método se usará en la siguiente mejora para mostrar el volumen actual es los displays.

Un detalle a tener en cuenta es que la frecuencia de barrido de los display no puede ser muy elevada, ya que si no, aparecen todas las cifras a la vez en todos los displays, la solución implementada consiste en mantener el reloj principal del sistema y usar un preescalador en el contador para reducir la velocidad de barrido.

3.8. Mejora Volumen:

Esta mejora consiste en que cada vez que se produzca un "rising edge" en el último switch se sube el volumen de reproducción, mientras que cuando se produce un "rising edge" en el penúltimo, se reduce el volumen de reproducción. Existen 21 niveles de audio siendo el 10 el sonido original, 0 sin volumen y 20 el volumen máximo. Además, en dos displays de 7 segmentos se puede ver cual es el volumen actual. El uso de los displays se explica en la mejora anterior, el funcionamiento del volumen consiste en multiplicar el valor del dato guardado en la memoria por un factor, cuando queremos el volumen original el factor valdrá 1 y cuando queremos volumen nulo el factor valdrá 0. Los factores están determinados por la ecuación siguiente:

$$factor = \frac{7^{\frac{nivel}{10}} - 1}{7 - 1} \quad (1)$$

Siendo nivel un valor de 1 a 20. La forma más sencilla para codificar esta ecuación es usar una ROM ya que el valor de nivel es un valor finito. La codificación es la siguiente:

Nivel	Ecuación	Codificación <4,5>sin signo
0	0	00000000
1	0.035802	00000001
2	0.079296	00000010
3	0.132132	000000100
4	0.196318	000000110
5	0.274292	000001000
6	0.369016	000001011
7	0.484088	000001111
8	0.623879	000010011
9	0.7937	000011001
10	1	000100000
11	1.250616	000101000
12	1.555069	000110001
13	1.924922	000111101
14	2.374224	001001011
15	2.920043	001011101
16	3.583112	001100010
17	4.388617	010001100
18	5.367156	010101011
19	6.555899	011010001
20	8	100000000

Cuadro 1: Codificación de la ROM

Por ello, únicamente quedará multiplicar el factor obtenido por el dato y descartar los bits decimales, en la práctica se observa que para valores elevados del nivel de volumen no se escucha nada, esto se cree que se debe a la saturación del volumen en el reproductor y no se escucha nada, la solución creemos que sería aumentar la frecuencia del pwm.