DSED PROYECTO FINAL

Carlos Gómez Rodríguez y Alejandro Ramos Martín

8 de enero 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: . . . 4
1.9. Tarea 1.9: . . . 4
1.10. Tarea 1.10: . . . 4
1.11. Tarea 1.11: . . . 4
1.12. Tarea 1.12: . . . 4
1.13. Tarea 1.13: . . . 4
1.14. Tarea 1.14: . . . 4
1.15. Tarea 1.15: . . . 4
1.16. Tarea 1.16: . . . 5

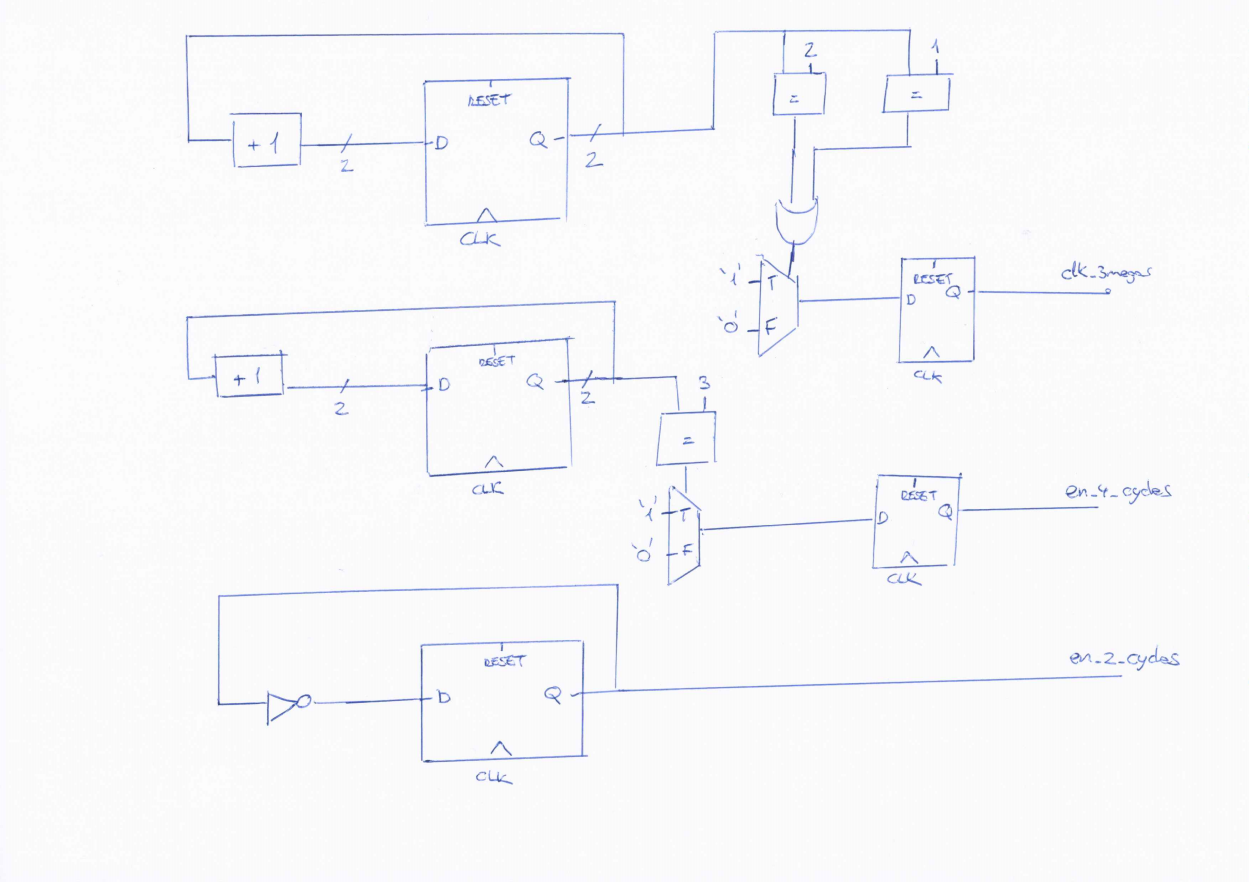
2. Bloque 2: Filtro FIR **5**
2.1. Tarea 2.1: . . . 5
2.2. Tarea 2.2: . . . 6
2.3. Tarea 2.3: . . . 9
2.4. Tarea 2.4: . . . 9
2.5. Tarea 2.5: . . . 10
2.6. Tarea 2.6: . . . 10
2.7. Tarea 2.7: . . . 10
2.8. Tarea 2.8: . . . 10
2.9. Tarea 2.9: . . . 10
2.10. Tarea 2.10: . . . 10
2.11. Tarea 2.11: . . . 10
2.12. Tarea 2.12: . . . 11
2.13. Tarea 2.13: . . . 11
2.14. Tarea 2.14: . . . 11

3. Bloque 3: Controlador y Memoria **11**
3.1. Tarea 3.1: . . . 11
3.2. Tarea 3.2: . . . 11
3.3. Tarea 3.3: . . . 11
3.4. Tarea 3.4: . . . 11
3.5. Tarea 3.5: . . . 11
3.6. Tarea 3.6: . . . 12
3.7. Tarea 3.7 (Opcional): . . . 12

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:

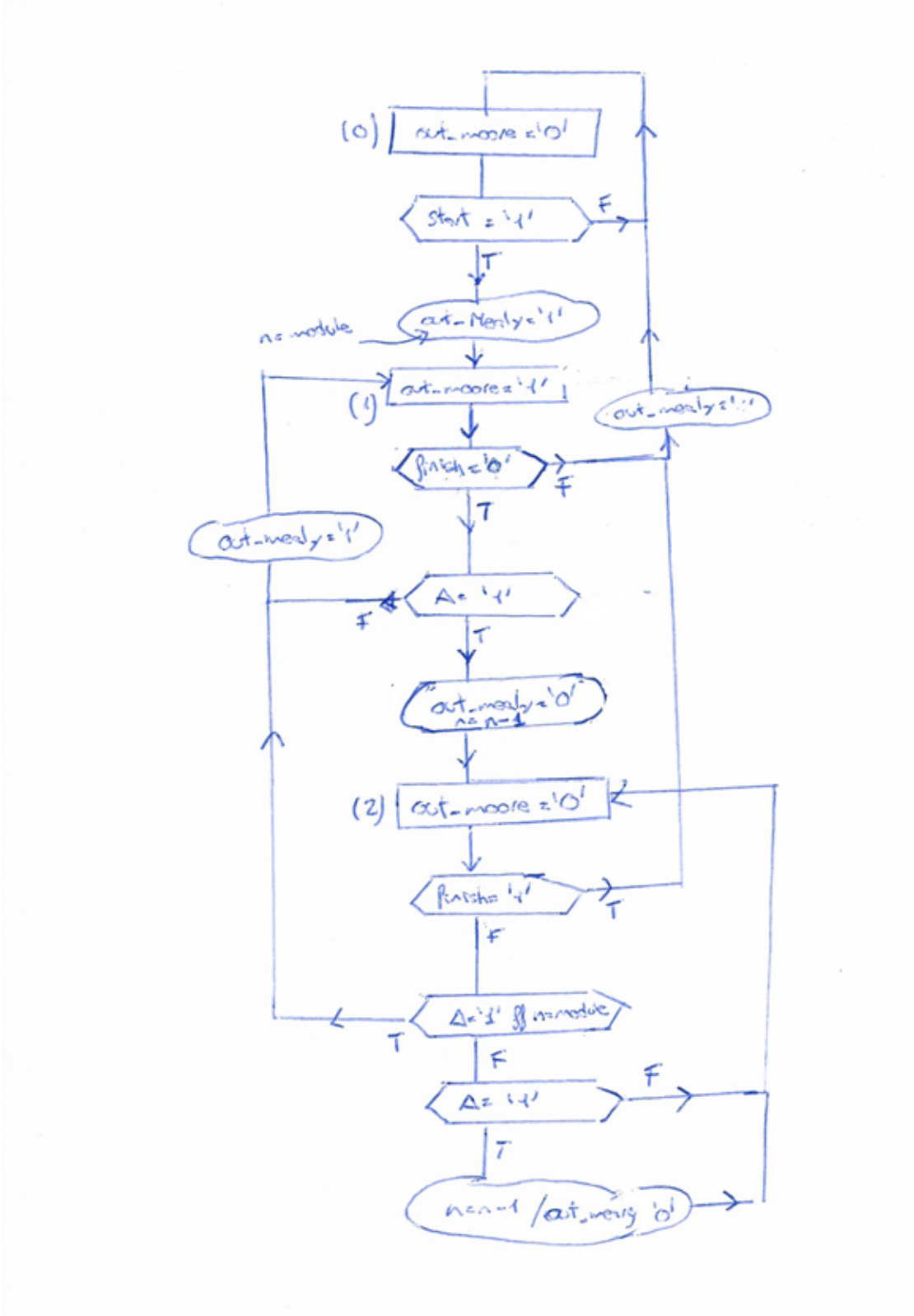
Implementa tu diseño en VHDL.

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.

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.

1.12. Tarea 1.12:

Implementa en VHDL la interfaz de audio completa. Para ello utiliza un estilo de código estructural que instancie 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.

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.

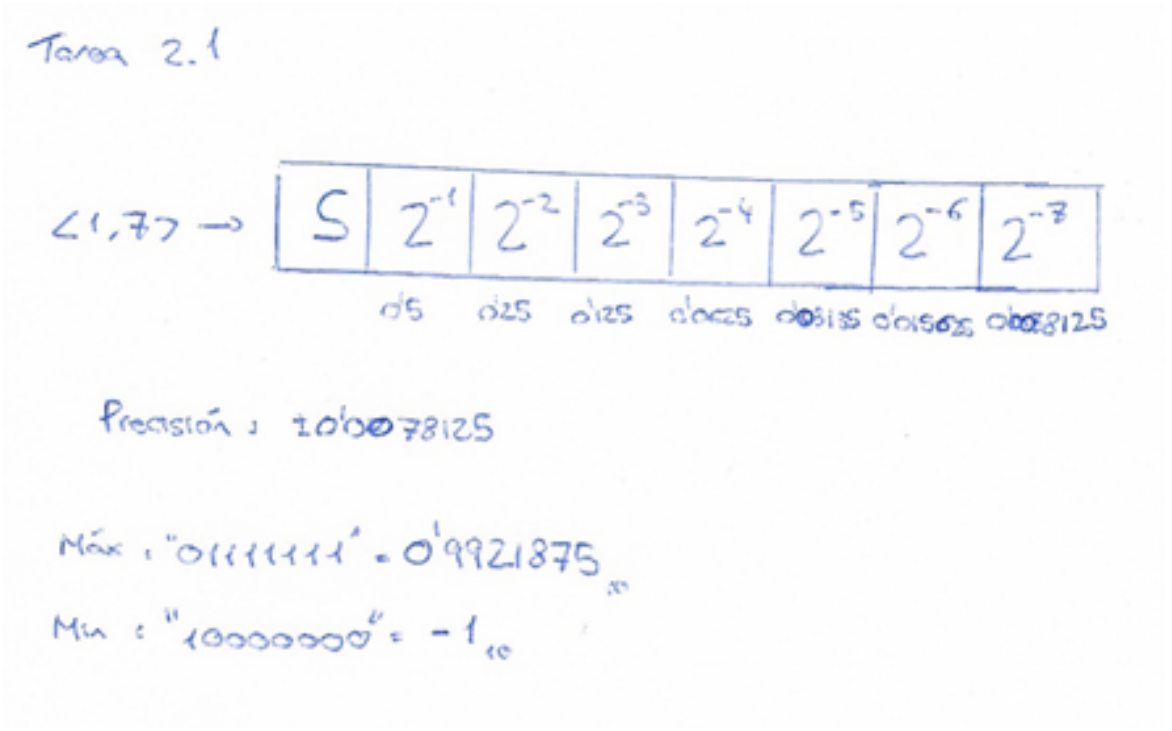
1.16. Tarea 1.16:

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:

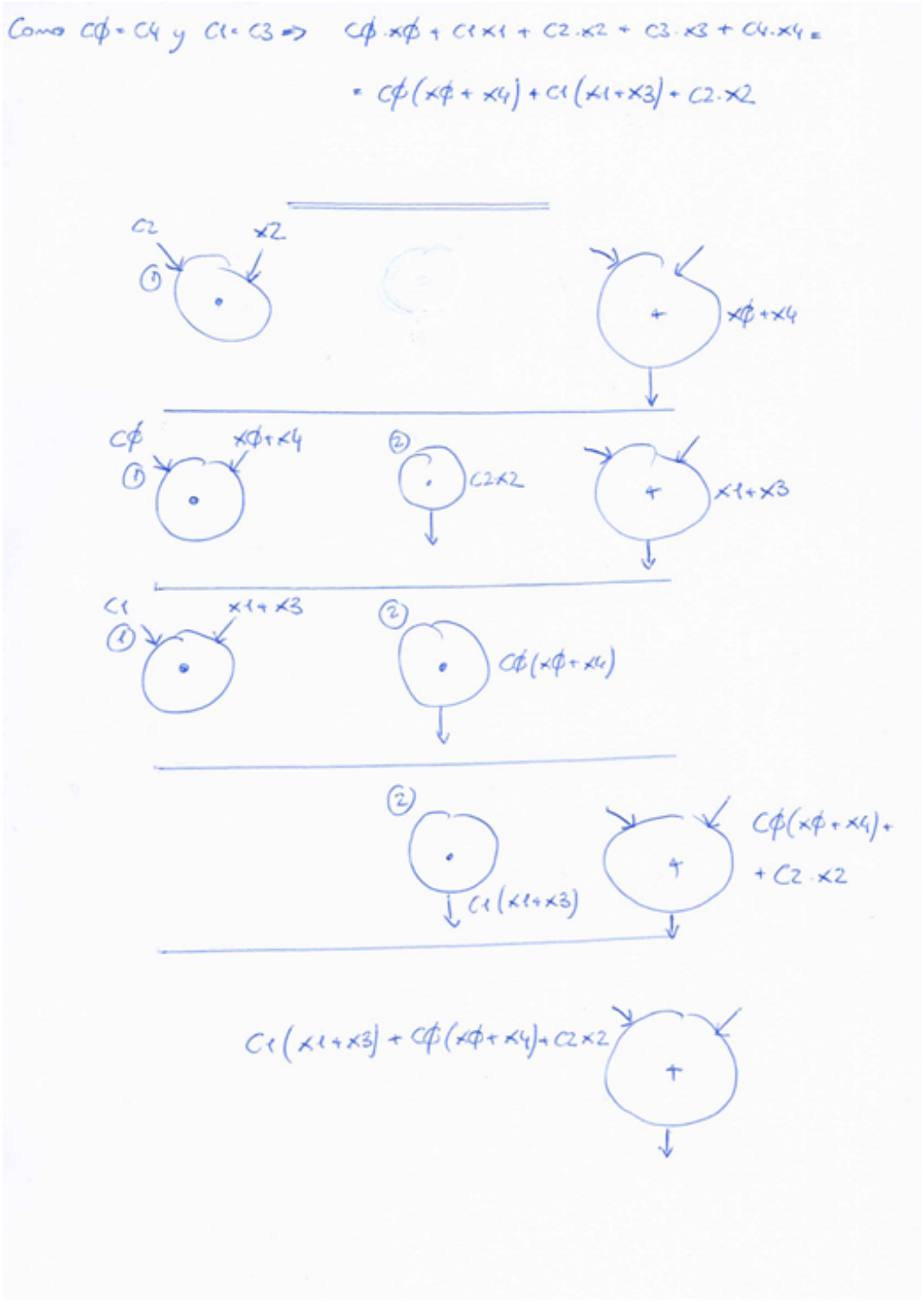
¿Cuál es el rango de un número de 8 bits en punto fijo <1,7>en complemento a dos? ¿Cuál es su precisión?



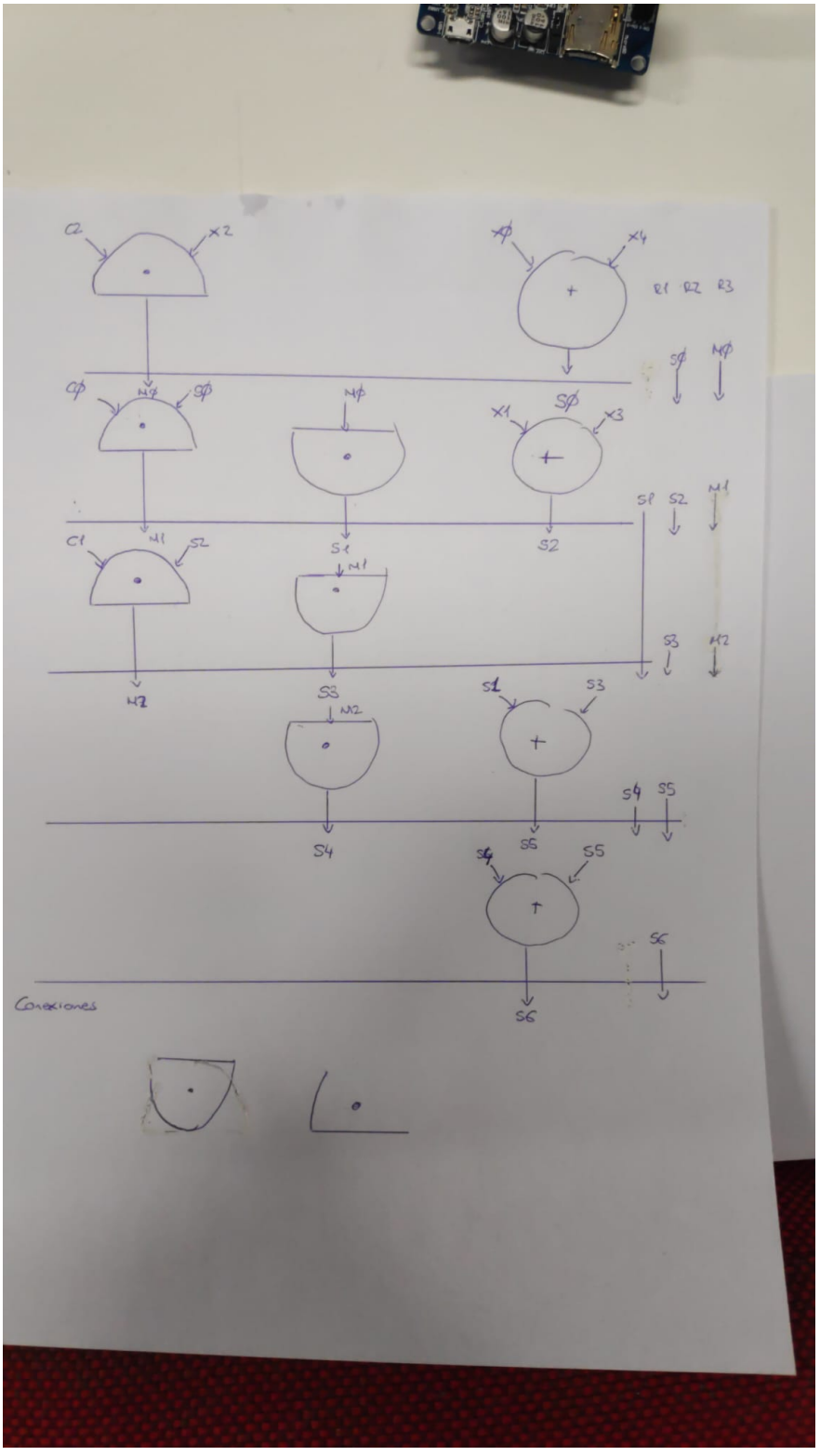
2.2. Tarea 2.2:

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.

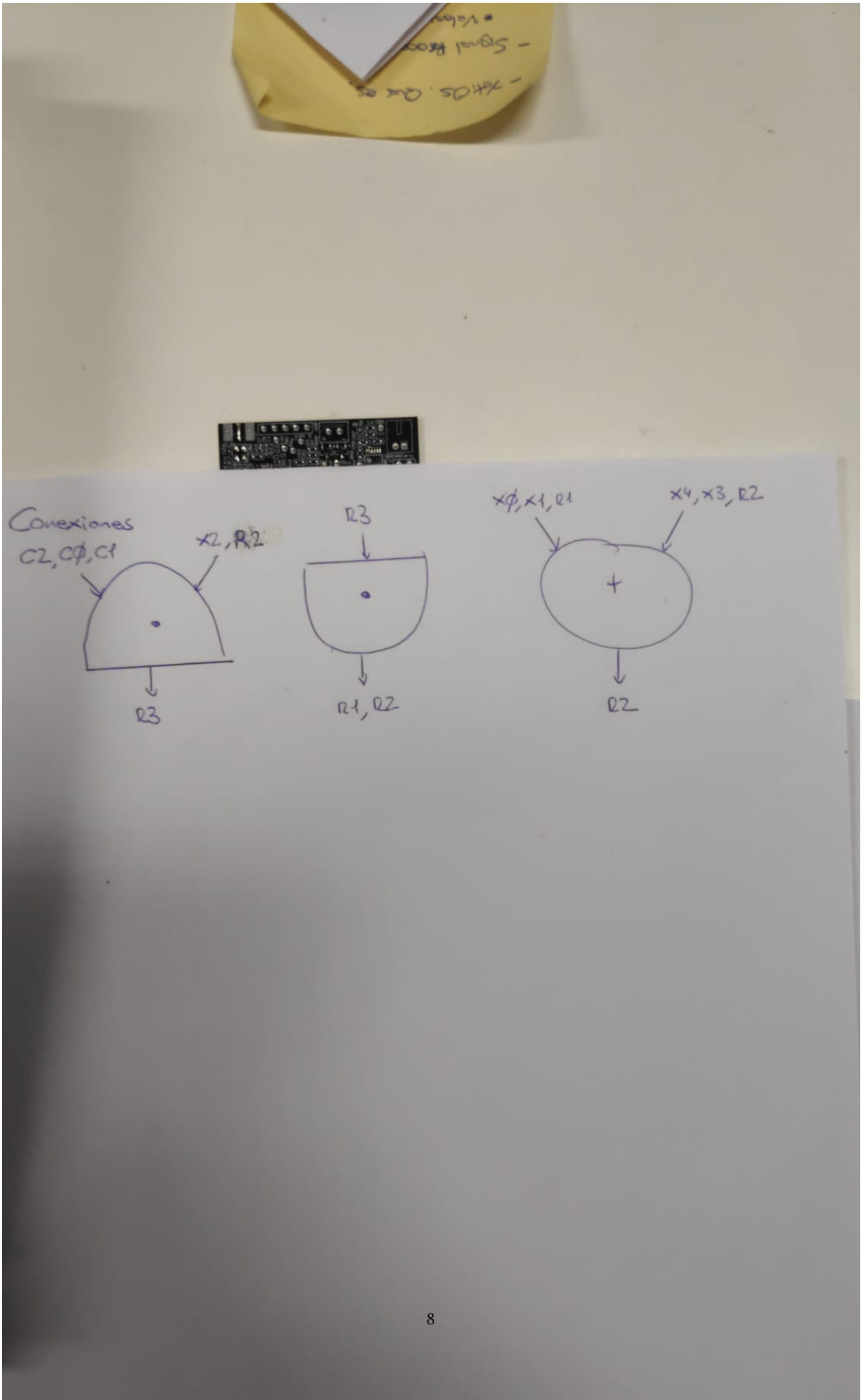
Planificación temporal



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



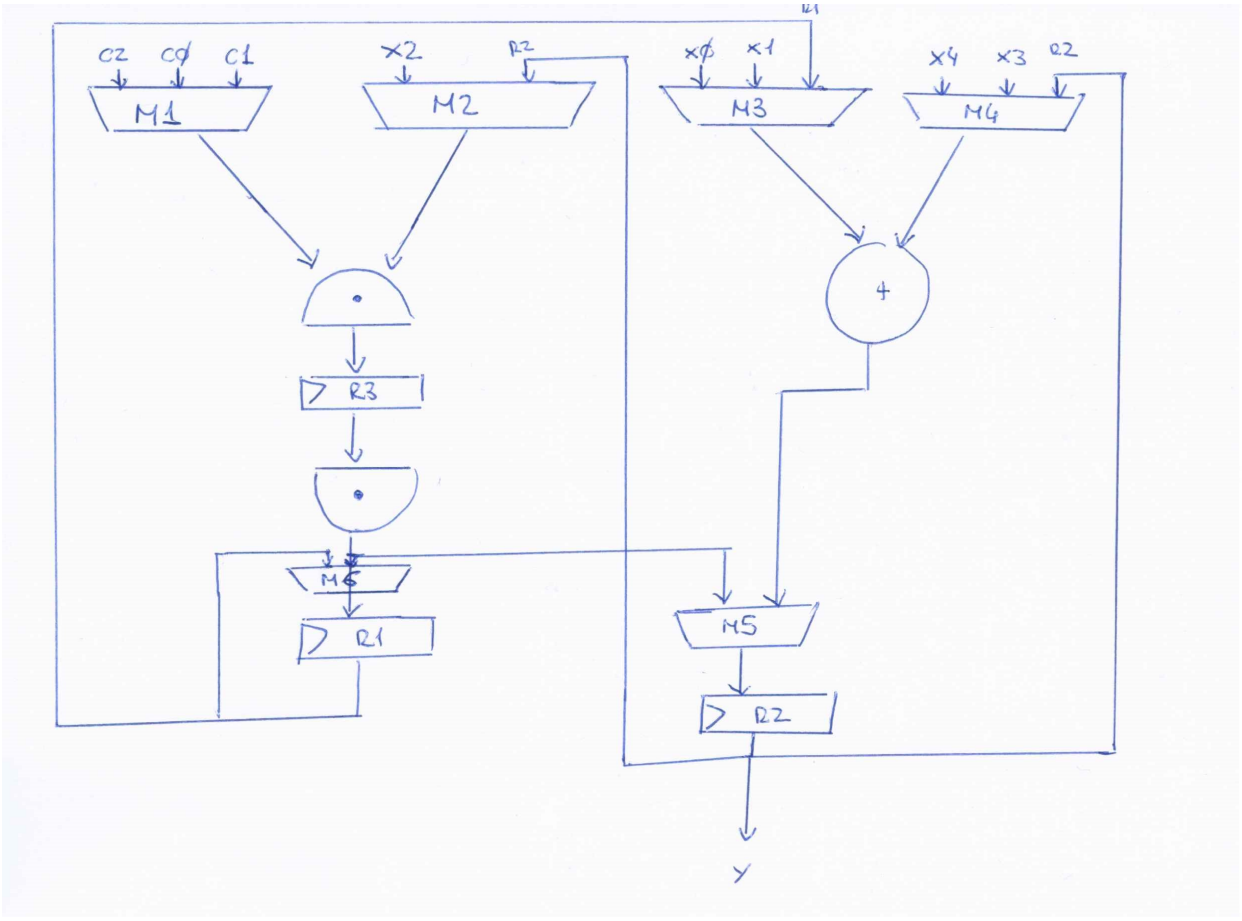
Análisis de conexiones para la extracción de multiplexores:


Conexiones
C2, Cϕ, C1
x2, R2
R3
R3
R1, R2
xϕ, x1, R1
x4, x3, R2
+
R2

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	T6 101
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		sum_out	R1	sum_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.

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 introducimos x0=+0.5 y x1=x2=x3=x4=0, la salida y=c0*x0. Para comprobar si un número encaja en la resolución de nuestras señales, hay que multiplicar el valor por 128 y descartar los decimales. Para el caso paso bajo:

$$c0 * x0 = 0{,}0195 \qquad y = 0{,}0195 * 2^7 = 2{,}496 == 2$$

En el caso planteado, el filtro quedaría como c0*x0+c1*0+c2*0+c3*0+c4*0. Positivo

$$Mayor = 127/128 = 0{,}9921875 \qquad 0{,}9921875 * C0 * 128 = 0{,}03125$$

$$Menor = 1/128 = 0{,}0078125 \qquad 0{,}0078125 * C0 * 128 = 0$$

Negativo

$$Mayor = -128/128 = -1 \qquad -1 * C0 = -0{,}03125$$

$$Menor = -1/128 = -0{,}0078125 \qquad -0{,}0078125 * C0 * 128 = -0{,}0078125$$

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.

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.

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).

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 estás 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.

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

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 signed ->unsigned como en unsigned) es cambiar el bit más significativo, a(7) = 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. Tarea 3.7 (Opcional):

Añade a continuación todas las hojas necesarias para describir las mejoras que hayas implementado.