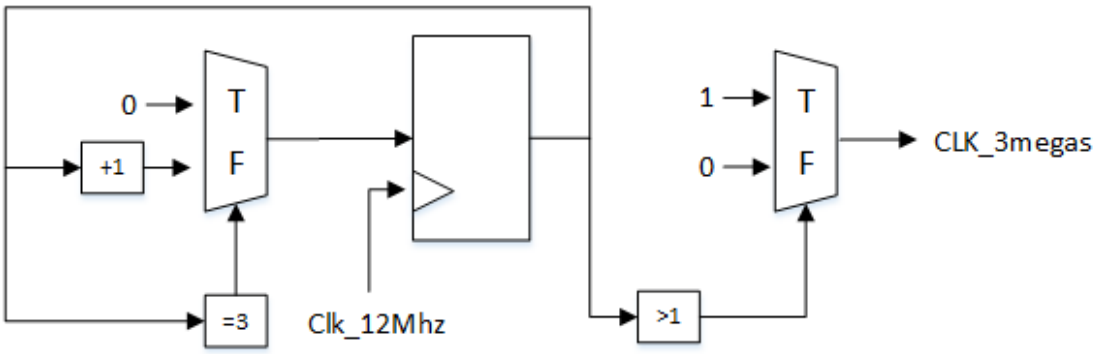
Sistema de grabación, tratamiento y reproducción de audio

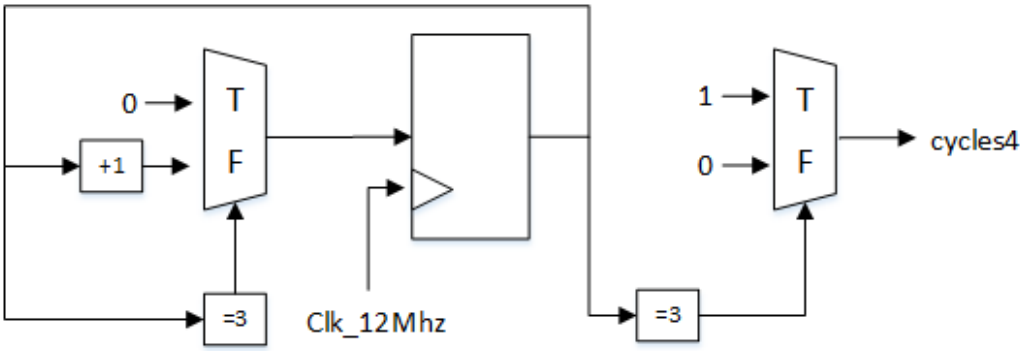
- Diseño de Sistemas Electrónicos Digitales -

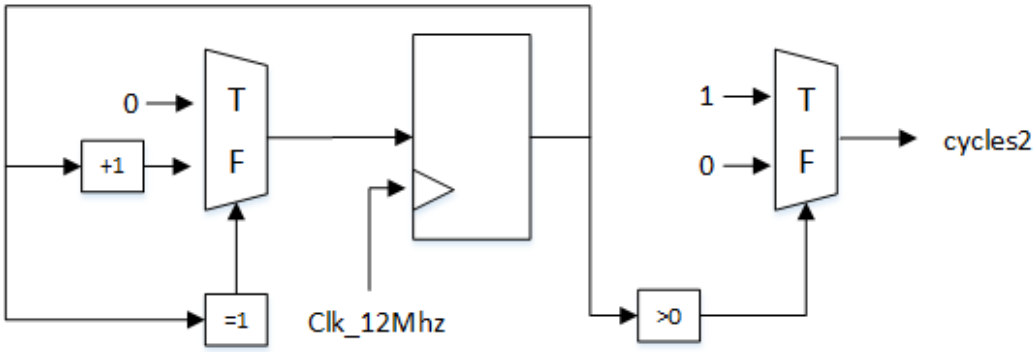
Jaime Pérez Sánchez
Sergio Pérez Morillo
Grupo 4

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.

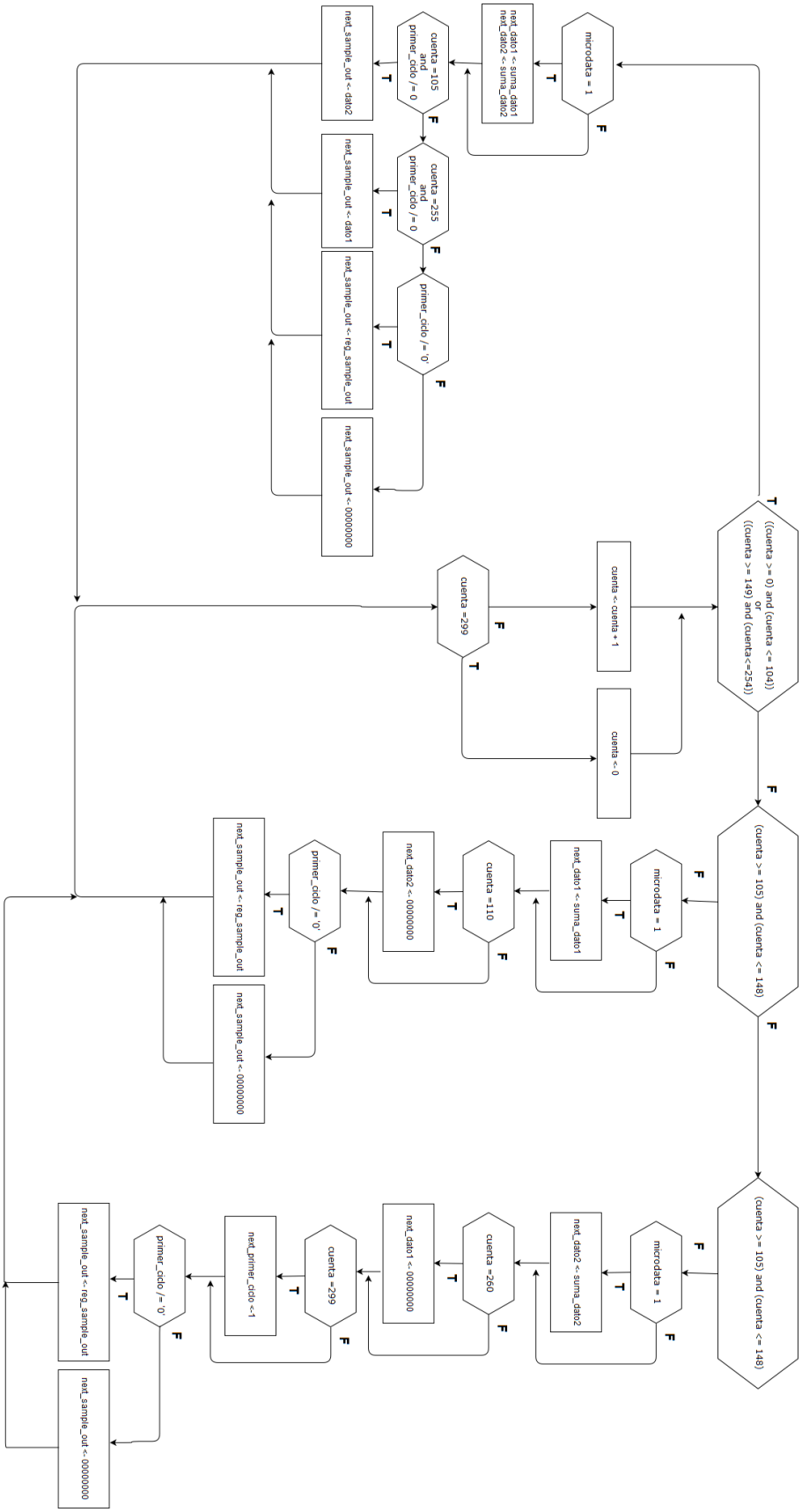






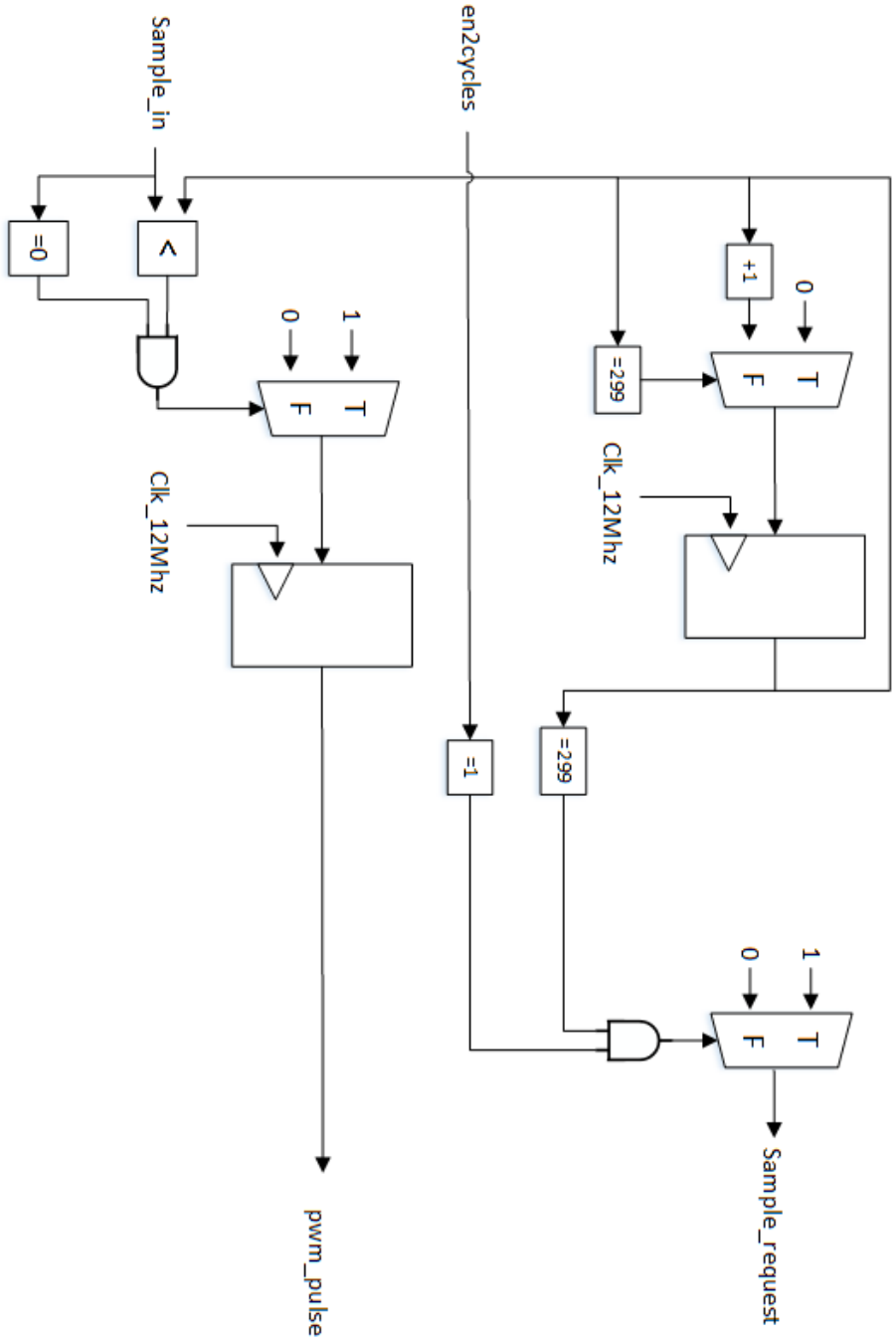
Tarea 1.4:

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



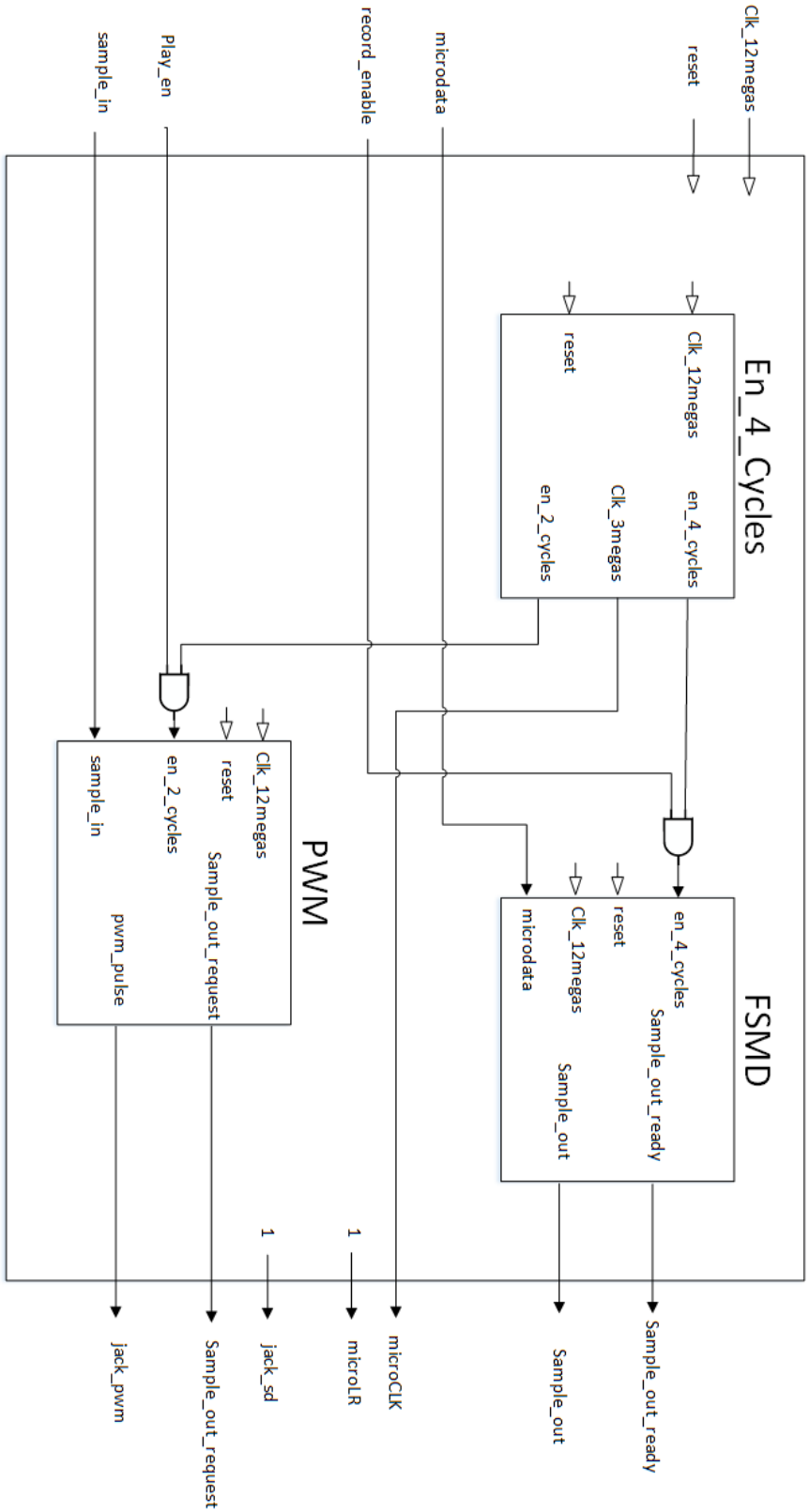
Tarea 1.8:

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.



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.



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?

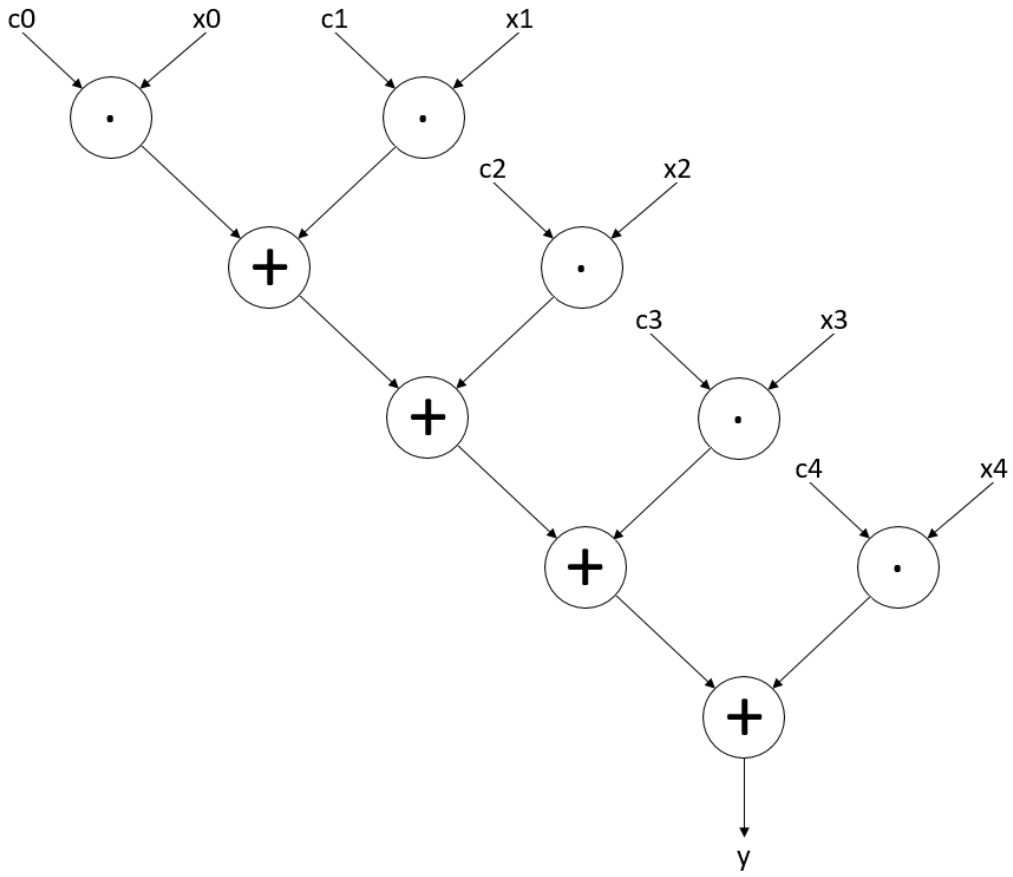
Rango: [1'0000000 ,, 0'1111111] → [-1 ,, 0'9921875]

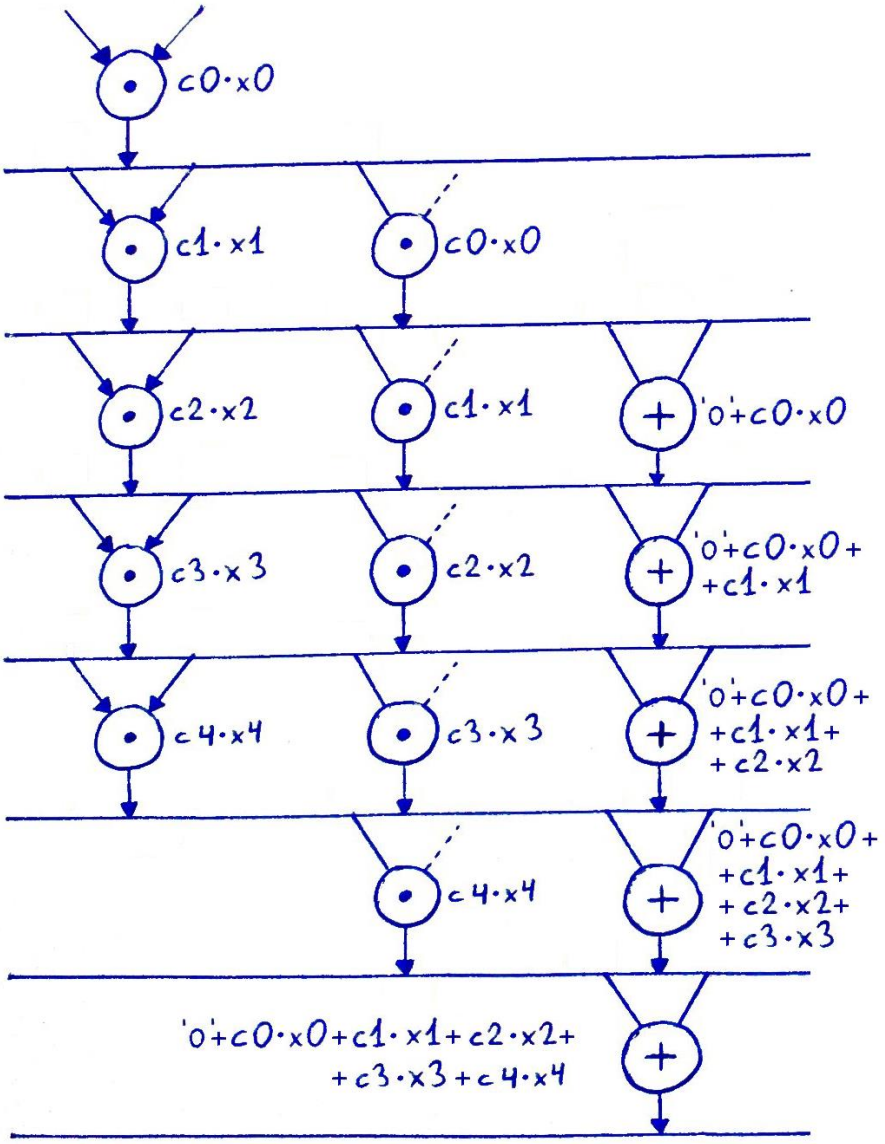
Precisión: $2^{-7} = 7{,}8125 * 10^{-3}$

Tarea 2.2:

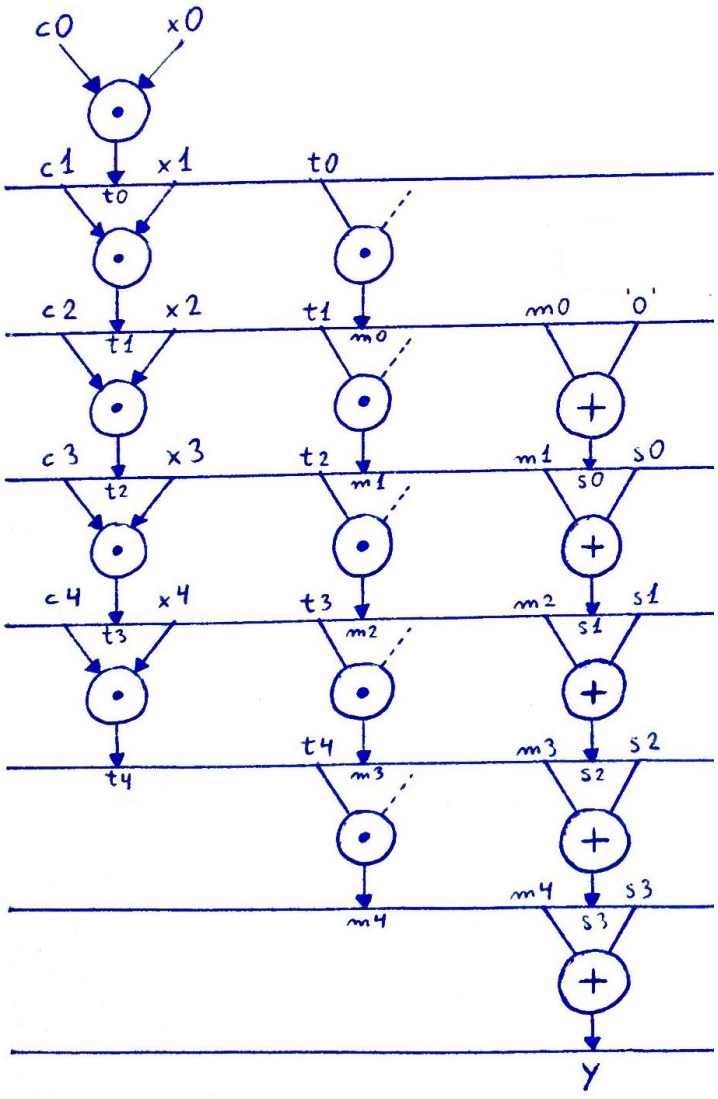
Realiza sobre papel la implementación de un filtro FIR de 5 etapas con la siguiente asignación: Dos medios multiplicadores y un sumador. Realiza todos los pasos descritos en la sección 6.4.

Planificación temporal:

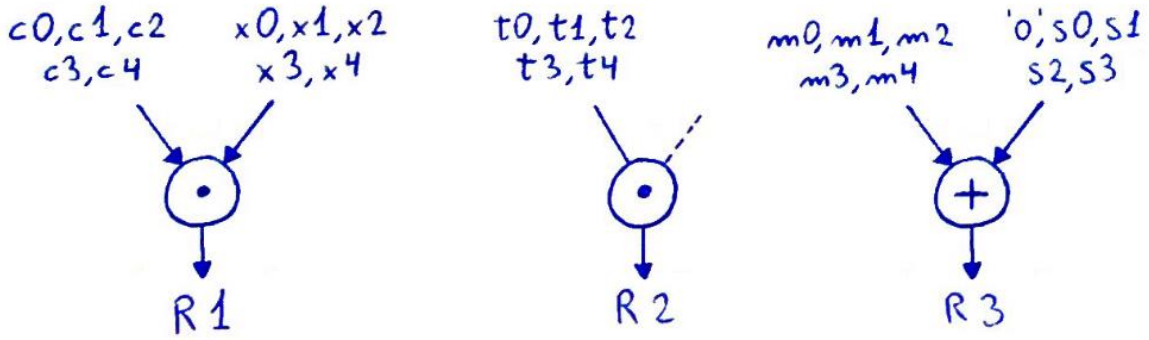


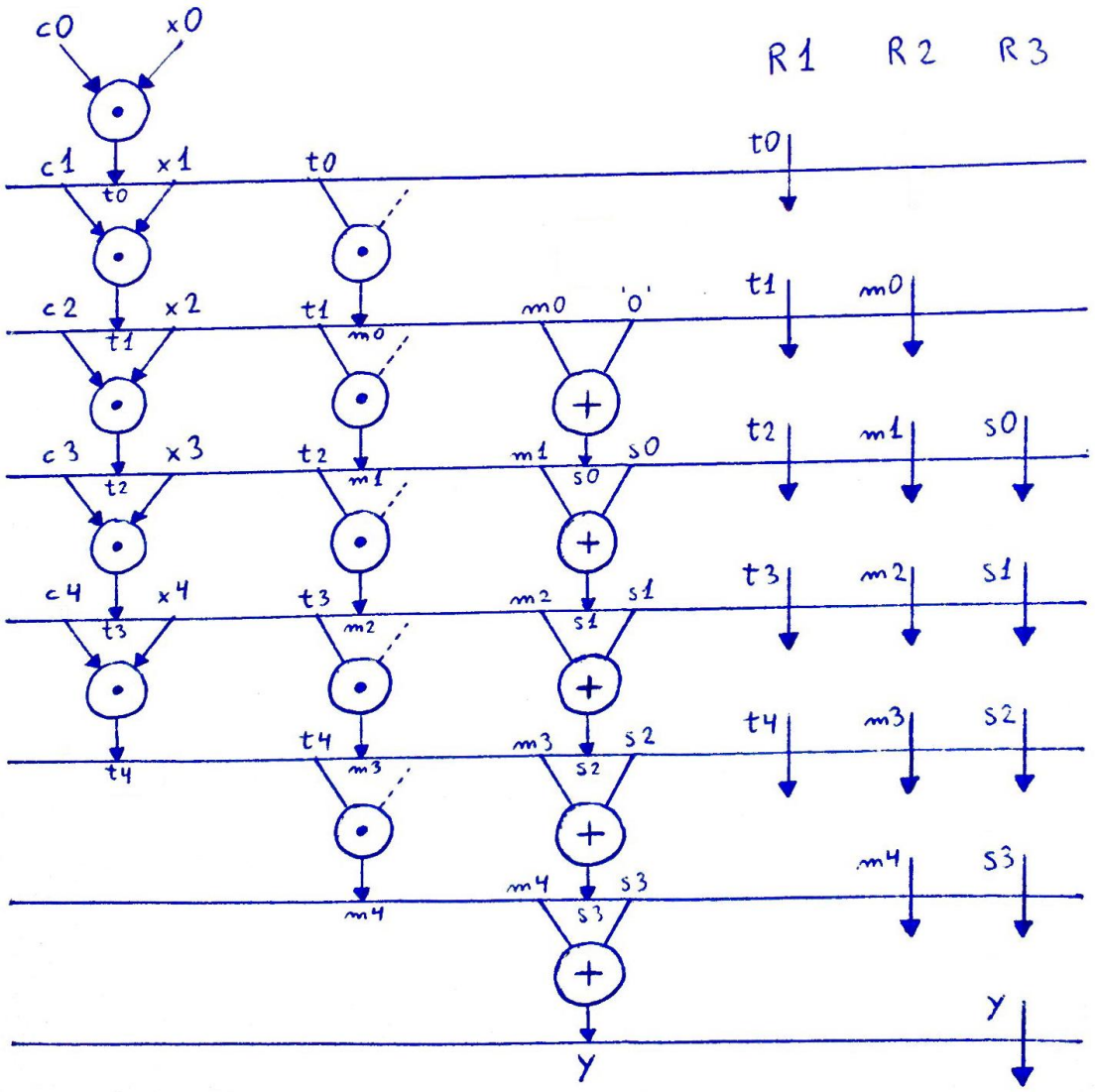

c0·x0
c1·x1
c0·x0
c2·x2
c1·x1
'0'+c0·x0
c3·x3
c2·x2
'0'+c0·x0+
+c1·x1
c4·x4
c3·x3
'0'+c0·x0+
+c1·x1+
+c2·x2
c4·x4
'0'+c0·x0+
+c1·x1+
+c2·x2+
+c3·x3
'0'+c0·x0+c1·x1+c2·x2+
+c3·x3+c4·x4

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

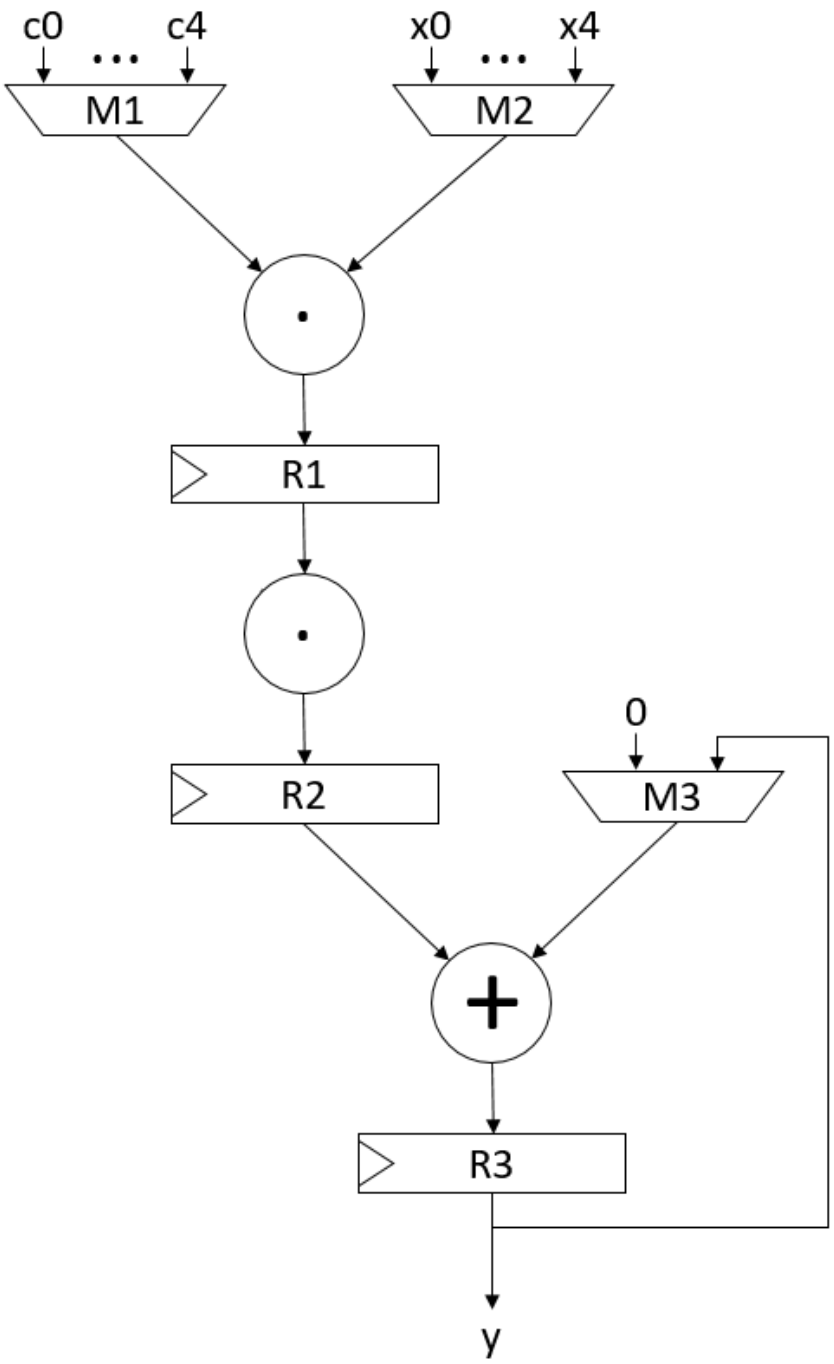


Análisis de conexiones para la extracción de multiplexores:




c0
x0
R1
R2
R3
c1
x1
t0
t0
t0
c2
x2
t1
t1
m0
'0'
t1
m0
c3
x3
t2
t2
m1
m1
s0
s0
t2
m1
s0
c4
x4
t3
t3
m2
m2
s1
s1
t3
m2
s1
t4
t4
m3
m3
s2
s2
t4
m3
s2
m4
m4
s3
s3
m4
s3
y
y

Implementación:



Cronograma:

	t0	t1	t2	t3	t4	t5	t6
M1	c0	c1	c2	c3	c4	-	-
M2	x0	x1	x2	x3	x4	-	-
M3	-	-	0	R3 (realim.)	R3 (realim.)	R3 (realim.)	R3 (realim.)
R1	t0	t1	t2	t3	t4	-	-
R2	-	m0	m1	m2	m3	m4	-
R3	-	-	s0	s1	s2	s3	y

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.

c0 – c4: signed (7 downto 0) → <1,7>

x0 – x4: signed (7 downto 0) → <1,7>

t0 – t4: signed (15 downto 0) → <2,14>

m0 – m4: signed (15 downto 0) → <2,14>

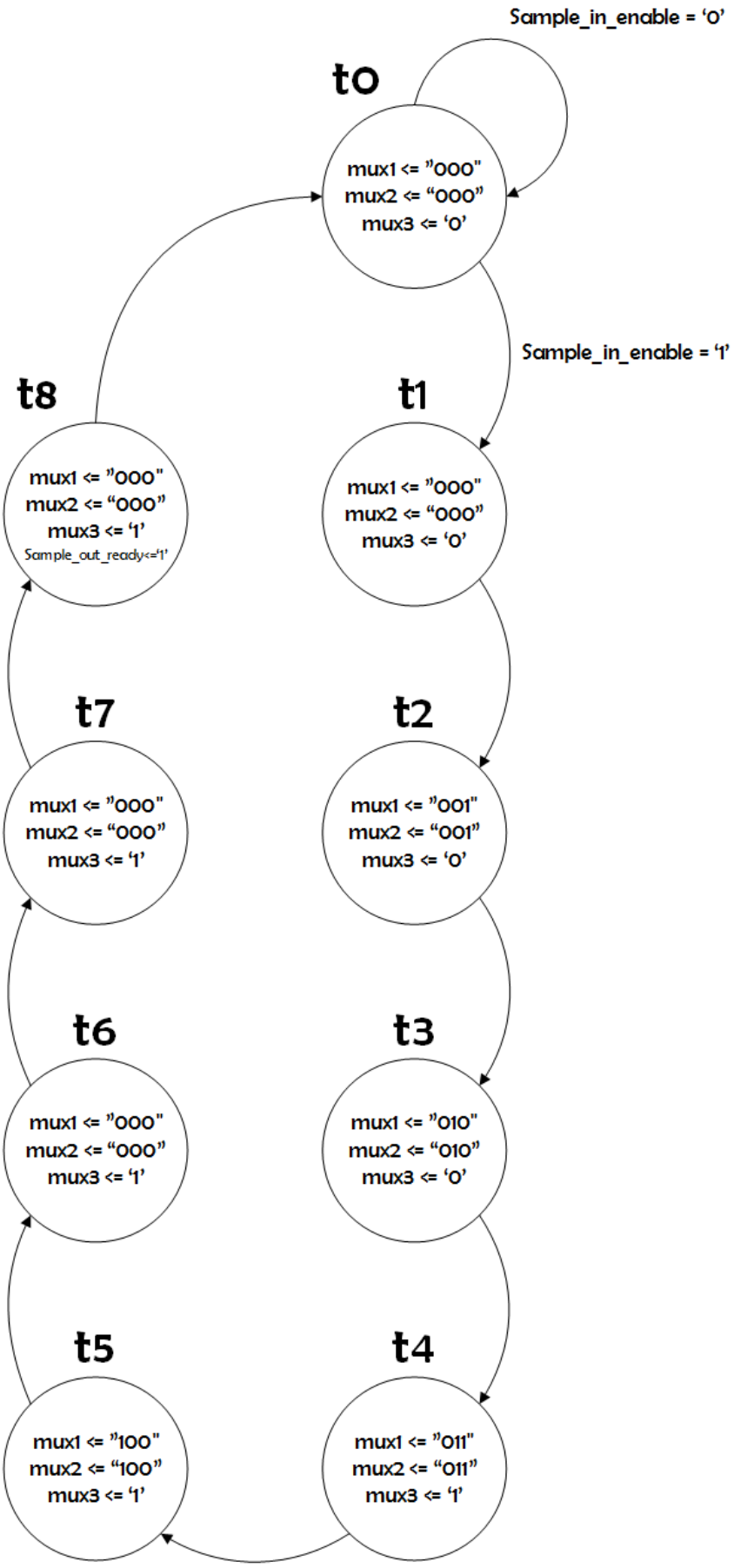
s0 – s3: signed (15 downto 0) → <2,14>

y: signed (14 downto 7) → <1,7> (truncada)

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.

Además de los estados indicados en el cronograma decidimos añadir dos estados extra. Uno inicial para cargar el registro de entrada con los datos, y otro final para capturar adecuadamente el resultado final en el controlador de sistema.


Sample_in_enable = '0'
t0
mux1 <= "000"
mux2 <= "000"
mux3 <= '0'
Sample_in_enable = '1'
t8
mux1 <= "000"
mux2 <= "000"
mux3 <= '1'
Sample_out_ready<='1'
t1
mux1 <= "000"
mux2 <= "000"
mux3 <= '0'
t7
mux1 <= "000"
mux2 <= "000"
mux3 <= '1'
t2
mux1 <= "001"
mux2 <= "001"
mux3 <= '0'
t6
mux1 <= "000"
mux2 <= "000"
mux3 <= '1'
t3
mux1 <= "010"
mux2 <= "010"
mux3 <= '0'
t5
mux1 <= "100"
mux2 <= "100"
mux3 <= '1'
t4
mux1 <= "011"
mux2 <= "011"
mux3 <= '1'

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.

Está realizado para el filtro paso bajo:

DECIMAL	C0	C1	C2	C3	C4
-1	-0,039	-0,2422	-0,4453	-0,2422	-0,039
$-7{,}8125*10^{-3}$	$-0{,}3047*10^{-3}$	$-1{,}8922*10^{-3}$	$-3{,}4789*10^{-3}$	$-0{,}3047*10^{-3}$	$-1{,}8922*10^{-3}$
$7{,}8125*10^{-3}$	$0{,}3047*10^{-3}$	$1{,}8922*10^{-3}$	$3{,}4789*10^{-3}$	$0{,}3047*10^{-3}$	$1{,}8922*10^{-3}$
0,9921875	0,03896	0,24031	0,44182	0,03896	0,24031

BINARIO	C0	C1	C2	C3	C4
10000000	11111011	11100001	11000111	11111011	11100001
11111111	11111111	11111111	11111111	11111111	11111111
00000001	00000000	00000000	00000000	00000000	00000000
01111111	00000100	00011110	00111000	00000100	00011110

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.

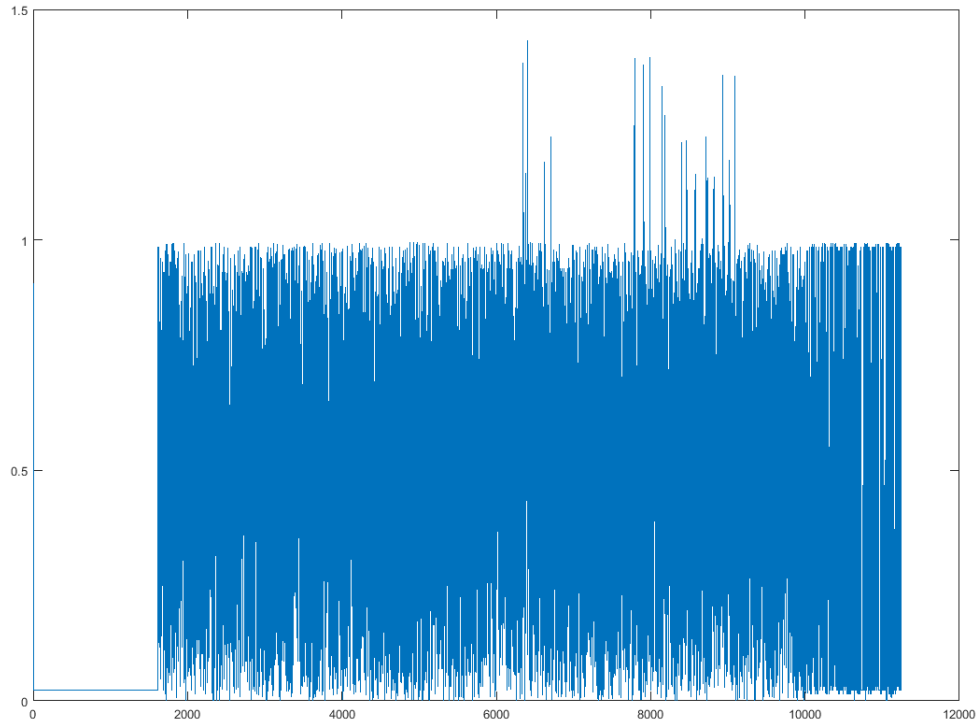
1. ...
2. 00000000
3. 00000010 (En este instante se introduce 0,5)
4. 00001111
5. 00011101 (En este instante se introduce 0,125)
6. 00010011
7. 00001001
8. 00000011
9. 00000000
10. ...

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.

Está realizado para el filtro paso bajo y son valores no normalizados sobre 1. La escala de los datos introducidos es sobre 64.

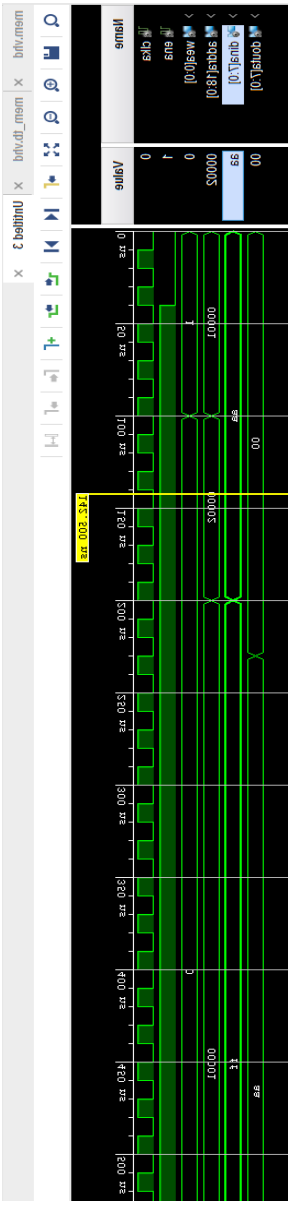
Observamos que los errores están acotados entre 0 y 1.5, por lo tanto, filtra correctamente.



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).

- clka: Reloj de la memoria RAM
- ena: Enable del componente. Si está a '0' no se pueden leer ni escribir datos.
- wea: Si está a '0' se active el modo lectura de datos, y a '1' el modo escritura.
- addra: En este puerto se indica la dirección de memoria a la que se quiere acceder, ya sea para leer o escribir.
- dina: Entrada de datos. Únicamente se utiliza en modo escritura de datos.
- douta: Salida de datos. Únicamente se utiliza en modo lectura de datos.



Tarea 3.2:
Determina cuánto tiempo de grabación va a poder estar almacenado en la memoria.

El sistema guarda 8 bits de datos cada 50 µs.

En la memoria hay 2^{19} direcciones de memoria de 8 bits cada una.

Por lo tanto, el tiempo máximo de grabación será: 2^{19} * 50µs = **26,214 segundos**

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 de transformación de binario a complemento a 2 es:

```
complemento_a2<= (not binario (7)) & ( binario (6 downto 0));
```

Es análoga para la transformación de complemento 2 a binario.

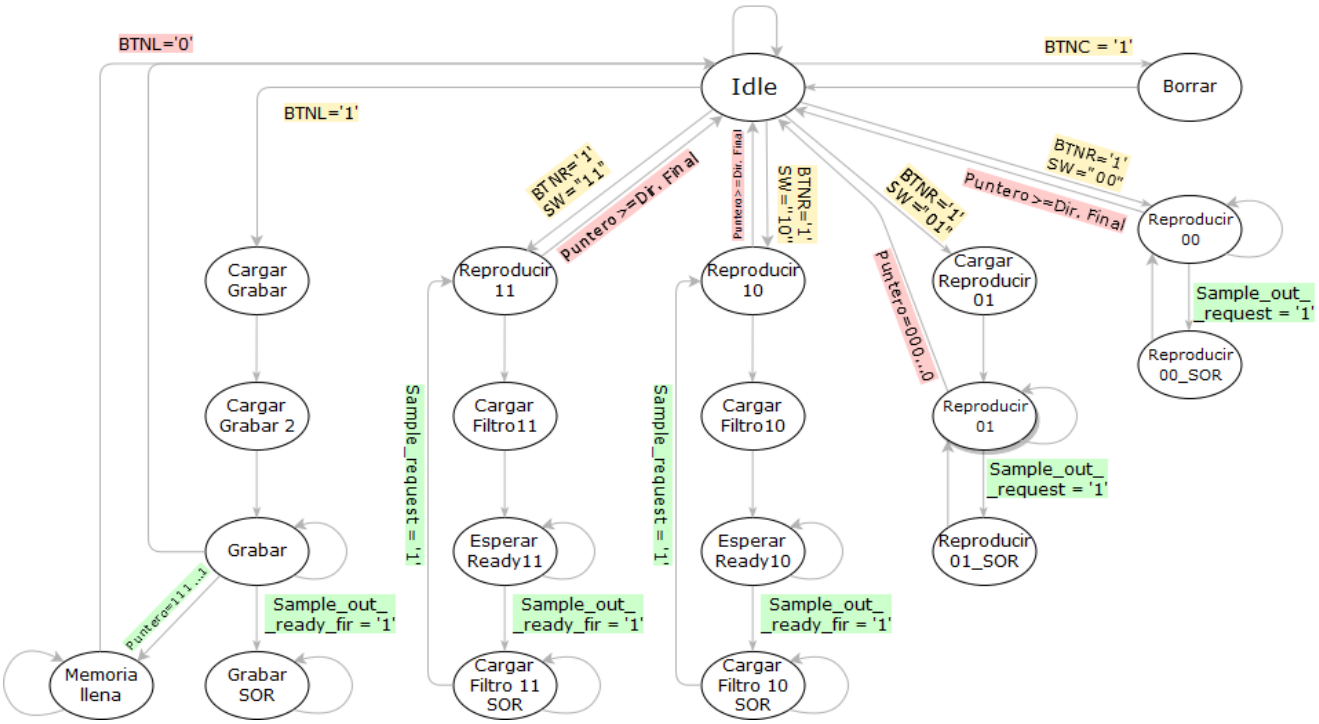
Tarea 3.4:
Añade a continuación todas las hojas necesarias para describir tu diseño y la planificación para llevarlo a cabo. Puedes incluir diagramas esquemáticos, cronogramas explicativos, un plan de pruebas, una planificación temporal y todo lo que consideres necesario para explicar las decisiones que has tomado.
Avisa al profesor antes de comenzar con el diseño para obtener el visto bueno.

El controlador es un FSMD en el que se registran los valores de dos punteros que sirven para recorrer las direcciones de la memoria RAM tanto para grabar, como borrar, o reproducir. Además de los punteros, se registran valores necesarios para el correcto funcionamiento de los filtros que se detallarán más adelante.

Los dos punteros registrados son:

- Puntero: Recorre la posiciones de la memoria RAM una por una y cuando termina su función se reinicia a cero.
- Dirección final: Como su nombre indica, registra la última posición recorrida por puntero, éste no se reinicia normalmente.

La máquina de estados del controlador diseñada es la siguiente:



A continuación se describen los diferentes estados agrupados por las funciones por las que fueron diseñados (grabar, reproducir…):

BORRAR

Estado *Borrar*

Al pulsar el botón btnc el valor de dirección final pasa a ser cero con lo que al intentar reproducir no recorre ninguna posición de la RAM.

REPRODUCIR

Cuando se pulsa el botón btnr se ponen las señales de entrada en audio interface y RAM a sus valores correspondientes para reproducir y si se necesita también las del filtro. Dependiendo de la combinación en los switch cero y switch uno, se reproduce de una manera u otra.

En todas las funciones de reproducción destacar que al pulsar el botón btnc se para la reproducción y se borra la memoria.

También mencionar el uso de la señal “enable_filtro”, que sirve para seleccionar la entrada de la interfaz de audio “sample_in” entre la señal de salida de la RAM “douta” o la señal de salida del filtro “sample_out_fir”, dependiendo de si se necesita filtrar o no.

Las diferentes formas de reproducir son:

Reproducción estándar

Estado *Reproducir00*

Reproduce la muestra recogida de la RAM. Cuando el puntero alcanza la dirección_final (terminan las muestras a reproducir en la RAM y con ello la reproducción) volvemos al estado de reposo.

Estado *Reproducir00_SOR*

Cuando la interfaz de audio solicita otra muestra (sample_out_request = '1') entramos en este estado, en el que incrementamos el puntero en uno para que la muestra a la entrada de la interfaz cambie a la siguiente en la RAM.

Reproducción al revés

Estado *cargar_reproducir01*

Cargamos la dirección final en puntero para recorrer la RAM del final al principio.

Estado *Reproducir01*

Análogo a *Reproducir00* pero en este caso paramos de reproducir al llegar a la dirección cero.

Estado *Reproducir01_SOR*

Análogo a *Reproducir00_SOR*, esta vez se resta el puntero no se suma.

Reproducción con filtro

Consta de 4 estados para el filtro paso bajo y otros 4 para el paso alto. En ellos se realizarán dos actividades en paralelo, activar el filtrado de datos y registrar el valor resultante, y enviar los datos a la interfaz de audio para que los reproduzca. Como funcionan igual solo se explica uno de ellos, los estados son los siguientes:

Estado *Reproducir1X*

En este estado cambia el dato que se está enviando a la interfaz de audio y se envía una señal al filtro (sample_in_en <= '1') para que comience a realizar las operaciones de filtrado del siguiente dato.

Estado *Cargar_filtro1X*:

Se desactiva la señal enviada al filtro (sample_in_en <= '0') y se pasa a la siguiente dirección de memoria RAM (puntero + 1).

Estado *Espera1X _ready*:

Espera en este estado hasta que el filtro indique que ha terminado de realizar las operaciones (sample_out_ready_fir = '1') y se registra el valor que proporcione el filtro para enviarlo más adelante.

Estado *Cargar_filtro1X _SOR*:

Espera en este estado hasta que la interfaz de audio solicite nuevos datos para reproducir (sample_out_request = ‘1’). Cuando esto ocurre vuelve al estado “Reproducir1X” y comienza de nuevo el bucle.

GRABAR

Se ponen las señales de entrada de la interfaz de audio y la RAM para poder grabar.

Estado *cargar_grabar*

Cuando se pulsa el botón btnl cargamos la dirección final en puntero para, o bien empezar a grabar, o seguir donde lo habíamos dejado.

Estado *cargar_grabar_2*

Necesario para cargar correctamente la dirección anterior en puntero.

Estado *grabar*

Graba en la memoria RAM el valor de la salida del audio interface “sample_out”. Si deja de pulsar el botón btnl vuelve al estado de reposo.

Estado *grabar_SOR*

Al llegar un simple_out_ready del audio interface se aumenta el puntero en uno para pasar a grabar en la siguiente dirección.

Estado *memoria llena*

Cuando la memoria RAM se llena pasamos a este estado que no hace nada y en el que se espera a que deje de pulsar el botón btnl para pasar al estado de reposo.

Sistema completo