Memoria Práctica E4
Filtro FIR compensador del CIC

David Martínez Esteso

Néstor García García

Tabla de contenido

Sección 1: Descripción del módulo.

La estructura a desarrollar en esta práctica se trata del diseño Verilog de un filtro FIR secuencial cuyo objetivo es compensar la respuesta en frecuencia del filtro CIC desarrollado en la práctica anterior. En este filtro la frecuencia de muestreo para los datos de entrada es de 50 kHz y la frecuencia del reloj de 100 MHz.

El esquema de la estructura del filtro FIR compensador del CIC es la siguiente:

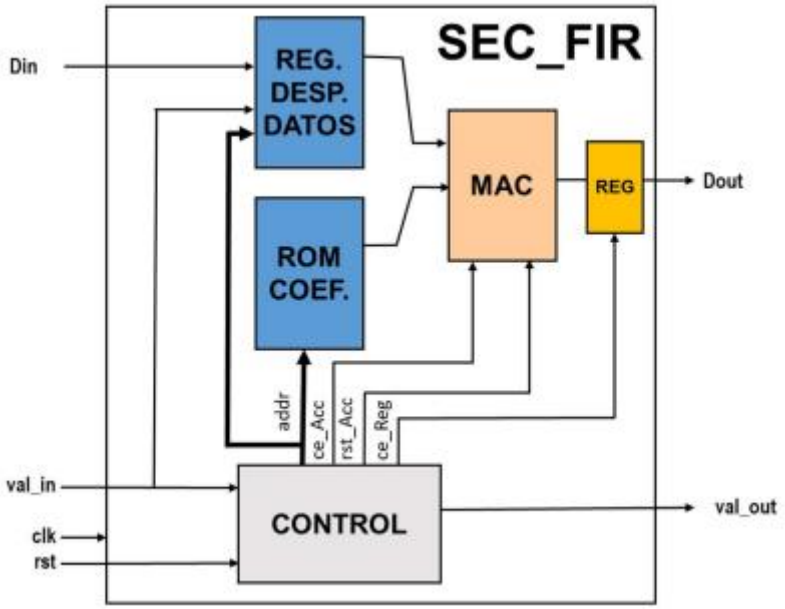


Ilustración 1: estructura filtro FIR compensador del CIC

- **¿Por qué necesitamos un filtro compensador de la respuesta del CIC?**

Los filtros CIC tienen el problema de que su respuesta en la banda de paso no es plana, cosa que no es deseable para ciertas aplicaciones

del filtro, por lo que es necesario la utilización de un filtro compensador a la respuesta de este filtro para solucionar este problema.

Simulaciones de comportamiento:

Una vez comparado el modelo el estudio del modelo simulink proporcionado, es decir se ha comparado el modelo sin cuantificación con el modelo cuantificado, se han obtenido la respuesta al impulso y se ha realizado el barrido en frecuencia con una señal sinusoidal en distintas frecuencias dentro del rango obteniendo los siguientes resultados:

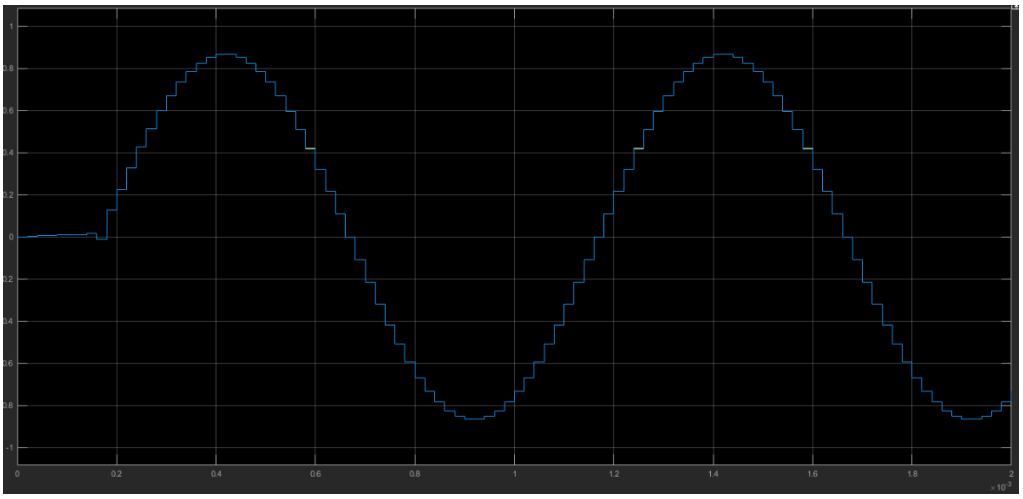


Ilustración 2: señal de entrada senoidal de 1 kHz

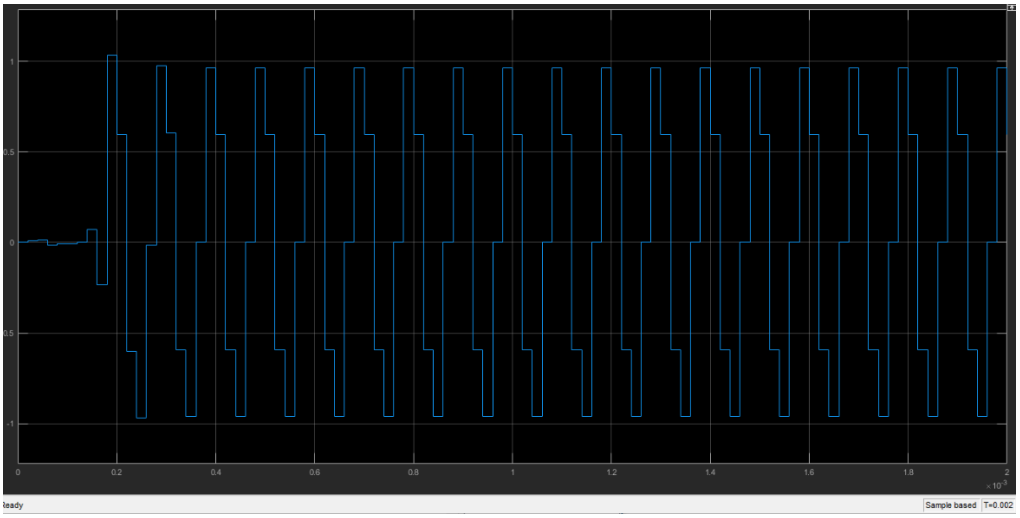


Ilustración 3: señal de entrada senoidal de 10 kHz

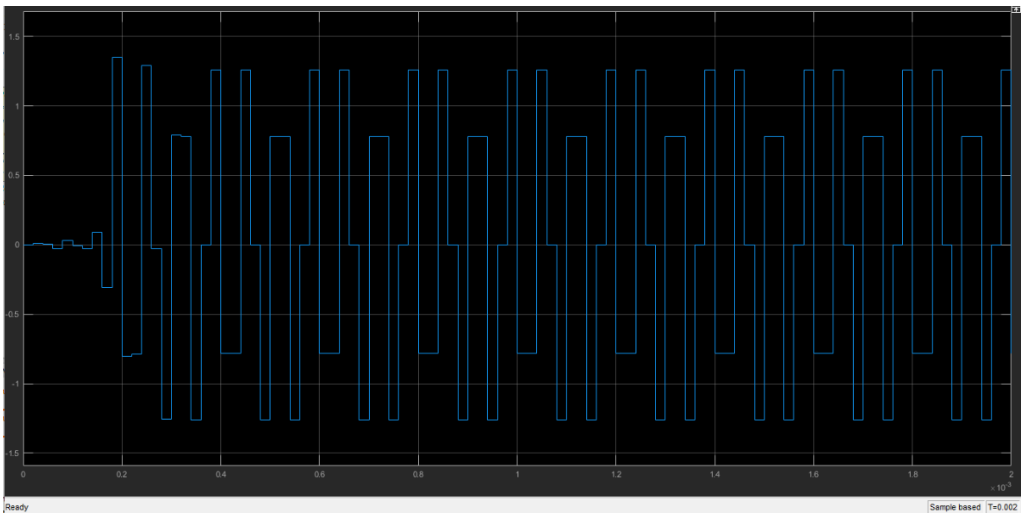


Ilustración 4: señal de entrada senoidal de 15 kHz

Se puede observar que en las respuesta a l filtro se produce un retardo cuando se encuentra en la muestra Ncoeff/2 +1, además de apreciarse como el número de muestras de la señal de salida son inversamente proporcionales a la frecuencia de la señal de entrada por lo que se obtiene una mayor precisión a menor frecuencia de entrada.

- **¿Por qué se ha escogido este tipo de arquitectura?**

Las arquitecturas secuenciales se emplean en casos en los que la frecuencia de muestreo (50 kHz) es menor a la frecuencia de reloj de operación (100 MHz). Tienen como ventaja el uso de menos recursos que operaciones tiene el algoritmo a implementar. En este caso se necesitan de c = 2000 ciclos de reloj para realizar todas las operaciones del algoritmo.

El esquema de la arquitectura del filtro secuencial es la siguiente:

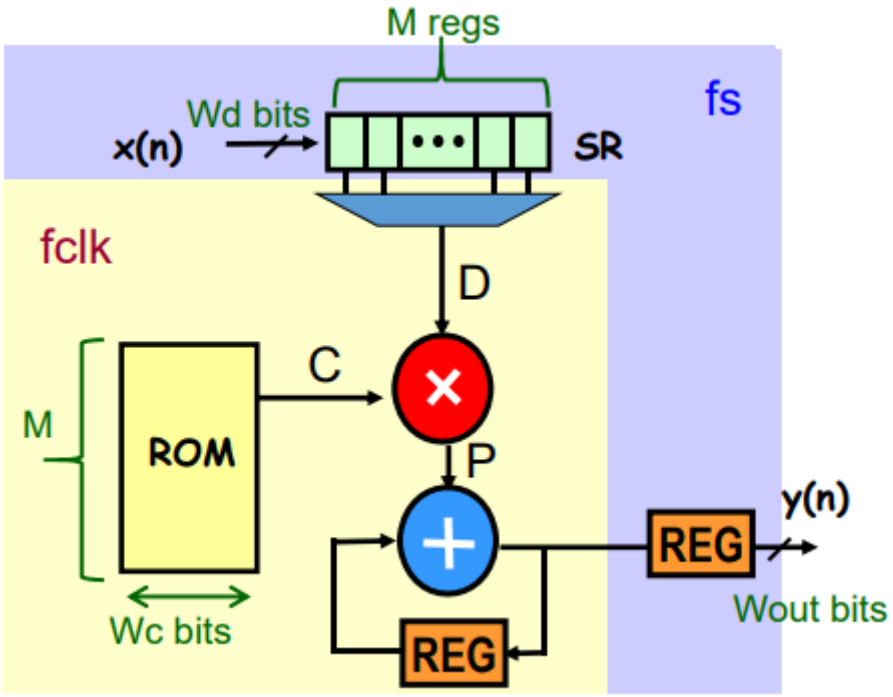


Ilustración 5: esquema filtro secuencial

Donde es M (17) es el número de registros, Wd (16) el número de bits de la entrada, Wc (18) el número de bits de los datos de la ROM y Wout (34) el número de bits de la salida del filtro secuencial.

Sección 2: Interfaz.

En la siguiente sección se define el interfaz, sus formatos y sus parámetros:

Módulo REG_MUX: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
ce	in	bit	Chip enable, habilita la carga de un nuevo dato
din	in	S[16,0]	Entrada de datos
sel	in	U[5,0]	Bus de dirección del registro
dout	out	S[16,0]	Salida de datos

Módulo REG_MULT_ACC: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
rst	in	bit	Reset síncrono, activo a nivel alto
coeff	in	S[18,0]	Señal coeficientes
din	in	S[16,0]	Entrada de datos
ce	in	bit	Chip enable, habilita la carga de un nuevo dato
dout	out	S[16,0]	Salida de datos

Módulo ROM: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
addr	in	U[5,0]	Direccionamiento de la memoria
data	out	S[18,0]	Salida de datos

Módulo CONTROL: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
rst	in	bit	Reset síncrono, activo a nivel alto
val_in	in	bit	Entrada de validación de la muestra de entrada
addr	out	S[16,0]	Salida dirección de datos
ce_Reg	out	bit	Chip enable, habilita la carga de un nuevo dato
rst_Acc	out	bit	Salida reset del acumulador
ce_Acc	out	bit	Salida chip enable del acumulador

Módulo SEC_FILTER: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
rst	in	bit	Reset síncrono, activo a nivel alto
val_in	in	bit	Entrada de validación de la muestra de entrada
din	in	S[16,0]	Entrada de datos
val_out	out	bit	Salida de validación de la muestra de salida
dout	out	S[18:0]	Salida de datos

Sección 3: Recursos hardware.

Logical Elements (LEs):

Se estiman los elementos lógicos utilizados teniendo en cuenta los bits utilizados en cada bloque del sistema y sus registros:

LEs (estimación REG_MUX): 16 bits de entrada registrados 17 veces = 17 x 17 = 272

LEs(estimación MULT_ACC): 16 bits de entrada y 34 bits de salida, asignación mult(34) asignación acc_out(34) y asignación dout(34) = 16 + 34 = 152

LEs (estimación ROM): 18 bits de salida = 18

LEs (estimación CONTROL): 4 salidas de un bit y registro de la señal state = 10

LEs (estimación SEC_FILTER): 16 bits de entrada, 34 bits de salida y truncado (15) = 65

LEs (estimación total) = 517

Se comprueba que el número estimado se asemeja a los LEs utilizados en el proyecto Quartus:

Estimated Total logic elements	548

M9Ks:

M9Ks(estimación)= Wc*2^(addr)*1(ROM) = 18*2^(5)*1 = 18 M9Ks.

Mults:

Nº multiplicadores = 1

Nº sumadores = 1

Total = 2

Se muestran a continuación los mults utilizados por el proyecto Quartus:

Embedded Multiplier 9-bit elements	2

Finalmente se muestra el sumario de recursos total del proyecto Quartus:

Resource	Usage
Estimated Total logic elements	548
Total combinational functions	256
Logic element usage by number of LUT inputs	
-- 4 input functions	192
-- 3 input functions	60
-- <=2 input functions	4
Logic elements by mode	
-- normal mode	219
-- arithmetic mode	37
Total registers	360
-- Dedicated logic registers	360
-- I/O registers	0
I/O pins	39
Embedded Multiplier 9-bit elements	2
Maximum fan-out node	clk~input
Maximum fan-out	360
Total fan-out	2095
Average fan-out	3.01

Ilustración 6: Sumario recursos proyecto Quartus

Sección 4: Frecuencia de operación.

Para la medida de la frecuencia máxima de operación se ha diseñado el módulo wrap registrando las distintas señales utilizadas en el módulo top del proyecto, para así simular todas las entradas que se incorporan al circuito en el mismo ciclo.

La frecuencia de operación del sistema obtenida mediante la herramienta Time Quest Timing analyzer es de 126.04 MHz. Se cumple por lo tanto la especificación de diseño para Fmax ≥ 125 MHz.

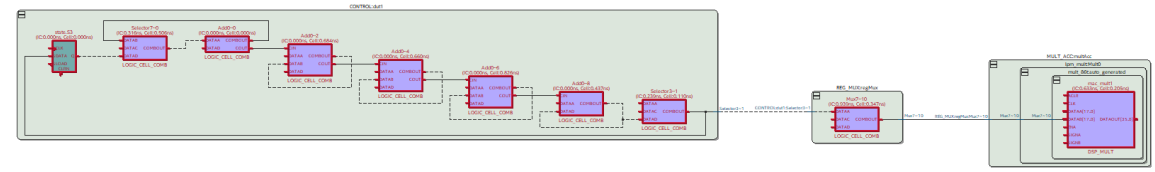
Slow 1200mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	126.04 MHz	126.04 MHz	clk	

El camino crítico se encuentra por el paso de la máquina de estados cuando se encuentra en el estado S3 como se muestra en las imagenes:

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.066	SEC_FILTER:sec_filter|CONTROL:dut1|state.S3	SEC_FILTER:sec_filter|dout_mux_reg[8]	clk	clk	8.000	0.071	7.808



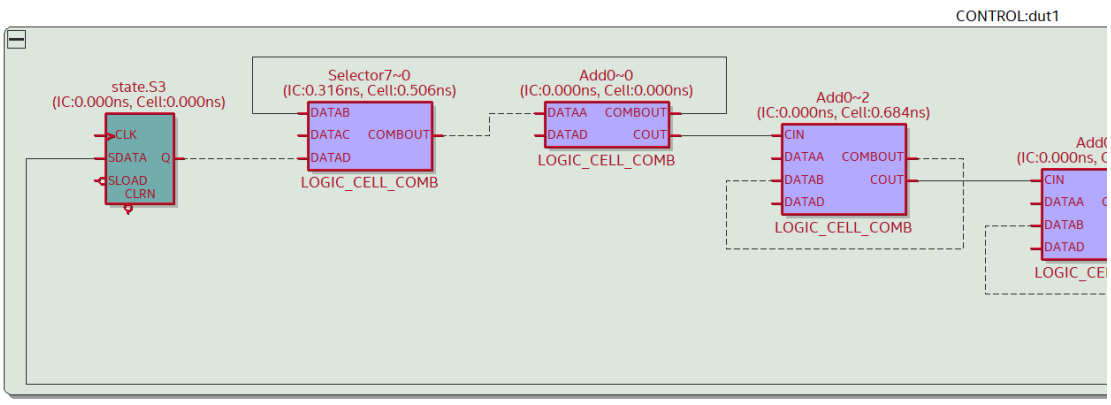


Ilustración 7: Visión RTL del camino crítico del proyecto

Sección 5: Verificación.

Bloque REG_MUX.v:

Este bloque funciona de modo que la señal ce (sel_reg1) habilita la carga del nuevo dato en la entrada (din), el cual se tiene en la posición del array de vectores reg_array(0) en el instante siguiente. A su vez, el resto de arrays de dicho vector se desplazan una posición hacia la derecha, eliminándose el array de la posición reg_array(16). El tamaño de este array viene determinado por el número de coeficientes del filtro. Por último, la salida dout apunta al vector en reg_array al que apunte sel_reg en ese instante.

En la simulación, se han introducido valores aleatorios a la entrada (din) y se ha observado el funcionamiento. se comprueba que el dato de entrada (din) en el instante en que se activa la señal de val_in, se almacena en reg_array en el siguiente instante, por lo que su índice (sel_reg1) será 0 en ese siguiente instante.

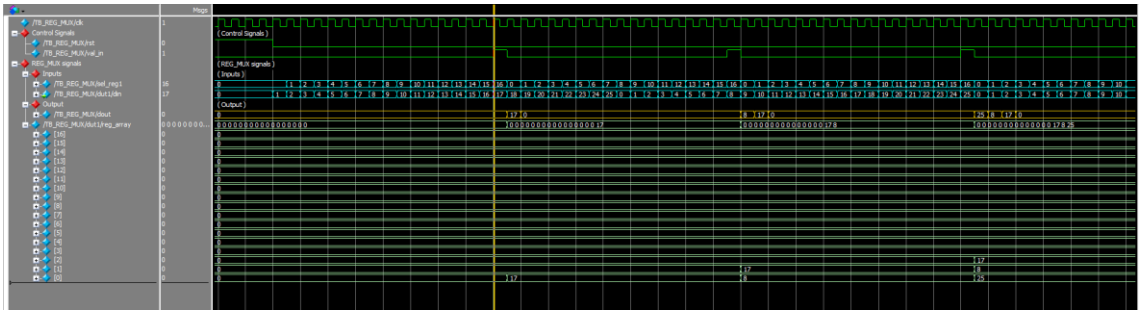


Ilustración 8: Formas de onda señales bloque REG_MUX

Bloque MULT_ACC.v:

Este bloque realiza la multiplicación de los coeficientes almacenados en la rom por la salida del bloque REG_MUX.v, y los acumula en un registro. Posee una señal de reset de acumulación (rst) síncrona y activa a nivel alto, y otra de habilitación del inicio de la acumulación cuando se tiene una nueva muestra (ce) activa a nivel alto.

Para su simulación, se han introducido valores aleatorios en la entrada (din = count en la imagen) y se ha comprobado su correcto funcionamiento. Se observa que el resultado de la acumulación se tiene un ciclo después de tener los datos en las entradas (din y coef).

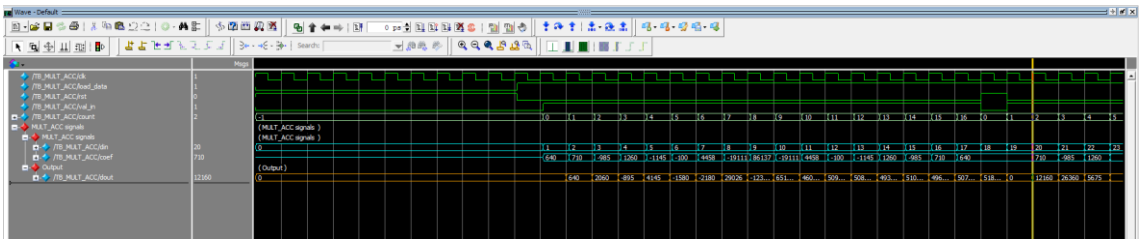


Ilustración 9: Formas de onda señales bloque MULT_ACC

Bloque ROM.v:

Este bloque trata de una memoria ROM, la cual almacena los 17 coeficientes del filtro a implementar. Su simulación resulta sencilla, cada ciclo de reloj se tiene el coeficiente de la rom al que apunta addr.

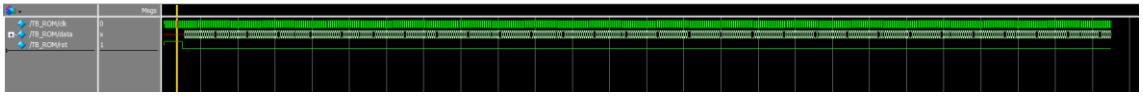


Ilustración 10: Formas de onda señales bloque ROM

Bloque CONTROL.v:

Este bloque es una máquina de estados que controla el funcionamiento del filtro con los bloques anteriores. Para su diseño se tienen en cuenta los retardos entre unos bloques y otros comentados en los anteriores apartados.

En la simulación obtenida, se tiene un estado S0 en el que se encuentra activa únicamente la señal de rst_acc (reset del acumulador), unos estados S1 y S2 entre los que se incrementa el addr (dirección para la rom y bit de selección del bloque REG_MUX) según se incrementan los ciclos hasta llegar al número de coeficientes. En esos estados, se habilita la señal de acumulación ce_acc (desactivando rst_acc). Una vez realizadas las correspondientes acumulaciones según el número de coeficientes, se activa la señal ce_reg, la cual registra el último valor de la acumulación siendo este la salida del sistema. Los índices de addr se han ajustado según ha sido necesario una vez simulado el sistema completo en el siguiente punto.

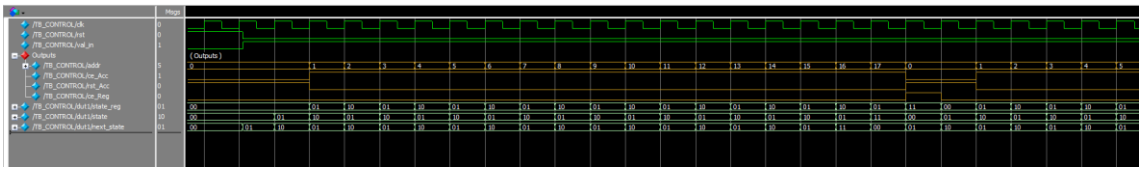


Ilustración 11: Formas de onda señales bloque CONTROL

Bloque SEC_FILTER.v:

Este bloque es el TOP del sistema. En él se tienen la señal de validación val_in, la cual indica dato válido en la señal de entrada (din) y estimula el resto de los bloques en consecuencia. También se tiene la señal val_out, que indica dato válido en salida Dout, y las señales de clk y rst.

Para la simulación, se tiene en cuenta que se tiene una muestra válida cada 2000 ciclos del reloj del sistema de 100 MHz, por lo que se tienen ciclos de sobra para el cálculo de la respuesta del filtro según lo programado en la máquina de estados.

Los resultados son los siguientes:

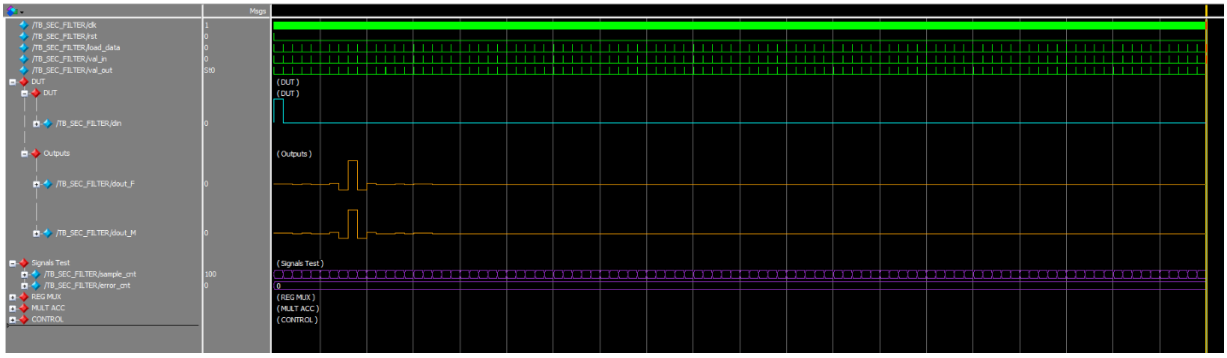


Ilustración 12: Formas de onda señales bloque SEC_FILTER senoidal respuesta al impulso

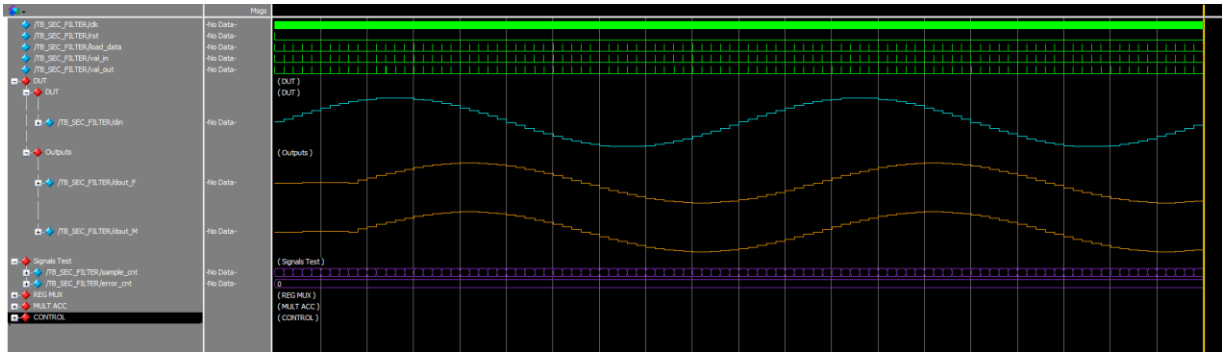


Ilustración 13: Formas de onda señales bloque SEC_FILTER senoidal 1 kHz

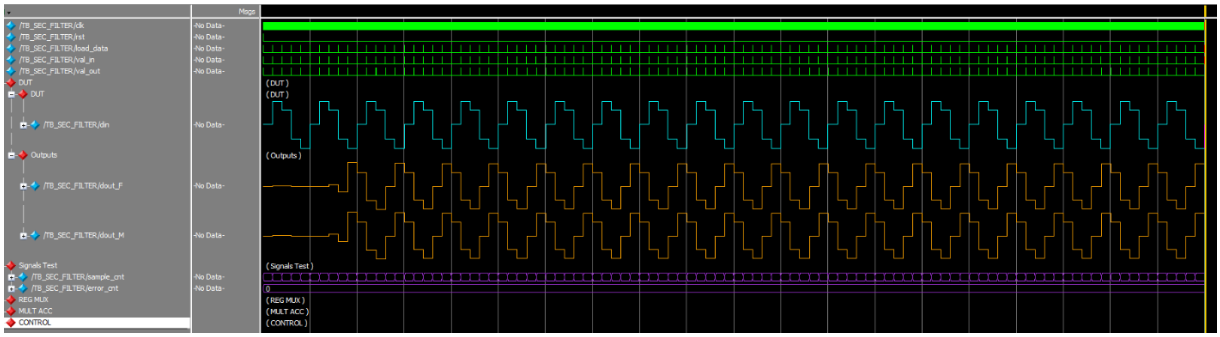


Ilustración 14: Formas de onda señales bloque SEC_FILTER senoidal 15 kHz

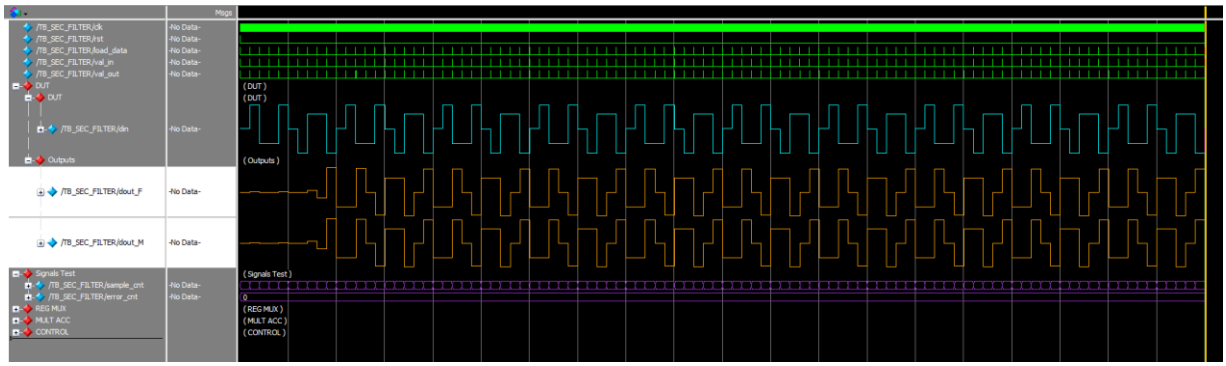


Ilustración 15: Formas de onda señales bloque SEC_FILTER senoidal 15 kHz

Se observa que el filtro funciona de forma correcta y se han cumplido los objetivos.

Sección 6: Resolución de problemas encontrados.

A continuación se presentan los problemas encontrados en la práctica y su solución:

-Para el diseño del módulo REG_MUX una vez implementado su archivo de test se podía observar como la secuencia de los registros no era como debía, el primer valor de la secuencia se trasladaba al final del array de salida cada vez que se producía un registro, de modo que no se proporcionaban los datos de forma correcta para que el multiplicador operara como corresponde. Para solucionar este problema se ha implementado un retardo a la salida de modo que los valores salen en el orden correcto y sincronizados con la entrada del módulo MULT_ADD.

-En el diseño de la máquina de estados CONTROL al realizar su verificación se podía observar como el comportamiento de los estados no se correspondía con el comportamiento de la señal addres requerido, para ello se ha registrado la señal de control de estados “state” para sincronizar el comportamiento de ambas partes.

-Una vez realizada la compilación del proyecto en Quartus no arrojaba en el sumario de compilación el uso de ningún elemento de memoria por lo que no se ha adjuntado la imagen correspondiente en su estimación.