Memoria Práctica E3
Filtro Interpolador: CIC

David Martínez Esteso

Néstor García García

Tabla de contenido.

Sección 1: Descripción del módulo.

El módulo a desarrollar e implementar se muestra a continuación:

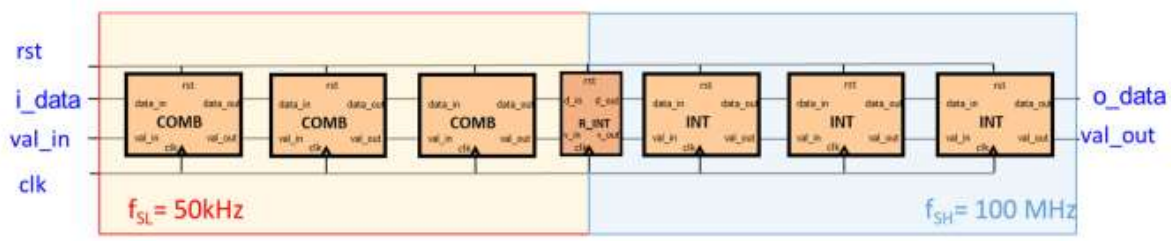

Ilustración 1: Esquema filtro interpolador CIC

El módulo se trata de un filtro interpolador por 2000 , el cuál cambia la tasa de muestreo de la señal de 50 kHz a 100 MHz. En este caso se trata de un filtro CIC de tres etapas, es decir, se compone de tres etapas peine, de un módulo expansor y generador de la señal de validación y de tres filtros integradores en este orden para la entrada y salida de la señal.

- ¿Qué es un filtro CIC?

El filtro CIC es un tipo de filtro FIR utilizado para el procesamiento de señales digitales con diferentes frecuencias de muestreo para diezmar e interpolar señales digitales. El filtro CIC consta de uno o más pares de filtros de peine integral. En un CIC interpolado la señal de entrada pasa a través de un filtro peine, muestreo ascendente (expansor) y por el mismo número de pasos de integración que el número de peines.

- ¿Cuál es la ganancia de un filtro CIC interpolador?

La ganancia viene dada por la siguiente expresión:

$G = (R*M)^N *1/R$

Siendo M el orden del filtro peine (1), R el factor de interpolación (2000) y N el número de etapas.

En este caso la ganancia del filtro será 4 000 000 (132.04 dB).

- ¿Para qué lo estamos utilizando en nuestro proyecto? ¿Qué ventajas introduce frente a otras posibles soluciones?

El hecho de que se escoja la implementación de un filtro tipo CIC es debido a la necesidad de incrementar la frecuencia de trabajo de 50 kHz a 100 MHz para el correcto funcionamiento del modulador FM-AM.

La ventaja del filtro CIC es el menor uso de recurso para su implementación con respecto a otros tipos de filtro, además no se utilizan multiplicadores (los coeficientes valen 1).

Cuantificación adecuada:

Teniendo en cuenta que la ganancia del filtro es de 4 000 000, necesitamos 22 bits ($Wg = 22 = \log_2(4000000)$) para evitar que produzca desbordamiento o haya pérdidas a la salida de las etapas peine y a la salida del filtro.

Simulaciones de comportamiento:

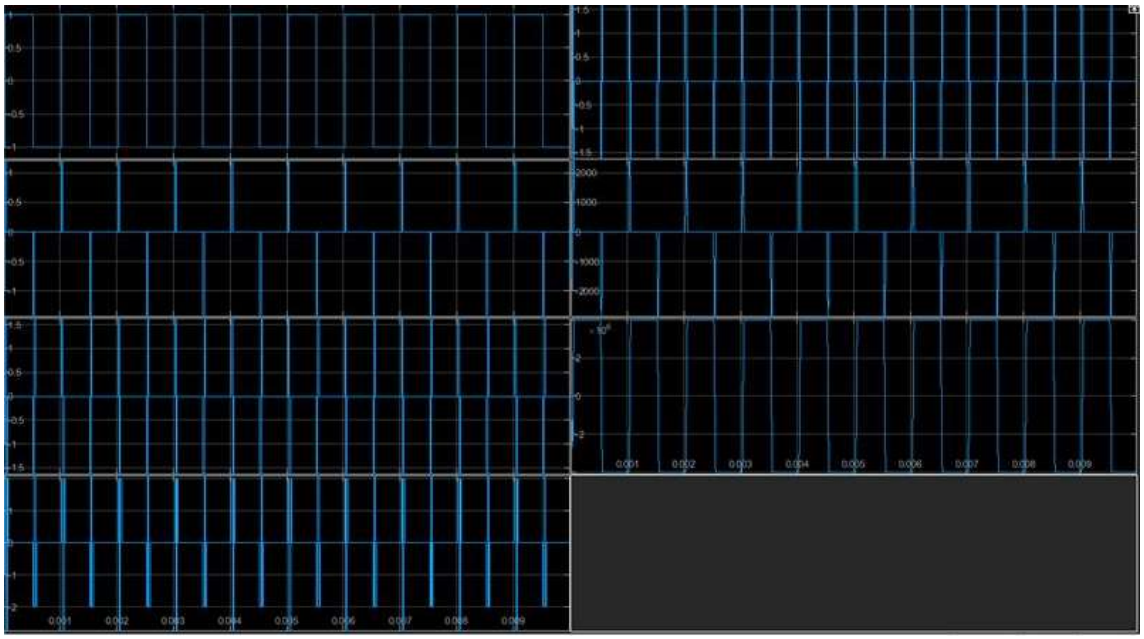

Ilustración 2: Señal de entrada cuadrada de 1kHz

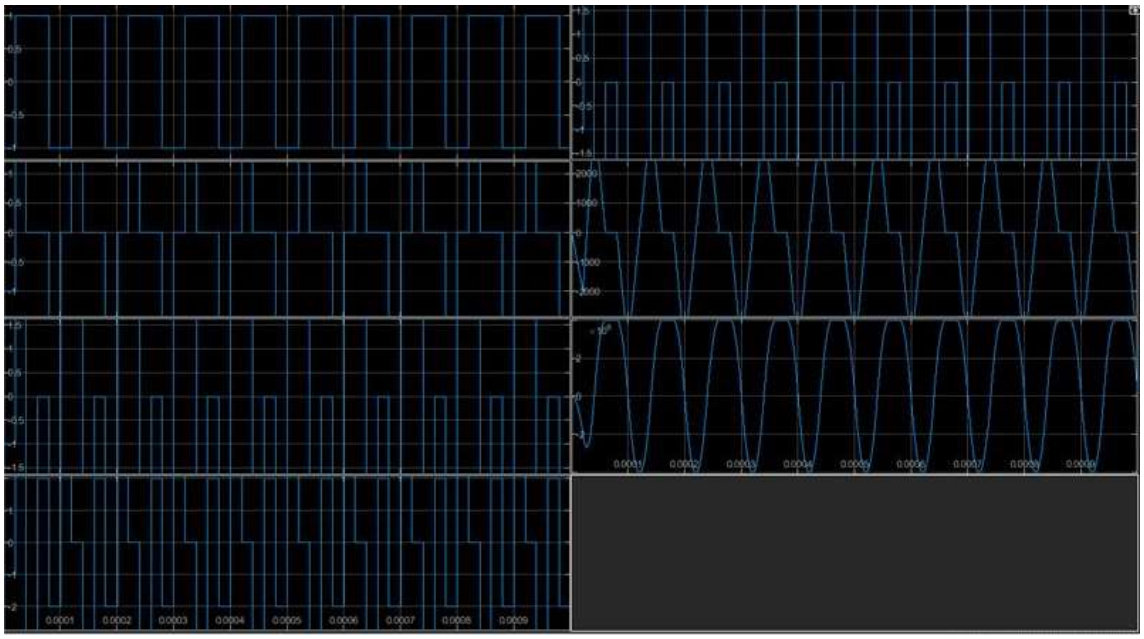

Ilustración 3: Señal de entrada cuadrada de 10kHz

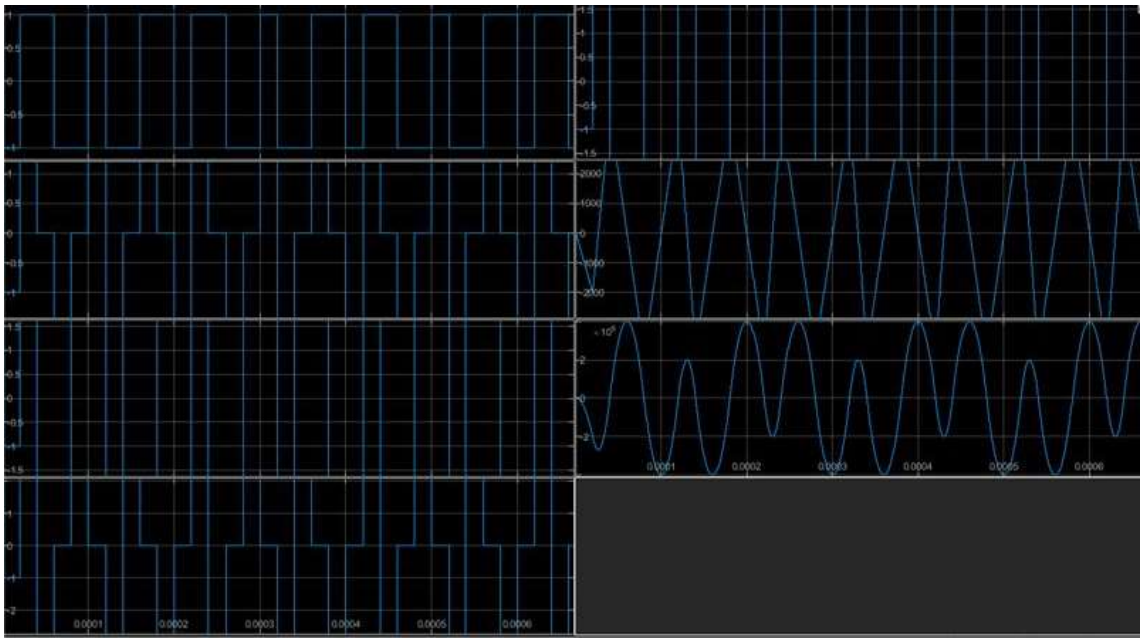

Ilustración 4: Señal de entrada cuadrada de 15kHz

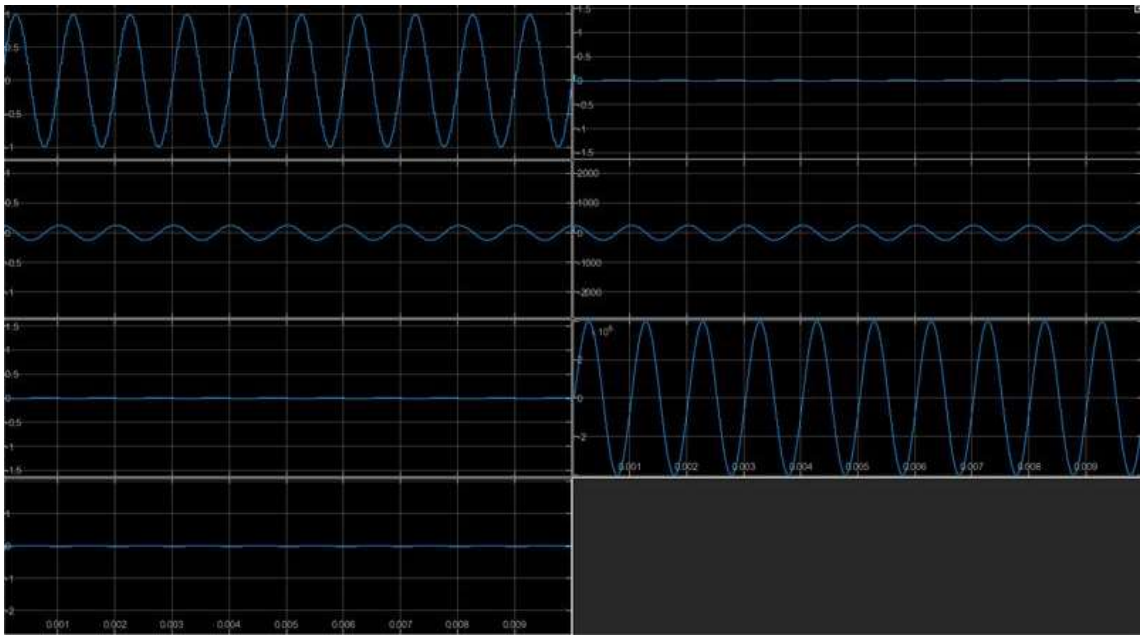

Ilustración 5: Señal de entrada senoidal de 1kHz

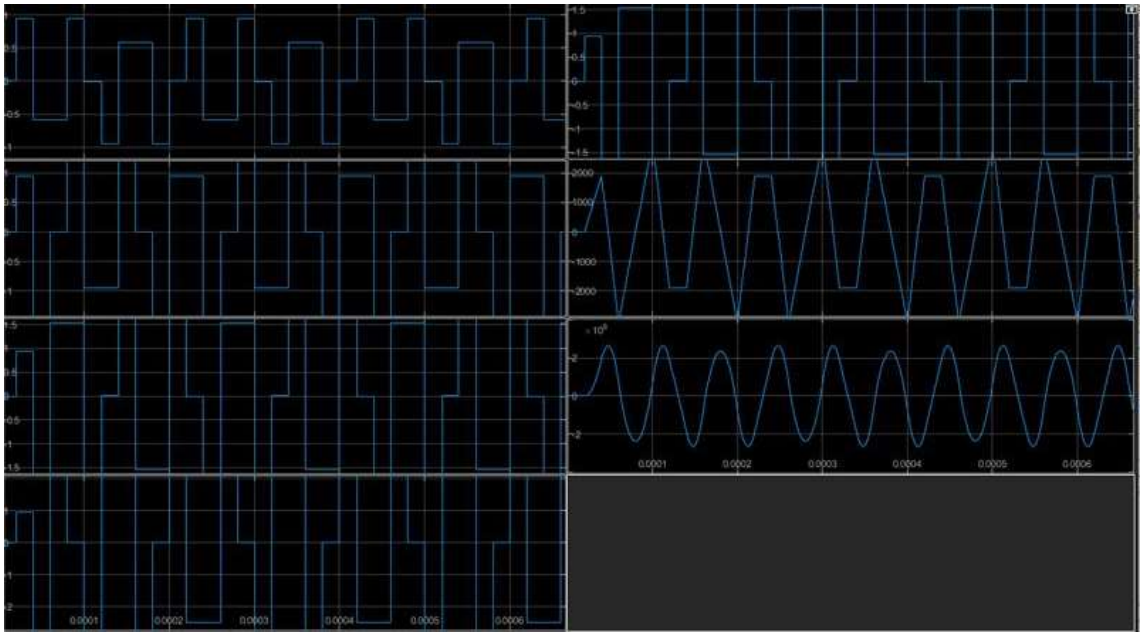

Ilustración 6: Señal de entrada senoidal de 15kHz

Tras las diferentes simulaciones con las señales de entrada, se puede ver 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.

Sección 2: Interfaz.

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

Módulo CIC: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
rst	in	bit	Reset síncrono, activo a nivel alto
i_data	in	S [16,15]	Entrada de datos a modular
val_in	in	bit	Entrada de validación de la muestra de entrada
o_data	out	S [16,15]	Salida de datos modulados
val_out	out	bit	Salida de validación de la muestra de salida

Módulo COMB / R_INT / INT: INTERFAZ			
Nombre	Tipo	Formato	Descripción
clk	in	bit	Entrada de reloj
rst	in	bit	Reset síncrono, activo a nivel alto
data_in	in	S [16,15]	Entrada de datos del modulo
val_in	in	bit	Entrada de validación de la muestra de entrada
data_out	out	S [16,15]	Salida de datos del modulo
val_out	out	bit	Salida de validación de la muestra de salida

Sección 3: Recursos hardware.

Logical Elemtents (LEs):

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

LEs(estimación) =

16 bits x 7 módulos x 2(entrada y salida) = 224

Registros módulos = 3 regs x 3(COMBs) + 2 regs (R_INT) + 2 regs x 3(INTs) = 17

Total LEs (estimación) = 241

Memorias M9Ks:

En el proyecto desarrollado no se emplean memorias.

Mults:

En este caso no se han requerido del uso de ningún tipo de estructura mult. Característica propia de los filtros CIC.

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

	Resource	Usage
1	Estimated Total logic elements	249
2		
3	Total combinational functions	249
4	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	0
2	-- 3 input functions	181
3	-- <=2 input functions	68
5		
6	⌄ Logic elements by mode	
1	-- normal mode	87
2	-- arithmetic mode	162
7		
8	⌄ Total registers	244
1	-- Dedicated logic registers	244
2	-- I/O registers	0
9		
10	I/O pins	36
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	rst~input
15	Maximum fan-out	249
16	Total fan-out	1568
17	Average fan-out	2.78

Ilustración 7: Sumario recursos proyecto Quartus

Se puede observar que el número de elementos lógicos utilizados por el proyecto Quartus es de 249 además de no utilizar ningún elemento tipo mult, concordando en buena medida con los valores estimados anteriormente.

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 278.94 MHz, restringida a 250 MHz ya que es la máxima frecuencia a la que opera la altera cyclone IV. Se cumple por lo tanto la especificación de diseño para Fmax ≥ 125 MHz.

Slow 1200mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	278.94 MHz	250.0 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

El camino crítico se encuentra en el paso del bloque R_INT.v al primer bloque integrador:

Slow 1200mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-2.585	CIC:cic|CIC_pc:cic1|R_INT:RINT|data_out[3]	CIC:cic|CIC_pc:cic1|INT:int1|data_out[37]	clk	clk	1.000	-0.071	3.509

Sección 5: Verificación.

La verificación del sistema se ha llevado a cabo mediante la realización de varios testbench. En primer lugar, se han generado los ficheros de texto necesarios procedentes del modelo de ejemplo (golden model) del fichero proyecto Matlab proporcionado. Utilizado los datos de estos ficheros se han desarrollado códigos Verilog para cada módulo individual, es decir para los módulos COMB, R_INT, INT y CIC (top).

Etapa peine (COMB.v):

Para el test de la etapa peine, se ha tomado como modelo de ejemplo el primer bloque peine de la simulación en Matlab. Por tanto, se han creado 2 ficheros de texto desde Matlab, uno correspondiente a la entrada de dicho bloque y otro a la salida. Para la realización del test, simplemente se ha introducido la señal de validación de entrada (val_in) y una vez se han tenido datos válidos en la salida (val_out), se han comparado la salida de nuestro bloque con la salida del bloque de ejemplo, obteniendo las formas de onda de abajo.

Forma de onda de etapa peine COMB:

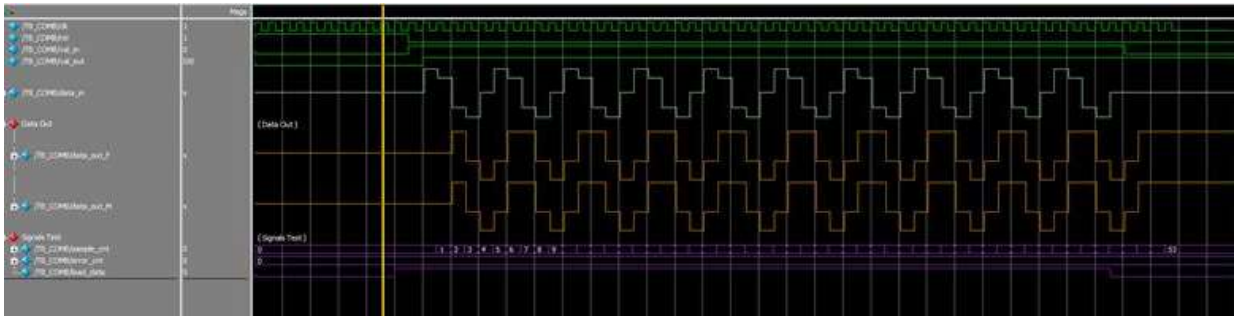

Ilustración 7: Formas de onda etapa peine COMB

Etapa integrador (INT.v):

Para el test de la etapa integrador, se ha tomado como modelo de ejemplo el segundo bloque peine de la simulación en Matlab, de forma arbitraria. Al igual que con el test del bloque peine, se han creado 2 ficheros de texto desde Matlab, uno correspondiente a la entrada de dicho bloque y otro a la salida. Para la realización del test, se ha introducido la señal de validación de entrada (val_in) y una vez se han tenido datos válidos en la salida (val_out), se han comparado la salida de nuestro bloque con la salida del bloque de ejemplo, obteniendo las formas de onda de abajo.

Forma de onda de etapa filtro integrador INT:

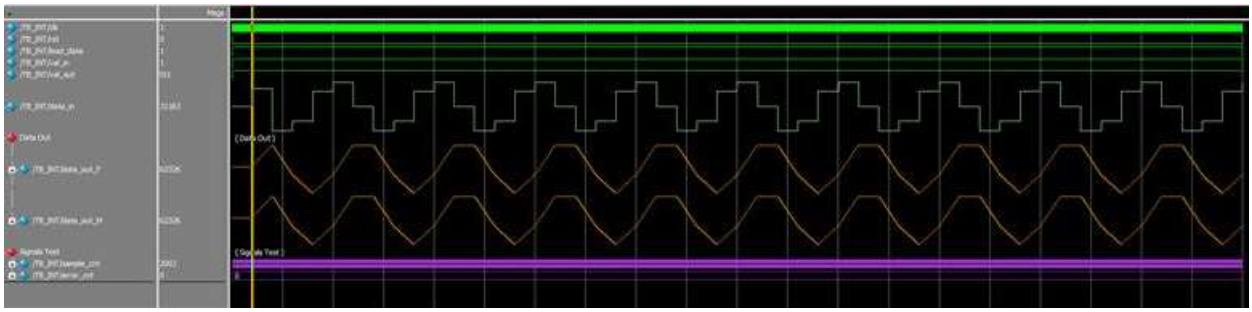

Ilustración 8: Formas de onda etapa filtro integrador INT

Etapa expansor (R_INT.v):

La función de este bloque es expandir la salida (introduciendo R-1 ceros entre muestras) y generar la señal de habilitación de los filtros integradores para conseguir un factor de interpolación R, en este caso 2000.

Para ello, simplemente se ha hecho que este bloque saque en su salida los datos en su entrada cuando la señal de dato válido (val_in) así lo marque, en caso contrario, sacará cero. A su vez la salida val_in se ha gestionado en el test bench para que sea activa cada 1999 ciclos, simulando así la frecuencia a la que trabaja la entrada de este bloque procedente de las etapas peine. Por último, la salida val_out se pone a 1 una vez llega el primer dato válido (val_in).

Forma de onda del módulo filtro expansor R_INT:

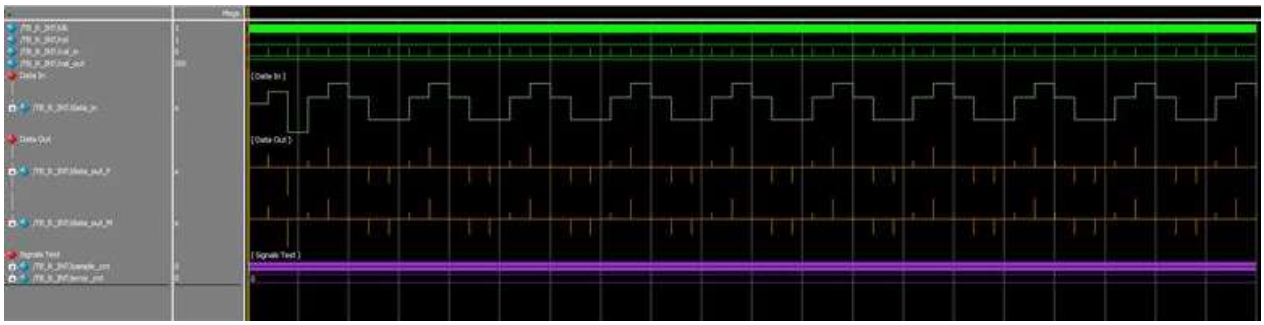

Ilustración 9: Formas de onda filtro expansor R_INT

Sistema completo:

El sistema completo tiene 3 bloques COMB seguidos de un bloque R_INT y 3 bloques INT. Para su testbench, se ha realizado de forma similar al testbench del bloque R_INT.v, donde val_in se activa cada 2000 ciclos y, tras pasar por el bloque expansor, las señales de validación de los siguientes bloques se mantienen constantes. Se ha simulado precisión completa para onda senoidal y cuadrada, y salida truncada para la onda cuadrada. Los resultados obtenidos se muestran a continuación.

Forma de onda filtro interpolador CIC completo onda senoidal:

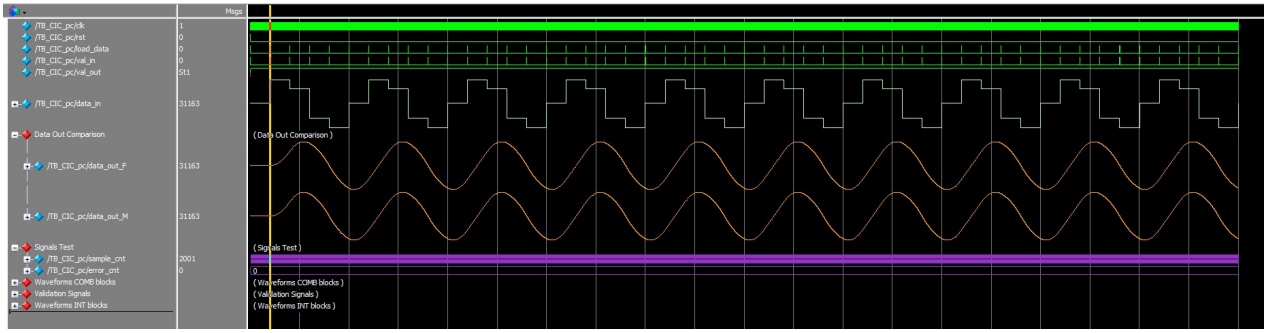

Ilustración 10: Formas de onda filtro interpolador CIC completo, onda senoidal

Forma de onda filtro interpolador CIC completo onda cuadrada:

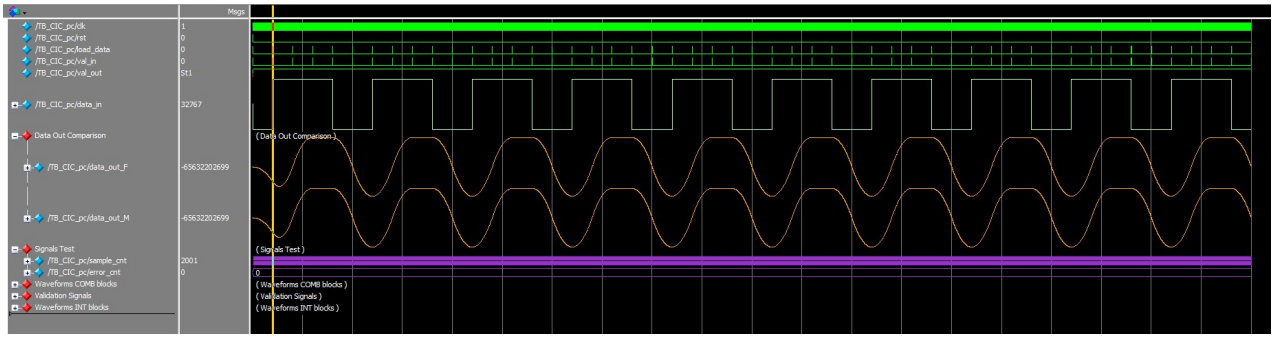

Ilustración 11: Formas de onda filtro interpolador CIC completo, onda cuadrada

Forma de onda filtro interpolador CIC truncado a 16 bits onda senoidal:

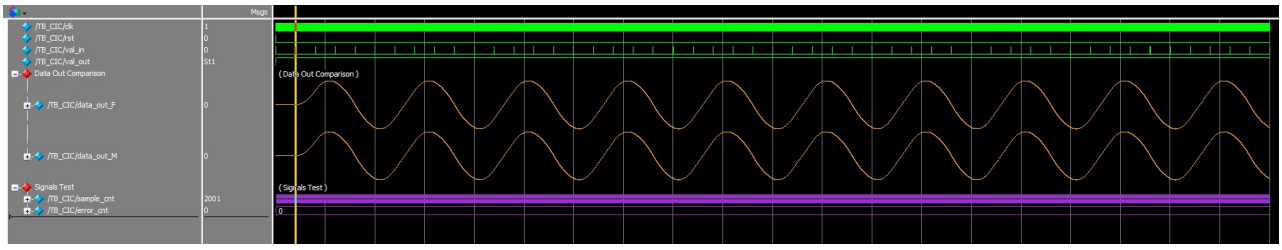

Ilustración 12: Formas de onda filtro interpolador CIC completo, onda senoidal truncada

Sección 6: Resolución de problemas encontrados.

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

-Durante el diseño del filtro expansor no lográbamos insertar los ceros entre muestras repitiéndose la muestra anterior durante los 2000 ciclos siguiente. Para solucionar este problema empleamos la señal load_data para decirnos en que caso no se está recogiendo ningún dato de las etapas peine y en ese caso obtener como salida los 0 necesarios durante los 1999 ciclos siguientes.

- En el momento en que mostrábamos la simulación del filtro completo hubo un momento en el que tan solo nos mostraba la forma de onda para la primera etapa peine, este era debido a que la señal registrada data_in_reg no se mostraba activa en el momento correspondiente. Posterior a esto sucedía un problema similar a este con los filtros integradores y la señal val_in.

- Por último una vez se había conseguido representar la señal senoidal los errores que mostraba entre muestras del fichero Matlab y el fichero con el código verilog estaban desfasadas entre sí. Para solucionar este problema ha sido necesario registras la señal de entrada reset tanta veces como fuera necesario para sincronizar el valor de las muestras a comparar.

Sección 7: Conclusiones.

Se ha implementado un filtro CIC en Verilog, con el objetivo de aumentar la tasa de muestras en un factor 2000, diferenciando entre unas etapas peine trabajando a 50 kHz, y unas etapas integrador, trabajando a 100 MHz.

Se ha comprobado que su utilización requiere de menos recursos en comparación a otro tipo de filtros. Cabe destacar que su implementación no requiere de multiplicadores, por lo que la frecuencia máxima obtenida es muy alta.