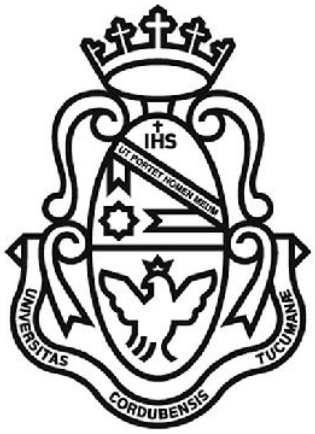


UNIVERSIDAD NACIONAL DE CÓRDOBA
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES
CARRERA INGENIERÍA ELECTRÓNICA

PROYECTO INTEGRADOR PARA LA OBTENCIÓN DEL TÍTULO DE GRADO INGENIERO ELECTRÓNICO

"FILTRO INTERPOLADOR DIGITAL PARA PLATAFORMA SDR BASADA EN FPGA"

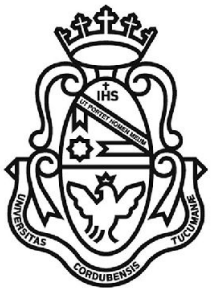
Alumno
RAFAEL ALMENDRA, Ariane Zain

Director
Dra. Ing. CORRAL BRIONES, Graciela

Co-Director
Ing. AYARDE, Juan Martin

Córdoba, República Argentina
Julio/2017

Tribunal Evaluador



UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales
Escuela de Ingeniería Electrónica

El Tribunal Evaluador reunido en éste acto y luego de haber aprobado la Solicitud de Aprobación de Tema y efectuado las distintas instancias de correcciones del Informe del Proyecto Integrador para la obtención del Título de Grado "Ingeniero Electrónico" y cumpliendo con el Reglamento correspondiente, declaran el Informe Final de la estudiante: **Ariane Zain RAFAEL ALMENDRA** como "aceptado sin correcciones" y la defensa oral Aprobada. Por lo tanto, luego de haber tenido en cuenta los aspectos de evaluación que indica el Reglamento, el Proyecto Integrador se considera Aprobado.

Se firma el Acta de Examen correspondiente y se distribuyen los ejemplares impresos.

Firma y aclaración del tribunal evaluador

Fecha:

NOTA:

Dedicatoria

Dedicado a mi madre que siempre me empujó hacia adelante cuando ya no tenía fuerzas, a mi padre que siempre apoyó mis proyectos, a la Dra. Lucatelli por su dedicación desinteresada y a mis hermanos y todos mis amigos que me ayudaron a levantarme cada vez que caí.

Ninguna persona es una isla, sin ustedes no habría llegado a esta meta.

Agradecimientos

Quiero agradecer a todo el personal del Laboratorio de Comunicaciones Digitales que más allá de ayudarme constantemente con cualquier dificutad que pudiera encontrar a lo largo del desarrollo del presente trabajo integrador me enseñaron lo que es trabajar en equipo y disfrutar el trabajo de cada día.

Gracias por la paciencia a lo largo de mi período de aprendizaje y la calidad humana presente.

Un logro más por parte de todo el equipo de trabajo del Laboratorio de Comunicaciones Digitales.

Resumen

La creciente penetración en el mercado de comunicaciones inalámbricas en banda ancha y el dinamismo con el cual van surgiendo nuevos estándares de comunicación dan origen al concepto de una plataforma de radio definida por software (SDR). La visión de una radio definida por software (SDR) es implementar en software diferentes sistemas de comunicaciones empleando una misma plataforma de hardware. Un sistema avanzado de SDR consta de dos componentes fundamentales: un componente de RF programable y otro componente reconfigurable basado en FPGA encargado de realizar el procesamiento de alta velocidad de señales. Puesto que la tasa de muestreo de los conversores ADC y DAC es generalmente fija y muy superior a las tasas requerida para el procesamiento en banda base de diferentes sistema de comunicaciones, los SDRs emplean módulos programables que realizan el cambio de tasa de procesamiento en tiempo real.

He aquí que el filtro CIC de Hogenauer cobra vital importancia, gracias a su capacidad de reconfigurarse para varios valores de cambio de tasa, ancho de banda y atenuación deseados. Es así como hoy en día puede apreciarse su utilización en receptores de RF de banda L, como los que nos presenta Pentek [1], en sus modelos de radio definida por software.

El presente trabajo tiene como objetivo la construcción de un filtro digital que realice la interpolación y filtrado de la señal a transmitir. Se desarrolla utilizando una metodología iterativa incremental recursiva. Se comienza con un estudio y repaso de conceptos de procesamiento digital de señales para poder encarar un primer caso de uso sencillo de la problemática planteada, validando e implementando el mismo en FPGA. Luego, se desarrolla un modelo de hasta 6 etapas del filtro digital en punto flotante y punto fijo con la capacidad de reconfiguración mediante variables. Éste modelo multietapa es sometido a una nueva verificación mediante simulación y se lo valida mediante un banco de prueba encargado de excitar al mismo mediante entrada impulsiva, senoidal y de ruido de distribución uniforme.

Área Temática y Asignaturas

Área Temática: Digitales, Comunicaciones.

Asignaturas: Teoría de Señales y Sistemas Lineales, Teoría de las Comunicaciones, Electrónica Digital.

Palabras Claves

FPGA - *Field Programmable Gate Arrays*, SRC - *Sampling Rate Conversion*, DUC - *Digital Up Converter*, CIC - *Cascaded Integrator Comb Filter*, SDR - *Software Defined Radio*

Abstract

The growing penetration of the broadband wireless communications in the market and the dynamism with which new standards of communication are emerging give rise to the concept of a software-defined radio platform (SDR). The vision of a software defined radio (SDR) is to implement in software different systems of communications using the same platform of hardware. An advanced SDR system consists of two fundamental components: a programmable RF component and another reconfigurable FPGA-based component in charge of performing high-speed signal processing. Since the sampling rate of the ADC and DAC converters is generally fixed and well above the rates required for the baseband processing of different communications systems, the SDRs employ programmable modules that perform in real-time the rate change processing .

Hence, the Hogenauer CIC filter is vitally important, thanks to its ability to reconfigure itself for several desired rate, bandwidth and attenuation values. This is how it can be seen today in L-band RF receivers, such as those presented by Pentek [1], in their software-defined radio models.

The present work aims at the construction of a digital filter that performs the interpolation and filtering of the signal to be transmitted. It is developed using an iterative incremental recursive methodology. It begins with a study and review of concepts of digital signal processing to be able to face a first case of simple use of the problem raised, validating and implementing the same in FPGA. Then, a model of up to 6 stages of the digital filter is developed in floating point and fixed point with the capacity of reconfiguration through variables. This multistage model is subjected to a new verification by means of simulation and it is validated by means of a test bench in charge of exciting it by impulsive, sinusoidal and uniform distribution noise input.

Key Words

FPGA - *Field Programmable Gate Arrays*, SRC - *Sampling Rate Conversion*, DUC - *Digital Up Converter*, CIC - *Cascaded Integrator Comb Filter*, SDR - *Software Defined Radio*

Índice de tablas

2.1. Atenuación en la banda de paso para grandes factores de cambio de tasa 19
2.2. Atenuación de las imágenes para grandes factores de cambio de tasa . . . 19
2.3. Requerimientos para el filtro interpolador 23
2.4. Resultados de implementación del Filtro Interpolador CIC N=3,M=1 y R=10 compensado con un filtro FIR de 48 coeficientes 24
2.5. Resultados de implementación del Filtro Interpolador Polifásico R=10 de 430 coeficientes 24

5.1. Resultados de medición temporal de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal 59
5.2. Resultados de medición de atenuación de la 1º,2º y 3º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal . . . 60
5.3. Resultados de medición de atenuación de la 4º,5º y 6º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal . . . 60
5.4. Resultados de medición de atenuación de la 7º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal 60
5.5. Recursos utilizados 73

Índice de figuras

2.1. Proceso de interpolación 7
2.2. Señales en el tiempo y frecuencia de un interpolador L=3 8
2.3. Filtro FIR de 12 coeficientes 9
2.4. Coeficientes de filtro FIR usados para calcular $x_1(0)$ 9
2.5. Estructura interpoladora de filtro polifásico (L=3) mediante bancos de subfiltros FIR 10
2.6. Estructura interpoladora de filtro polifásico (L=3) de mínimo almacenamiento mediante conmutadores de coeficientes 10
2.7. Filtros Promediadores 12
2.8. Filtro CIC Interpolador de 1 etapa 14
2.9. Filtro CIC Interpolador de N etapas 15
2.10. Respuesta en el dominio del tiempo de un filtro CIC N=1, M=2, R=2 . . 17
2.11. Respuestas de un filtro CIC N=1, M=2 y R=2 18
2.12. Diagrama de ceros y polos de un filtro CIC N=1, M=1, R=8 20
2.13. Diagrama de ceros y polos de un filtro CIC N=1, M=2, R=8 20
2.14. Comparación de la salida de un filtro CIC N=1, R=8 para valores de M=1 y M=2 21
2.15. Comparación de las salidas de un filtro CIC R=8, M=2 para N=1 y N=4 22
2.16. Respuesta del filtro Interpolador CIC compensado 23
2.17. Respuesta de ambas arquitecturas de filtro interpolador 25

3.1. Filtro CIC Interpolador de N etapas 28
3.2. Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada impulsiva de amplitud 511, simulada en punto flotante 32
3.3. Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada impulsiva de amplitud 511, simulada en punto flotante 32
3.4. Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada senoidal de f=781.25Khz, A=0.5, phi=0,offset=0, simulada en punto flotante 33
3.5. Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada senoidal de f=781.25Khz, A=0.5, phi=0,offset=0, simulada en punto flotante 34
3.6. Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada de ruido de media 0 y varianza 1, simulada en punto flotante 35

3.7. Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada de ruido de media 0 y varianza 1, simulada en punto flotante 36

4.1. Diagrama de bloques de un filtro comb 38
4.2. Diagrama de bloques del cambio de tasa 38
4.3. Diagrama de bloques de un integrador 38
4.4. Modularización del la sección comb del filtro interpolador CIC 39
4.5. Modularización de la sección integradora del filtro interpolador CIC . . . 39
4.6. Diagrama de bloques de la salida del filtro interpolador CIC 39
4.7. Diagrama de bloques de entradas del filtro interpolador CIC 40
4.8. Diagrama de bloques del generador de entrada impulsiva 41
4.9. Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada impulsiva de amplitud 511 simulada en punto fijo . . . 42
4.10. Diagrama de bloques del generador de onda senoidal 43
4.11. Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada senoidal de amplitud 15, frecuencia 781,25 KHz, sin desfasaje ni offset de continua, simulada en punto fijo 44
4.12. Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada de ruido con media 0 y varianza 1 simulada en punto fijo 45
4.13. Diagrama de bloques del generador de ruido de distribución uniforme . . 46
4.14. Diseño de Altera de un filtro interpolador CIC R=75, M=1 y N=3 utilizado para la verificación el filtro interpolador CIC propio diseñado 48
4.15. Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada impulsiva de amplitud 511 49
4.16. Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada senoidal de amplitud 15, de frecuencia 390,625 Khz y de offset de fase de $\frac{\pi}{4}$ 50
4.17. Acercamiento de la salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada senoidal de amplitud 15, de frecuencia 390,625 Khz y de offset de fase de $\frac{\pi}{4}$ 51
4.18. Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada de ruido de media 0 y varianza 1 52
4.19. Acercamiento de la salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada de ruido de media 0 y varianza 1 53

5.1. Generación de tren de impulsos para validación en FPGA Cyclone II EP2C35F672C6 55
5.2. Medición del período de la señal de entrada del tren de impulsos 55
5.3. Medición del ancho del pulso de la señal de entrada del tren de impulsos 56
5.4. Medición del ancho del pulso de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada de tren de impulsos 56
5.5. Medición de la separación entre pulsos de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada de tren de impulsos 57

5.6. Medición del periodo de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=3 y entrada de tren de impulsos . . 57
5.7. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 1 período en LUT . 61
5.8. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 1 período en LUT . 61
5.9. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 2 períodos en LUT 62
5.10. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 2 períodos en LUT 62
5.11. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 4 períodos en LUT 63
5.12. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 4 períodos en LUT 63
5.13. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 8 períodos en LUT 64
5.14. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 8 períodos en LUT 64
5.15. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 16 períodos en LUT 65
5.16. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 16 períodos en LUT 65
5.17. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 32 períodos en LUT 66
5.18. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 32 períodos en LUT 66
5.19. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 64 períodos en LUT 67
5.20. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 64 períodos en LUT 67
5.21. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 128 períodos en LUT 68
5.22. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 128 períodos en LUT 68
5.23. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 256 períodos en LUT 69
5.24. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 256 períodos en LUT 69
5.25. Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 512 períodos en LUT 70
5.26. Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 512 períodos en LUT 70
5.27. Salida del filtro interpolador CIC sintetizada para R=8, M=1 y N=1 . . 71
5.28. Salida del filtro interpolador CIC sintetizada para R=8, M=2 y N=1 . . 72

A.1. Filtro CIC Decimador e Interpolador 94

A.2. Ejemplo de la respuesta en frecuencia de un filtro CIC con los parámetros N=4, M=1, R=7 y $f_c = \frac{1}{8}$ 95
A.3. Diagrama de Gannt 97

Lista de Símbolos y Convenciones

CIC Cascade Integrator Comb. 3, 4, 6, 8, 12–16, 23, 28, 29, 33, 74, 75

DSP procesamiento digital de señales. 4

DTFT transformada de Fourier de tiempo discreto. 7

FCEFyN Facultad de Ciencias Exactas, Físicas y Naturales. 5

FIR Finite Impulse Response. 6, 8, 11, 12, 15

FPGA Field Programmable Gate Array. 3–5, 37, 43, 45, 54, 74

HDL Hardware Description Language. 5, 37

LCD Laboratorio de Comunicaciones Digitales. 3–5

LUT Look Up Table. 58

RF radiofrecuencia. 2

SDR radio definida por software. 2, 3

SeCyT Secretaría de Ciencia y Tecnología. 2

UNC Universidad Nacional de Córdoba. 2

Índice general

Tribunal Evaluador ii

Dedicatoria iii

Agradecimientos iv

Resumen v

Área Temática y Asignaturas v

Palabras Claves v

Abstract vi

Key Words vi

Lista de tablas vii

Lista de figuras viii

Lista de Símbolos y Convenciones xii

1. Introducción **2**
- 1.1. Marco de investigación . . . 2
  - 1.1.1. Objetivo Principal . . . 3
  - 1.1.2. Objetivos Secundarios . . . 3
- 1.2. Metodología de Investigación . . . 3
- 1.3. Herramientas de trabajo . . . 4
  - 1.3.1. Herramientas de software . . . 4
  - 1.3.2. Herramientas de hardware . . . 5

2. Marco teórico **6**
- 2.1. Concepto de Interpolación . . . 6
- 2.2. Arquitecturas de filtros interpoladores . . . 8
  - 2.2.1. Filtros Polifásicos . . . 8
  - 2.2.2. Filtros CIC . . . 11
- 2.3. Filtro Interpolador CIC . . . 14
  - 2.3.1. Descripción . . . 14

2.3.2. Aplicaciones 23

3. Diseño en punto flotante **26**
3.1. Algoritmos de implementación 26
3.1.1. Filtro comb 26
3.1.2. Cambio de tasa 27
3.1.3. Filtro integrador 27
3.1.4. Multietapa 28
3.2. Verificación por simulación 29
3.2.1. Entrada impulsiva 31
3.2.2. Entrada de onda senoidal 31
3.2.3. Entrada de ruido de distribución uniforme 35

4. Diseño en punto fijo **37**
4.1. Algoritmos de implementación 37
4.1.1. Filtro comb 37
4.1.2. Cambio de tasa 38
4.1.3. Filtro integrador 38
4.1.4. Multietapa 38
4.2. Verificación por simulación 40
4.2.1. Entrada impulsiva 41
4.2.2. Entrada de onda senoidal 43
4.2.3. Entrada de ruido de distribución uniforme 45
4.2.4. Verificación comparativa 47

5. Validación en FPGA **54**
5.1. Entrada Impulsiva 54
5.1.1. Hardware utilizado 54
5.1.2. Generación de tren de impulsos 54
5.1.3. Mediciones 55
5.2. Entrada Senoidal 58
5.2.1. Hardware utilizado 58
5.2.2. Generación de onda senoidal 58
5.2.3. Mediciones 58
5.3. Entrada de Ruido de distribución uniforme 71
5.3.1. Hardware utilizado 71
5.3.2. Generación de ruido de distribución uniforme 71
5.3.3. Mediciones 71
5.4. Recursos utilizados 73

6. Conclusión y trabajos futuros **74**
6.1. Conclusión 74
6.2. Debilidades 75
6.3. Mejoras y trabajos futuros 75

Bibliografía **77**

A. Scripts de MATLAB **79**
A.1. main . . . 79
A.2. config . . . 80
A.3. Simulation Booleans . . . 82
A.4. Input Selector . . . 83
A.5. Calculo Seno . . . 83
A.6. Funciones . . . 84
A.6.1. Normalize frequency . . . 84
A.6.2. Input Config . . . 84
A.6.3. Impulse . . . 85
A.6.4. Sin . . . 85
A.6.5. Noise . . . 86
A.6.6. Comb Filter . . . 86
A.6.7. Integrator Filter . . . 86
A.6.8. Rate Change Factor . . . 87
A.6.9. Floating Point Simulation . . . 87
A.6.10. Word Width . . . 88

Anexos del Proyecto Integrador **90**
Solicitud de Aprobación de Tema . . . 91
Nota de Aprobación Final del Director . . . 99

Capítulo 1

Introducción

En años recientes, la gran demanda por tecnologías de comunicación inalámbrica dieron como resultado el surgimiento de nuevos estandares de comunicación. Ésto trajo consigo un incremento en los requerimientos en cuanto a la flexibilidad de los receptores de interoperar con todos estos estandares. En este contexto surge el concepto de radio definida por software (SDR) que es una radio que puede reconfigurar sus características (demodulación, corrección de errores, etc.) de acuerdo al tipo de estandard de comunicación con el que se está tratando de interactuar.

El concepto de *radio* no significa literalmente FM/AM, representa cualquier tipo de comunicación que utilice como medio de transmisión el aire como puede ser la televisión o la telefonía celular.

Para realizar el procesamiento digital de alta velocidad de las etapas próximas a los DACs/ADCs se recurre a desarrollar el sistema en plataformas de hardware reconfigurables evitando la poca maniobrabilidad que ofrece un hardware definido. Es por ello que una FPGA resulta ideal al momento de implementar un diseño SDR, como en el presente trabajo.

El presente proyecto integrador se centra en el procesamiento digital de la señal a transmitir antes de incursionar en la etapa de radiofrecuencia (RF). Se encarga de realizar un aumento en la frecuencia de muestreo de la señal a transmitir, acondicionando la señal para la posterior amplificación y filtrado para transmitir por antena.

En este primer capítulo se presenta el marco metodológico, las herramientas de trabajo empleadas y los objetivos a los que se desea llegar en el desarrollo del presente proyecto integrador.

1.1. Marco de investigación

El presente trabajo integrador se enmarca dentro de un proyecto de diseño de sistemas de comunicaciones con múltiples antenas que cuenta con el financiamiento de la Secretaría de Ciencia y Tecnología (SeCyT)-Universidad Nacional de Córdoba (UNC).

Este proyecto integrador se realizó dentro del grupo de investigación de Comunicaciones Inalámbricas del Laboratorio de Comunicaciones Digitales (LCD), que aportó todos los conocimientos y recursos necesarios que permitieron la realización de este trabajo.

1.1.1. Objetivo Principal

Construir un filtro digital que realice la interpolación y filtrado de la señal a transmitir, con una arquitectura reconfigurable en cuanto al ancho de banda y a la conversión de tasas de muestreo en un amplio rango de valores.

1.1.2. Objetivos Secundarios

- Comprender los fundamentos de los filtros Cascade Integrator Comb (CIC)
- Realizar un modelo de simulación en punto flotante y verificar su desempeño con los resultados analíticos.
- Implementar una arquitectura digital y verificar su desempeño con los resultados obtenidos de la simulación en punto flotante
- Investigar los conceptos básicos de procesamiento digital de señales.
- Generar un modelo de simulación en punto flotante.
- Generar un modelo de simulación en punto fijo.
- Implementar un caso de uso de entrada impulsiva en una FPGA y medir los resultados.
- Implementar un caso de uso de entrada senoidal en una FPGA y medir los resultados.
- Simular la respuesta del filtro para entrada impulsiva, senoidal y de ruido uniforme.
- Publicar los resultados y generar un informe final.

1.2. Metodología de Investigación

El modelo de proceso utilizado para desarrollar este trabajo es iterativo incremental. Cada etapa tuvo un tiempo previo de investigación y aprendizaje. A continuación se describe cada una de ellas:

- Estudio de conceptos básicos de procesamiento digital de señales y filtros digitales.

  Esta primera etapa se inicia para tomar los conocimientos necesarios sobre el área de SDR y procesamiento digital de señales. Se contó con un libro introductorio del área [2], del cual se fueron realizando ejercicios en MATLAB.

- Estudio de conceptos básicos de filtros CIC.

  Como segunda etapa tuvo lugar el estudio del filtro CIC mediante el paper [3], desmenuzando el problema en micro simulaciones para entender su comportamiento.

- Diseño de modelo para una sola etapa en punto flotante y punto fijo.

  Luego de comprender el comportamiento del filtro CIC, se procedió a modelar el mismo mediante algoritmos en MATLAB y en DSP Builder para una sola etapa verificando la concordancia con el análisis teórico previo.

- Implementación y medición en FPGA del modelo de una sola etapa.

  Como parte de la verificación del modelo se procedió a realizar una prueba con entrada impulsiva en una placa DE0 [4]. También se realizaron pruebas para entrada senoidal y de ruido uniforme en una placa DE2 [5].

- Parametrización de ambos modelos para múltiples etapas.

  Una vez verificado el modelo para una sola etapa, se procedió a modularizar el mismo para múltiples etapas.

- Verificación mediante testbenchs.

  Se generaron diferentes pruebas de simulación para verificar el comportamiento del nuevo modelo de múltiples etapas para entradas impulsiva, senoidal y de ruido uniforme. Se comparó el desempeño del filtro con un ejemplo de filtro interpolador CIC diseñado por Altera Corporation.

- Estudio de resultados.

  La última etapa fue el estudio y publicación de los resultados obtenidos, los cuales fueron analizados de distintas maneras y puntos de vista.

1.3. Herramientas de trabajo

1.3.1. Herramientas de software

Todas las herramientas de software fueron instaladas para un entorno de Windows 7 y sus respectivas licencias fueron otorgadas por el LCD para realizar íntegramente el proyecto integrador.

- MATLAB: Robusta herramienta de diseño de modelos de ingeniería en general. Se utiliza para aprendizaje automático, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica y muchos otros campos [6].

- SIMULINK: Herramienta de entorno de programación visual que funciona sobre el entorno de programación MATLAB. Es ampliamente usado en Ingeniería Electrónica en temas relacionados con el procesamiento digital de señales (DSP), involu-

crando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros. También es muy utilizado en Ingeniería de Control y Robótica [7].

- DSP Builder: DSP Builder para Intel® FPGAs es una herramienta de diseño para DSP que permite la generación de código HDL de algoritmos de DSP directamente desde un entorno de SIMULINK. Esta herramienta permite diseñar algoritmos, establecer la tasa de datos deseada, frecuencia del clock, y ofrece en dispositivos la simulación exacta de bit y ciclos, sintetizando HDL optimizado de punto fijo y punto flotante, autoverificado en ModelSim - Intel FPGA software y autoverificado/cosimulado en hardware. Esta herramienta añade librerías junto a las existentes en SIMULINK [8], [9], [10].
- Quartus Prime: es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en Hardware Description Language (HDL) [11].

1.3.2. Herramientas de hardware

Dentro de las herramientas de hardware nos encontramos: por un lado con las placas FPGA, donde se realizaron las pruebas de implementación, suministradas por el LCD y, por otro lado, con las distintas herramientas de medición y conectores utilizados que fueron suministrados por el pañol de Electrónica de la Facultad de Ciencias Exactas, Físicas y Naturales (FCEFyN).

- Fuente variable en tensión y corriente.
- Osciloscopio digital Yokogawa DL1520L.
- Analizador de espectro Signal Hound USB-SA44B.
- Placa FPGA Cyclone IV EP4CE22F17C6 de desarrollo DE0 Nano [4].
- Placa FPGA Cyclone II EP2C35F672C6 de desarrollo DE2 [5].
- Adaptador VGA/BNC.
- Puntas banana, cocodrilo y BNC para los intrumentos anteriormente listados.

Capítulo 2

Marco teórico

Los filtros CIC son implementaciones computacionalmente eficientes de filtros pasabajos de reducido ancho de banda que están habitualmente embebidos en implementaciones de hardware de decimación e interpolación en sistemas de comunicaciones modernos [12].

Son una clase de filtros con Finite Impulse Response (FIR) que consisten en una o varias etapas derivadoras y una o varias etapas integradoras en cascada, interconectadas a través de un interpolador o decimador en su estructura central. Se usan comunmente en sistemas de procesamiento digital de señales multi-tasa donde se requiere un alto factor de cambio del periodo de muestreo [13].

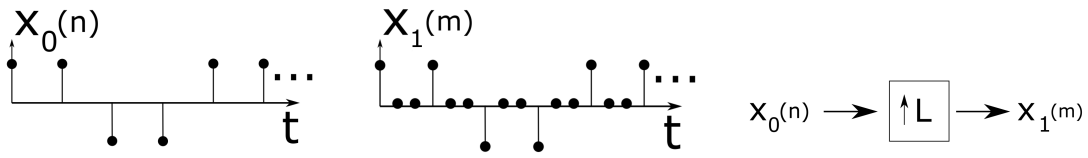
A lo largo de este capítulo se presenta el marco teórico que guía el presente trabajo integrador. Se describe el proceso de interpolación de una señal y se desarrollan dos arquitecturas de interpolación, la arquitectura polifásica y la arquitectura CIC. Luego se describe el comportamiento de un filtro interpolador CIC para finalmente realizar una comparación de utilización de recursos en la implementación en hardware de cada una de estas arquitecturas.

2.1. Concepto de Interpolación

En el area del procesamiento digital de las señales, interpolar significa generar valores intermedios de una nueva curva que pasa a su vez sobre los valores previamente muestreados. Esto se logra mediante el agregado de L-1 ceros entre muestras y su posterior filtrado, lo cual es equivalente a muestrear una señal analógica limitada en banda una cantidad de tiempo L más pequeña [14].

Sea $x_0(n)$ una señal de tiempo discreto y $x_1(m)$ la misma señal sobremuestreada, la ecuación

$$x_1(m) = \begin{cases} x_0\left(\frac{m}{L}\right) & \text{si } m = 0, \pm(L), \pm(2L), ... \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.1}$$



(a) Señal en el tiempo antes de la interpolación

(b) Señal en el tiempo interpolada con L=3

(c) Diagrama de bloques del proceso de interpolación

Figura 2.1: Proceso de interpolación

representa numéricamente cómo se conforma muestra a muestra $x_1(m)$. La Figura 2.1a muestra la gráfica de la señal de entrada $x_0(n)$ en el tiempo, la Figura 2.1b muestra la gráfica de la señal de salida $x_1(m)$ para un sobremuestreo L=3 y la Figura 2.1c el diagrama de bloques que representa el sobremuestreo. Se puede observar que $x_1(m)$ coincide con $x_0(n)$ para todos los valores múltiplos enteros de L.

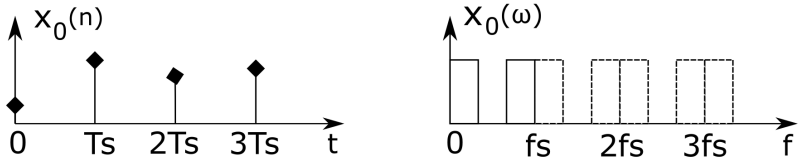
Ahora bien, en el dominio de la frecuencia, ¿Qué implicancia tiene sobremuestrear una señal? Sea $x_0(n)$ una señal digital muestreada a f_s y $x_1(m)$ su respectiva señal sobremuestreada a una tasa $f_{s'} = Lf_s$. La transformada de Fourier de tiempo discreto (DTFT) de $x_1(m)$ es [15]:

$$\begin{aligned} X_1(\mathrm{e}^{j\omega}) &= \sum_{m=-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} x_0(n)\delta[m-nL] \right) \mathrm{e}^{(-j\omega m)} \\ &= \sum_{n=-\infty}^{\infty} x_0(n)\mathrm{e}^{(-i\omega Ln)} \\ &= X_0(\mathrm{e}^{j\omega L}) \end{aligned} \tag{2.2}$$

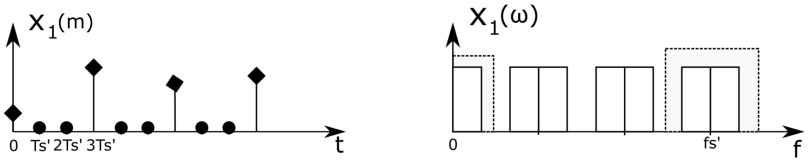
Por lo que, luego de interpolar la señal, se aprecia la aparición de *imágenes* centradas en múltiplos de f_s, como se ilustra en la Figura 2.2b.

Sea $x_0(n)$ la señal de entrada ilustrada en la Figura 2.2a, y $x_1(m)$ su respectiva señal sobremuestreada, el último paso en el proceso de interpolación es el filtrado de $x_1(m)$ con un filtro digital pasabajo llamado *filtro interpolador* que se encarga de eliminar las imágenes espectrales. Idealmente este filtro debe tener una respuesta en frecuencia como se ilustra en líneas de puntos en la Figura 2.2b. En la práctica no es posible alcanzar tal respuesta, por lo que a la salida del filtro se obtiene una señal como la observada en la Figura 2.2c.

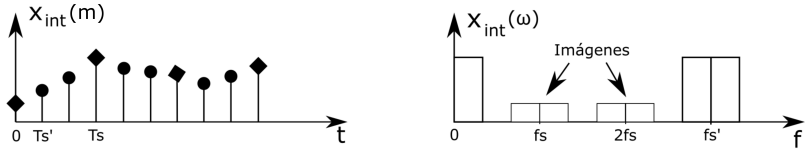
La precisión de todo el proceso de interpolación depende de la atenuación en la banda de rechazo del filtro interpolador pasabajo. Mientras mejor sea esta atenuación, más precisa será la interpolación. Además, debido al agregado de muestras nula, se obtiene una pérdida de amplitud de valor L. Por ende, para obtener una ganancia unitaria entre las secuencias $x_0(n)$ y $x_{int}(m)$, el filtro interpolador debe tener una ganancia de valor L.



(a) Señal en el tiempo y la frecuencia antes de la interpolación



(b) Señal en el tiempo y la frecuencia luego de la interpolación



(c) Señal en el tiempo y la frecuencia luego del filtrado de las imágenes

Figura 2.2: Señales en el tiempo y frecuencia de un interpolador L=3

2.2. Arquitecturas de filtros interpoladores

Para la implementación y análisis de un filtro interpolador se introducen 2 arquitecturas, una basada en los filtros polifásicos y otra basada en los filtros CIC. En la actualidad existen otras arquitecturas disponibles, como por ejemplo los filtros de Farrow, pero no es competencia del presente proyecto integrador ahondar en estas otras [16].

2.2.1. Filtros Polifásicos

Para llegar a la arquitectura final de este filtro se parte suponiendo que para el filtrado de la señal $x_1(m)$ se dispone de un filtro FIR de 12 coeficientes como se muestra en la Figura 2.3.

Para obtener la señal $x_{int}(m)$ a salida del filtro, se convoluciona la respuesta impulsiva del filtro $h(k)$ con la secuencia de valores de la señal de entrada $x_1(m)$, la cual dispone de L-1 ceros entre cada muestra. Estos ceros no aportan información en la construcción de $x_{int}(m)$, por lo que sólo es necesario considerar los valores no nulos de la secuencia $x_1(m)$ que constituyen en realidad la secuencia sin interpolar $x_0(n)$. De esa forma para conformar la nueva señal $x_{int}(0)$, sólo se usa 4 coeficientes del filtro:

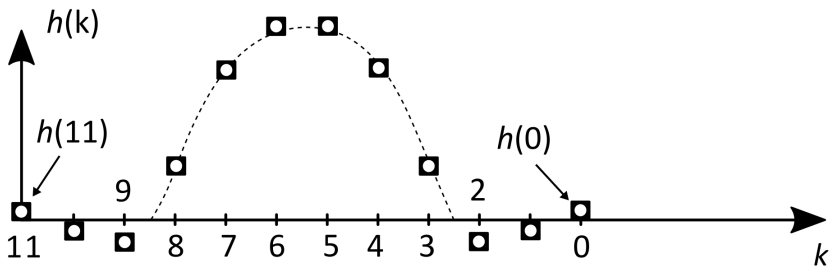


Figura 2.3: Filtro FIR de 12 coeficientes

$h(11), h(8), h(5)$ y $h(2)$ como puede apreciarse en la Figura 2.4.

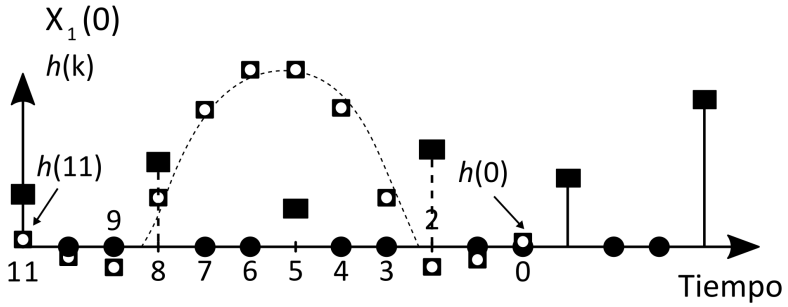


Figura 2.4: Coeficientes de filtro FIR usados para calcular $x_1(0)$

De la misma manera, para obtener la siguiente señal de salida $x_{int}(1)$, se desplaza la respuesta impulsiva del filtro hacia la derecha y sólo se hace uso de $h(9), h(6), h(3)$ y $h(0)$. Si se vuelve a desplazar hacia la derecha la respuesta impulsiva del filtro, sólo se hace uso de $h(10), h(7), h(4)$ y $h(1)$. Se puede observar un patrón, hay L=3 diferentes conjuntos de coeficientes usados para computar $x_{int}(m)$, ya que si se desplaza una vez más la respuesta impulsiva del filtro, para obtener la próxima salida, se vuelve a hacer uso de los mismos coeficientes que para $x_{int}(0)$. Entonces, cada valor de $x_{int}(m)$ se calcula de la forma:

$$\begin{aligned}
x_{int}(0) =& h(2)x_0(3) + h(5)x_0(2) + h(8)x_0(1) + h(11)x_0(0) \\
x_{int}(1) =& h(0)x_0(4) + h(3)x_0(3) + h(6)x_0(2) + h(9)x_0(1) \\
x_{int}(2) =& h(1)x_0(4) + h(4)x_0(3) + h(7)x_0(2) + h(10)x_0(1) \\
x_{int}(3) =& h(2)x_0(4) + h(5)x_0(3) + h(8)x_0(2) + h(11)x_0(1) \\
x_{int}(4) =& h(0)x_0(5) + h(3)x_0(4) + h(6)x_0(3) + h(9)x_0(2) \\
x_{int}(5) =& h(1)x_0(5) + h(4)x_0(4) + h(7)x_0(3) + h(10)x_0(2) \\
& \ldots
\end{aligned} \tag{2.3}$$

Se demuestra entonces, que no es necesario generar la secuencia $x_1(m)$, esto es una ventaja

del *filtrado polifásico* ya que permite economizar recursos para el procesamiento de la señal.

La Figura 2.5 presenta el *modelo conmutador* para filtros de interpolación polifásicos. El conmutador realiza una rotación completa con la llegada de cada valor de entrada $x_0(n)$. Por lo que, por cada muestra a la entrada, se calculan 3 nuevos valores de $x_{int}(m)$. De forma general, se necesitan L subfiltros, siendo L el factor de interpolación.

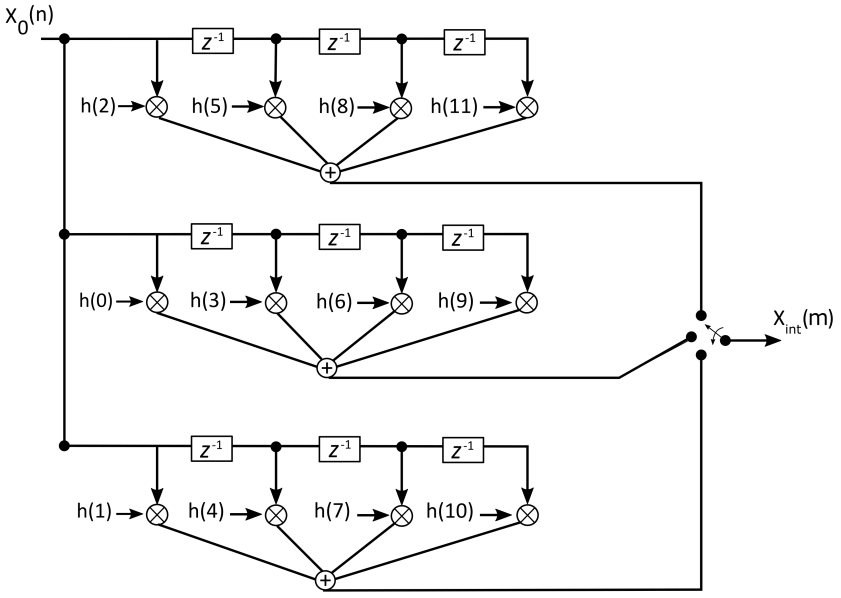


Figura 2.5: Estructura interpoladora de filtro polifásico (L=3) mediante bancos de subfiltros FIR

Agregando 4 conmutadores adicionales, 1 para cada conjunto de coeficientes, se prescinde de la necesidad de 8 registros de almacenamiento. Se presenta, en la Figura 2.6, una estrucutra de mínimo almacenamiento donde cada conmutador adicional agregado gira completamente, rotando a través de cada set de coeficientes, con la llegada de cada muestra de x_0.

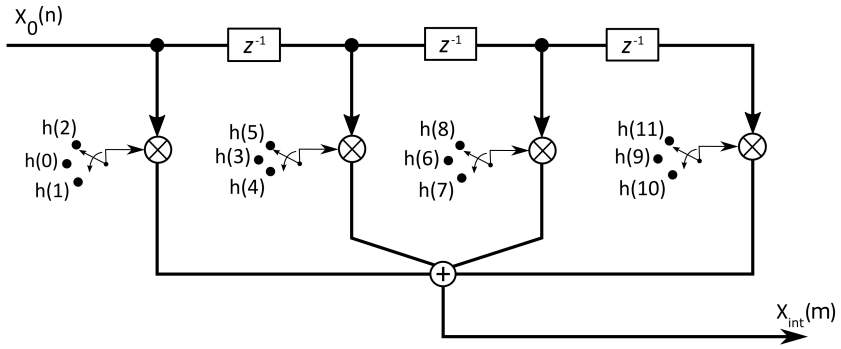


Figura 2.6: Estructura interpoladora de filtro polifásico (L=3) de mínimo almacenamiento mediante conmutadores de coeficientes

Dicha estructura de mínimo almacenamiento se describe de forma general como:

$$\begin{aligned} H(z) =& h(2) + h(5)z_{in}^{-1} + h(8)z_{in}^{-2} + h(11)z_{in}^{-3} \\ &+[h(0) + h(3)z_{in}^{-1} + h(6)z_{in}^{-2} + h(9)z_{in}^{-3}]z_{out}^{-1} \\ &+[h(1) + h(4)z_{in}^{-1} + h(7)z_{in}^{-2} + h(10)z_{in}^{-3}]z_{out}^{-2} \end{aligned} \tag{2.4}$$

donde z_{in}^{-1} es un retardo a la tasa de muestreo de entrada, y z_{out}^{-1} es un retardo a la tasa de muestreo de salida. Con L=3, $z_{in}^{-1} = z_{out}^{-3}$, $z_{in}^{-2} = z_{out}^{-6}$ y $z_{in}^{-3} = z_{out}^{-9}$, entonces:

$$\begin{aligned} H(z) =& h(2) + h(5)z_{out}^{-3} + h(8)z_{out}^{-6} + h(11)z_{out}^{-9} \\ &+[h(0) + h(3)z_{out}^{-3} + h(6)z_{out}^{-6} + h(9)z_{out}^{-9}]z_{out}^{-1} \\ &+[h(1) + h(4)z_{out}^{-3} + h(7)z_{out}^{-6} + h(10)z_{out}^{-9}]z_{out}^{-2} \\ H(z) =& h(2) + h(5)z_{out}^{-3} + h(8)z_{out}^{-6} + h(11)z_{out}^{-9} \\ &+ h(0)z_{out}^{-1} + h(3)z_{out}^{-4} + h(6)z_{out}^{-7} + h(9)z_{out}^{-10} \\ &+ h(1)z_{out}^{-2} + h(4)z_{out}^{-5} + h(7)z_{out}^{-8} + h(10)z_{out}^{-11} \\ H(z) =& \sum_{k=0}^{11} h(k)z_{out}^{-k} \end{aligned} \tag{2.5}$$

llegando a la forma de *descomposición polifásica*.

Se concluye finalmente diciendo que [17]:

- La cantidad de coeficientes usados para el filtro FIR puede ser tanto par como impar.
- Para compensar al pérdida de amplitud que se genera por la interpolación, se puede o bien incrementar el valor de los coeficientes del filtro por L o multiplicar el vector de salida $x_{int}(m)$ por L.
- Para facilitar la implementación, se recomienda que el filtro FIR utilizado sea de un número de coeficientes múltiplo entero del factor de interpolación L escogido.

2.2.2. Filtros CIC

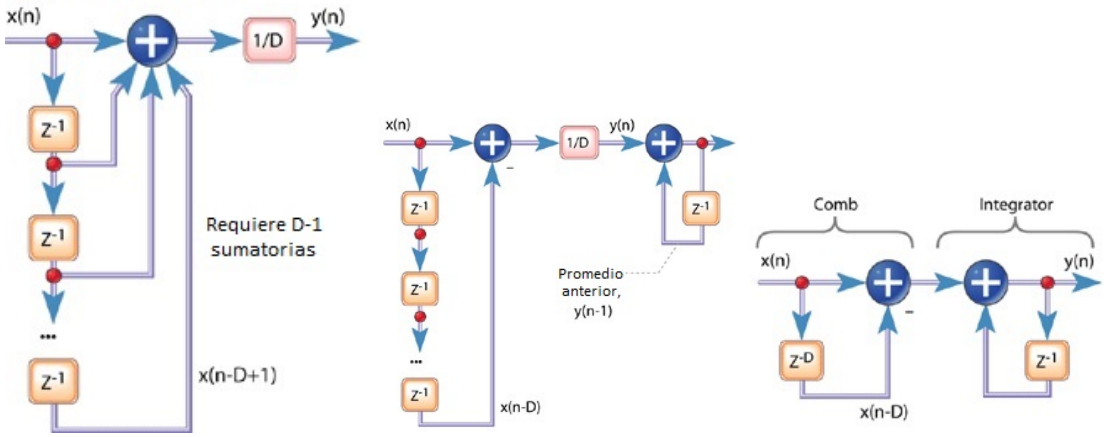
En la práctica, grandes cambios de tasa son logrados con múltiples etapas en cascada. Algunos de los beneficios de los filtros en cascada son:

- Una reducción global en la carga computacional.
- Diseños de filtros más sencillos.
- Reducido requerimiento de hardware en memoria.
- Disminución de los efectos negativos producidos por coeficientes finitos y acotados en longitud de palabra.

El filtro CIC está formado por diferenciadores digitales (combs) en cascada con acumuladores digitales (integradores) en cascada añadidos subsecuentemente en igual número. Entre los integradores y los diferenciadores se coloca un interruptor digital que aumenta o disminuye la frecuencia de muestreo del sistema por un factor R. Sea f_s, la frecuencia de muestreo del sistema, los integradores trabajan a esta frecuencia de muestreo y los diferenciadores trabajan a una frecuencia R veces más baja.

La magnitud de la respuesta en frecuencia del filtro CIC tiene una caída continua de la forma $\frac{sin(x)}{x}$. Como esto no es deseado, típicamente se agrega un filtro FIR compensador con respuesta en frecuencia inversa al filtro CIC, de la forma $\frac{x}{sin(x)}$, que permita obtener en su conjunto una respuesta plana en la banda de interés. Este filtro compensador siempre trabaja a la frecuencia más baja, $\frac{f_s}{R}$, por lo que se coloca antes del filtro CIC en operaciones de interpolación y después del filtro CIC en operaciones de decimación.

Los filtros CIC se originan de la noción de un filtro recursivo sumador móvil, que es en sí una forma eficiente de un filtro promediador.



(a) Filtro promediador móvil (b) Filtro recursivo sumador móvil (c) Filtro CIC

Figura 2.7: Filtros Promediadores

En la Figura 2.7a se ilustra el proceso de filtrado mediante promediado móvil. Se observa que son necesarias D-1 sumas, y una multiplicación por $\frac{1}{D}$, para obtener cada punto de la salida y(n).

La salida temporal del filtro promediador móvil es expresado como:

$$y(n) = \frac{1}{D}[x(n) + x(n-1) + x(n-2) + x(n-3) + ... + x(n-D+1)] \tag{2.6}$$

donde n es nuestro índice en el dominio del tiempo. Llevando la expresión al dominio z:

$$Y(z) = \frac{1}{D}[X(z) + X(z)z^{-1} + X(z)z^{-2} + X(z)z^{-3} + ... + X(z)z^{-D+1}] \tag{2.7}$$

y realizando algunas operaciones matemáticas, se llega a la función de transferencia:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{D}[1 + z^{-1} + z^{-2} + ... + z^{-D+1}] = \frac{1}{D}\sum_{n=0}^{D-1} z^{-n} \tag{2.8}$$

En la Figura 2.7b se presenta el filtro recursivo sumador móvil. En este proceso de filtrado, la entrada actual $x(n)$ es restada con la entrada más antigua $x(n-D)$, luego promediada por D y sumada a la salida anterior $y(n-1)$. Es llamada recursiva porque tiene un lazo de realimentación. Cada salida del filtro es guardada en una memoria y luego usada para calcular la próxima salida. La salida temporal del filtro recursivo sumador móvil se expresa:

$$y(n) = \frac{1}{D}[x(n) - x(n-D)] + y(n-1) \tag{2.9}$$

cuya función de transferencia está dada por:

$$H(z) = \frac{1}{D}\frac{1 - z^{-D}}{1 - z^{-1}} \tag{2.10}$$

La ecuación 2.8 es la expresión no recursiva y la ecuación 2.10 es la expresión recursiva de un promediador de punto D, siendo ambas equivalentes.

El filtro promediador móvil de la figura 2.7a debe realizar D-1 sumas por cada muestra de salida. Mientras que, el filtro recursivo sumador móvil tiene la ventaja de que sólo necesita una resta por muestra de salida, independientemente del retraso D. Esta eficiencia computacional hace al filtro recursivo sumador móvil atractivo en muchas aplicaciones en donde se busca la reducción del ruido mediante un promediado.

Simplificando la línea del retardo como sugiere la ecuación 2.10 e ignorando el escaleo de $\frac{1}{D}$, se obtiene la clásica forma de un filtro CIC de primer orden, como se observa en la Figura 2.7c. La primer sección del filtro CIC es llamada comb, cuyo retardo diferencial es D, y la sección de realimentación es llamada integrador. La etapa comb substrae una muestra D veces retrasada de la muestra actual y el integrador es simplemente un acumulador. La ecuación de salida del filtro CIC es:

$$y(n) = x(n) - x(n-D) + y(n-1) \tag{2.11}$$

y su función de transferencia es:

$$H_{CIC}(z) = \frac{1 - z^{-D}}{1 - z^{-1}} \tag{2.12}$$

Finalmente se concluye afirmando que el filtro CIC es equivalente a un filtro recursivo sumador móvil con la ventaja de que en su arquitectura sólo es necesario realizar operaciones de suma y resta.

Agregando un bloque de incremento de tasa por un factor R entre la etapa comb y la etapa integradora, como ilustra la Figura 2.8, el retardo diferencial respecto a la tasa alta toma como valor RM, donde R es el factor de interpolación y M es el retardo diferencial a la tasa más baja ($\frac{F_s}{R}$).

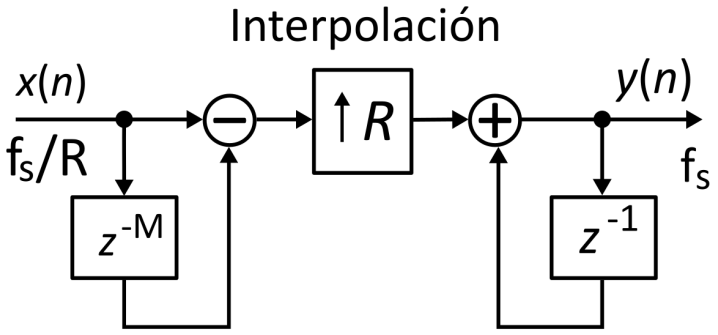


Figura 2.8: Filtro CIC Interpolador de 1 etapa

Teniendo en cuenta este cambio de tasa, la ecuación 2.12 queda expresada como:

$$H_{CIC}(z) = \frac{1 - z^{-RM}}{1 - z^{-1}} \tag{2.13}$$

En la práctica M se mantiene en los valores 1 o 2 y el rechazo a las imágenes es manejado mediante el número de etapas N en cascada del filtro CIC. En la siguiente sección se realiza un estudio más detallado de las implicancias de estos parámetros de diseño.

2.3. Filtro Interpolador CIC

Un filtro interpolador CIC es adecuado para un filtrado anti imágenes de señales interpoladas a un alto cambio de tasa donde el ancho de banda de la señal de interés es lo más estrecha posible.

2.3.1. Descripción

Definir un filtro de interpolación CIC se resume a definir los valores de la cantidad de etapas N, el retardo diferencial de la etapa comb M y el factor de cambio de tasa R, como ilustra la Figura 2.9.

La sección integradora de los filtros CIC consiste en N integradores en cascada trabajando a la tasa alta de muestreo (f_s). Cada integrador es implementado con un sólo polo con realimentación unitaria de la forma:

$$H_I(z) = \frac{1}{1 - z^{-1}} \tag{2.14}$$

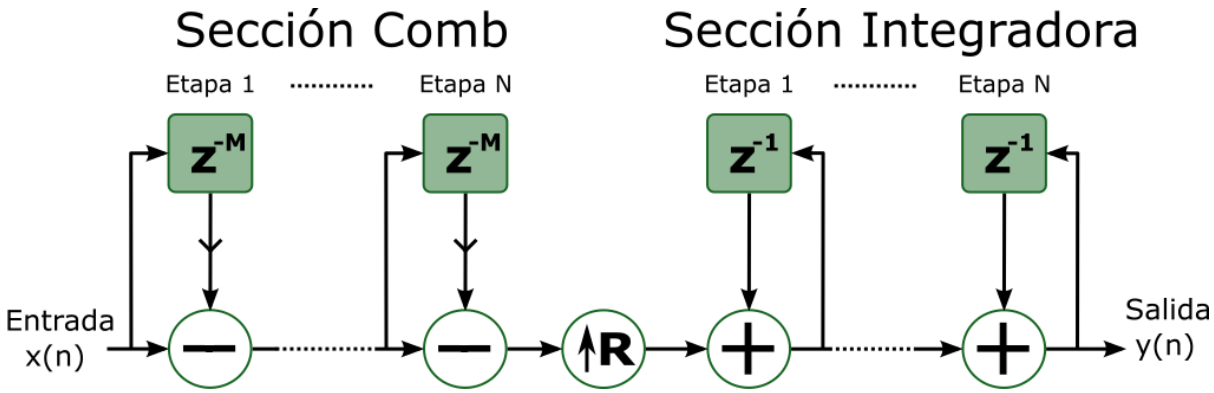


Figura 2.9: Filtro CIC Interpolador de N etapas

La sección comb consiste en N filtros comb en cascada operando a la tasa baja de muestreo ($\frac{f_s}{R}$). Cada etapa de filtro comb se ve afectada por el delay diferencial M, que llevado a la tasa alta f_s conforma la función de transferencia:

$$H_C(z) = 1 - z^{RM} \tag{2.15}$$

Multiplicando las ecuaciones 2.14 y 2.15 se obtiene la respuesta total de un filtro CIC de una etapa, resultando en la ecuación 2.13 como previamente se concluyó.

Si ahora se eleva el número de etapas a N, se obtiene la ecuación general del filtro CIC:

$$H(z) = H_I(z)^N H_C(z)^N = \frac{(1 - z^{RM})^N}{(1 - z^{-1})^N} \tag{2.16}$$

Entonces, el filtro CIC es implementable funcionalmente con N etapas de filtros FIR uniformes en cascada.

Operando con la ecuación 2.13 y reemplazando con $z = e^{\frac{j2\pi f}{R}}$:

$$\begin{aligned}
H_{CIC}(e^{\frac{j2\pi f}{R}}) =& \frac{1 - e^{\frac{-j2\pi fRM}{R}}}{1 - e^{\frac{-j2\pi f}{R}}} = \frac{e^{\frac{-j2\pi fRM}{2R}}(e^{\frac{j2\pi fRM}{2R}} - e^{\frac{-j2\pi fRM}{2R}})}{e^{\frac{-j2\pi f}{2R}}(e^{\frac{j2\pi f}{2R}} - e^{\frac{-j2\pi f}{2R}})} \\
H_{CIC}(e^{\frac{j2\pi f}{R}}) =& \frac{e^{-j\pi fM} 2j \sin(\frac{2\pi fRM}{2R})}{e^{\frac{-j\pi f}{R}} 2j \sin(\frac{2\pi f}{2R})} \\
H_{CIC}(e^{\frac{j2\pi f}{R}}) =& \frac{e^{-j\pi fM} \sin(\pi fM)}{e^{\frac{-j\pi f}{R}} \sin(\frac{\pi f}{R})} \\
H_{CIC}(e^{\frac{j2\pi f}{R}}) =& e^{-j\pi f(M-\frac{1}{R})} \frac{\sin(\pi fM)}{\sin(\frac{\pi f}{R})} 0
\end{aligned} \tag{2.17}$$

se obtiene que la ecuación de ganancia del filtro CIC resultante se puede aproximar a una respuesta del tipo $\frac{sin(x)}{x}$ de un filtro pasabajo centrado en $f = 0$ Hz. Para determinar el

comportamiento en $f = 0$ Hz se debe resolver la indeterminación:

$$|H_{CIC}(e^{\frac{j2\pi f}{R}})|_{f=0} = \left|\frac{\sin(0)}{\sin(0)}\right| = \frac{0}{0} \tag{2.18}$$

Aplicando l'Hôpital:

$$|H_{CIC}(e^{\frac{j2\pi f}{R}})|_{f=0} = \frac{\cos(\pi M f)\pi M}{\cos(\frac{\pi f}{R})\frac{\pi}{R}} = \frac{\cos(0)\pi M}{\cos(0)\frac{\pi}{R}} = MR \tag{2.19}$$

Entonces, la ganancia del filtro CIC en continua es igual al retardo diferencial del filtro comb a la tasa alta MR.

Examinando el comportamiento de un filtro CIC N=1, M=2 y R=2 en el dominio del tiempo discreto ante una entrada impulsiva, como se ilustra en la Figura 2.10, se procede a analizar su comportamiento. El impulso ingresa al filtro comb, que reproduce el mismo impulso apenas ingresa y luego de M muestras lo invierte. A esta secuencia se la interpola por el factor R, llegando a la tasa alta f_s. Una vez que ya contamos con la señal a la tasa alta, se procede a integrarla. Al llegar el impulso positivo al integrador, éste se eleva al valor del impulso, manteniendo éste valor hasta la llegada del impulso invertido, que se encarga de llevar a cero la salida del integrador. Quedando a la salida del integrador, una señal cuadrada de ancho MR muestras.

La respuesta en frecuencia normalizada del filtro examinado se puede observar en la Figura 2.11a, su respuesta en fase en la Figura 2.11b y su respectivo diagrama de polos y ceros en la Figura 2.11c.

Si bien se tiene un polo del filtro directamente en el círculo unitario, no hay error de cuantificación de coeficientes en la función de transferencia 2.16. Los coeficientes del filtro CIC son todos uno y pueden ser representados con precisión perfecta en formato de número de punto fijo. Aunque sea un filtro recursivo, los filtros CIC son garantizados estables, de fase lineal y tienen respuesta al impulso finita.

En el trabajo realizado por Hogenauer [3], se define a f_c como:

$$f_c = \frac{BW_{filter}}{\frac{f_s}{R}} \tag{2.20}$$

donde la frecuencia de corte del filtro f_c queda experesada relativa a la frecuencia de muestreo baja $\frac{f_s}{R}$.

Ahora bien, no hay que confundir a f_c con la frecuencia de corte de un filtro a -3 dB sino que es simplemente una frecuencia para la cual Hogenauer nos presenta mediciones de atenuación en función de los parámetros de diseño a través de dos tablas de referencia.

La tabla 2.1 presenta valores de atenuación en la banda de paso del filtro CIC dado un producto de retardo diferencial (M) y ancho de banda relativo (f_c).

La tabla 2.2 presenta valores de atenuación de las imágenes del filtro CIC ubicadas en f_{ia} dado un retardo diferencial (M), un ancho de banda (f_c) y el número de

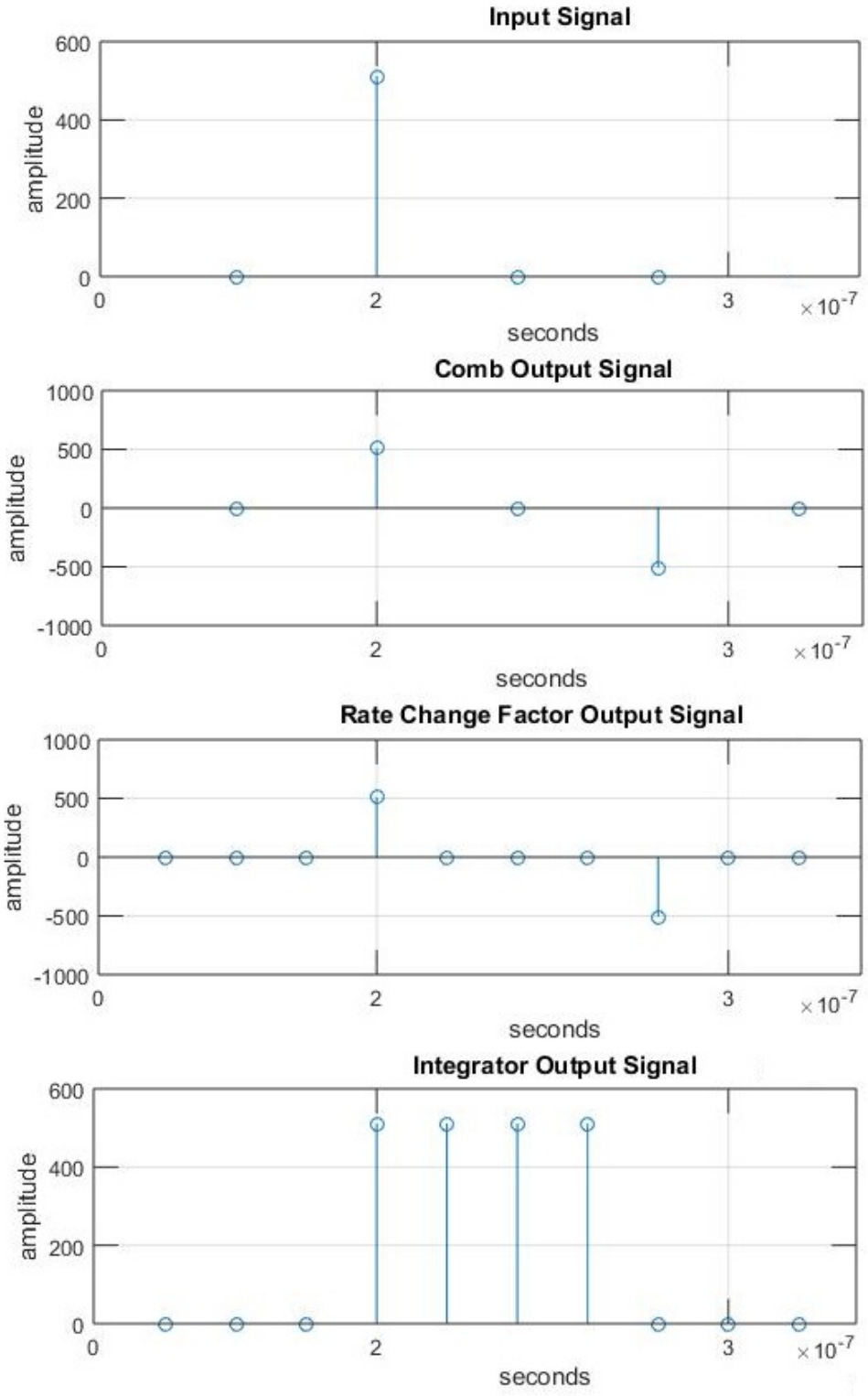
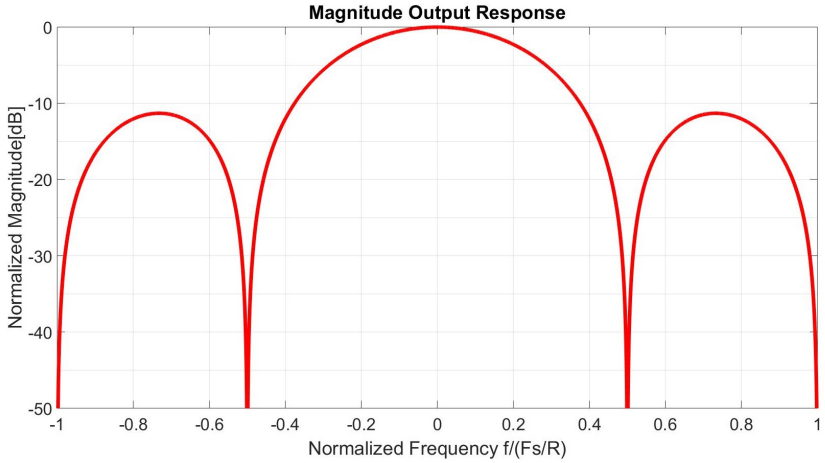
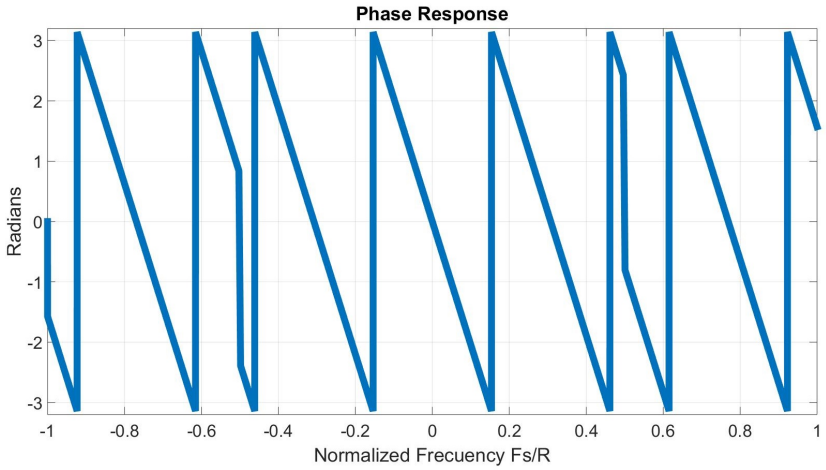


Figura 2.10: Respuesta en el dominio del tiempo de un filtro CIC N=1, M=2, R=2

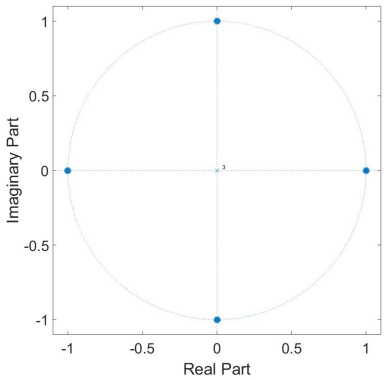
etapas N. Se considera como error de imagen al máximo error sobre todas las bandas de imágenes y estará caracterizado por la mínima atenuación presente en el primer lóbulo



(a) Respuesta de magnitud en frecuencia normalizada a $\frac{f_s}{R}$ de un filtro CIC N=1, M=2, R=2



(b) Respuesta de fase en frecuencia normalizada a $\frac{f_s}{R}$ de un filtro CIC con N=1, M=2 y R=2



(c) Diagrama de ceros y polos de un filtro CIC con N=1, M=2 y R=2

Figura 2.11: Respuestas de un filtro CIC N=1, M=2 y R=2

lateral, en $f_{ia} = 1 - f_c$.

Ancho de banda relativo al producto del delay diferencial M*fc	**Atenuación en la banda de paso en fc en función del número de etapas N [dB]**					
	1	**2**	**3**	**4**	**5**	**6**
0,00781250	0.00	0.00	0.00	0.00	0.00	0.01
0,01562500	0.00	0.01	0.01	0.01	0.02	0.02
0,03125000	0.01	0.03	0.04	0.06	0.07	0.08
0,06250000	0.06	0.11	0.17	0.22	0.28	0.34
0,12500000	0.22	0.45	0.67	0.90	1.12	1.35
0,25000000	0.91	1.82	2.74	3.65	4.56	5.47

Tabla 2.1: Atenuación en la banda de paso para grandes factores de cambio de tasa

Delay Diferencial M	**Ancho de banda relativo f_c**	**Atenuación de las imágenes en f_{ia} [dB] en función del número de etapas N**					
		1	**2**	**3**	**4**	**5**	**6**
1	0,0078125	42.1	84.2	126.2	168.3	210.4	252.5
1	0,0156250	36.0	72.0	108.0	144.0	180.0	215.9
1	0,0312500	29.8	59.7	89.5	119.4	149.2	179.0
1	0,0625000	23.6	47.2	70.7	94.3	117.9	141.5
1	0,1250000	17.1	34.3	51.4	68.5	85.6	102.8
1	0,2500000	10.5	20.9	31.4	41.8	52.3	62.7
2	0,00390625	48.1	96.3	144.4	192.5	240.7	288.8
2	0,00781250	42.1	84.2	126.2	168.3	210.4	252.5
2	0,01562500	36.0	72.0	108.0	144.0	180.0	216.0
2	0,03125000	29.9	59.8	89.6	119.5	149.4	179.3
2	0,06250000	23.7	47.5	71.2	95.0	118.7	142.5
2	0,12500000	17.8	35.6	53.4	71.3	89.1	106.9

Tabla 2.2: Atenuación de las imágenes para grandes factores de cambio de tasa

Siguiendo el caso de uso planteado anteriormente N=1, M=2 y R=2, en la Figura 2.11a puede verse la aparición de los nulos en múltiplos de $f = \frac{1}{M}$ en frecuencia normalizada. Éstos nulos corresponden a los ceros de la función de transferencia 2.15 del filtro comb, que se encargan de ubicar MR ceros igualmente espaciados alrededor del círculo unitario donde se obtiene una magnitud de salida nula en cada uno de esos puntos.

Cambiando el caso de uso por los parámetros R=8, M=1 y N=1, se obtiene la gráfica de polos y ceros de la Figura 2.12.

Manteniendo N=1 y R=8, pero modificando M=2 se obtiene la gráfica de polos y ceros de la Figura 2.13, donde lógicamente se observa el doble de ceros alrededor del círculo unitario.

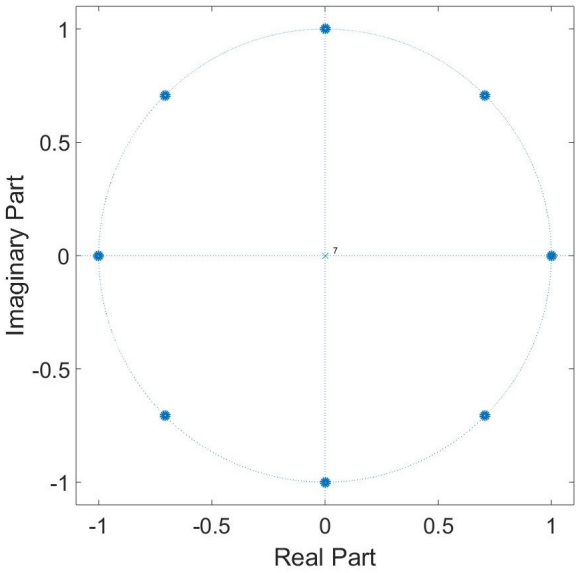


Figura 2.12: Diagrama de ceros y polos de un filtro CIC N=1, M=1, R=8

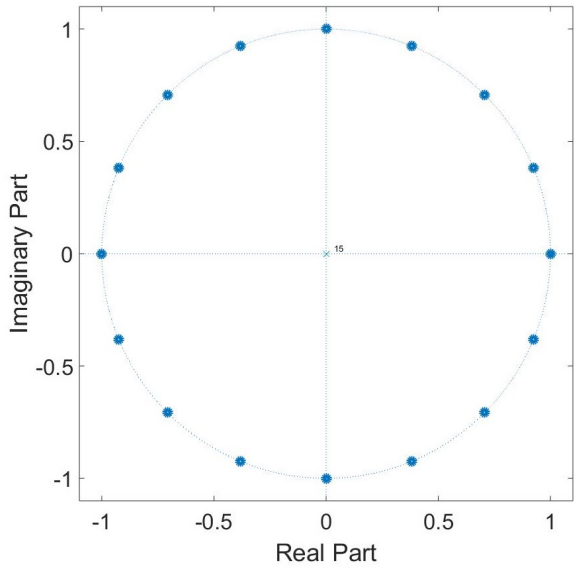


Figura 2.13: Diagrama de ceros y polos de un filtro CIC N=1, M=2, R=8

Observando la salida del filtro CIC, en la Figura 2.14 se puede ver claramente la mejora en atenuación tomando como referencia el máximo del lóbulo de la primer imagen. Esto significa que si tomo un M=2 obtengo una mayor atenuación en el máximo del primer lóbulo lateral resultando en una atenuación mayor de la primer imagen y subsiguientes, pero una disminución de la mitad en lo que respecta al ancho de banda de la banda de paso del filtro con respecto a un M=1.

Las imágenes se hacen presentes en múltiplos enteros de $\frac{f_s}{R}$ en frecuencia normalizada. Tomando R=8, N=1 y M=2, con $f_c = \frac{1}{8}$ y $f_{ia} = \frac{7}{8}$ el error de imagen para M=1 es de $-16{,}95$ dB. Mientras que, para M=2 es de $-17{,}64$ dB.

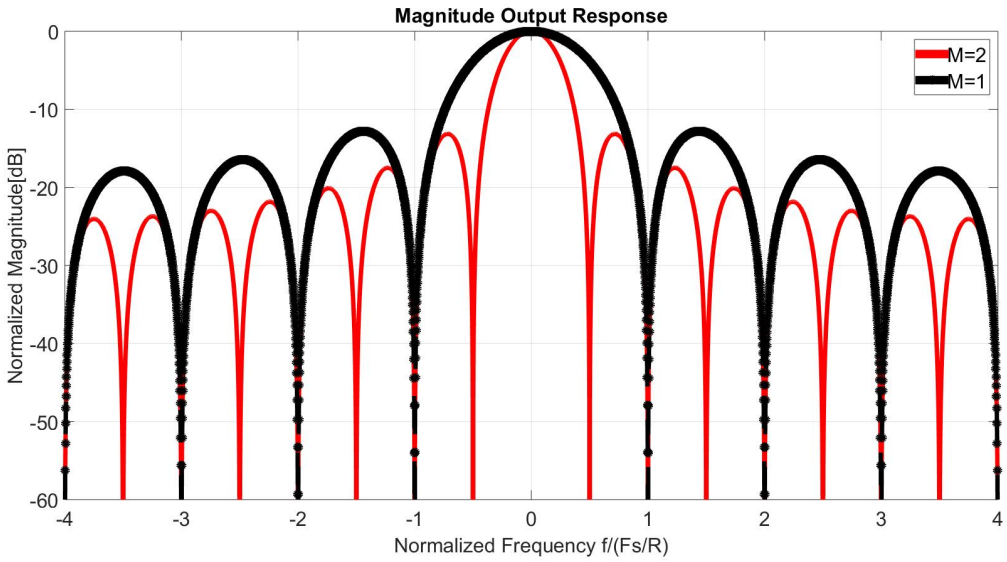


Figura 2.14: Comparación de la salida de un filtro CIC N=1, R=8 para valores de M=1 y M=2

El crecimiento máximo de registro es definido como la máxima salida de magnitud posible dado el peor caso de señal de entrada relativa al máximo valor de magnitud de entrada. Este crecimiento es usado en el proceso de diseño de los filtros CIC para que no se pierda ningún dato debido a un overflow de los registros y es del valor:

$$G_{max} = (RM)^N \tag{2.21}$$

Por ello siempre a nivel de implementación se prefiere trabajar con un retardo diferencial M=1 de ser posible. Eso permitiría reducir a la mitad la resolución necesaria en cada etapa, en comparación con un delay diferencial M=2.

Para mejorar la atenuación de rechazo de las imágenes en el filtro CIC se incrementa el número de etapas. Ésto influye directamente en la ganancia del filtro, que aumenta de forma exponencial como puede reflejarse en:

$$P(f) = \left[\frac{\sin(\pi M f)}{\sin(\frac{\pi f}{R})}\right]^{2N} \tag{2.22}$$

Al aumentar las etapas también se afecta la caída en la banda de paso e incrementa la cantidad de requerimientos de hardware al agregar más sumadores. Una penalidad adicional del filtro es el ancho de la palabra necesario para cada sumador. Ya que los filtros para mantenerse estables deben trabajar con una precisión completa, el número de bits necesario en cada sumador es: $Nlog_2(RM)$ conllevando a un aumento significativo en el ancho de la palabra para filtros de orden superior.

En la Figura 2.15 se puede apreciar la salida del filtro CIC R=8 y M=2 en frecuencia normalizada para N=1 y N=4. Nótese que con el aumento de N, el ancho de

banda efectivo de la banda de paso del filtro CIC disminuye y se atenúa cada vez más pronunciadamente. Sin un correcto compensador de la señal de entrada, sería imposible utilizar el filtro CIC por sí sólo debido a la caída continua presente en la banda de paso del filtro CIC.

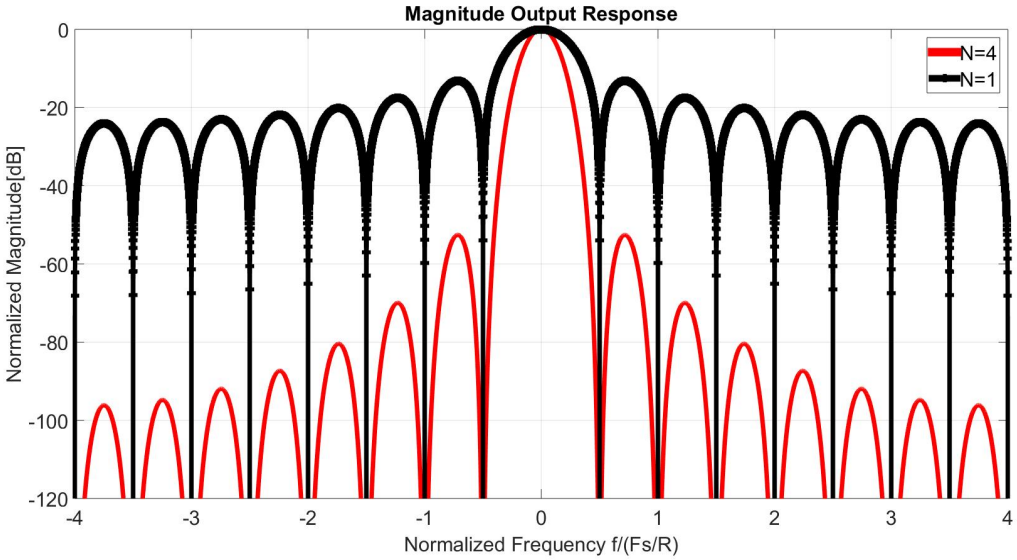


Figura 2.15: Comparación de las salidas de un filtro CIC R=8, M=2 para N=1 y N=4

El último parámetro de diseño del filtro CIC por analizar es el factor de cambio de tasa R. Es éste parámetro el que define a su vez la cantidad de imágenes luego de la interpolación. Estas imágenes se hacen presentes a la salida del filtro CIC en forma de lóbulos adyacentes al lóbulo centrado en 0 Hz.

Al diseñar un filtro CIC, siempre conviene trabajar con una f_c lo menor posible, al ser inversamente proporcional al factor de cambio de tasa R. Esto permite adoptar el máximo R posible. Y es allí donde se hacen presentes las ventajas de los filtros CIC.

Si bien la ganancia de un filtro CIC es como se muestra en la ecuación 2.21, los integradores pueden experimentar desbordamiento de forma individual debido a la indeterminación a 0 Hz. Ante este escenario, el uso de una artirmética de complemento a dos resuelve esta situación siempre y cuando el ancho de la palabra de cada integrador acomode la máxima diferencia entre 2 muestras sucesivas.

Los filtros interpoladores, al insertar ceros entre muestra y muestra, se reduce la ganancia de los filtros CIC en $\frac{1}{R}$ como se explicó anteriormente en la sección 2.1. Por ello la ganancia neta de un filtro de interpolación CIC es:

$$G_{CIC_{Interpolador}} = \frac{(MR)^N}{R} \tag{2.23}$$

2.3.2. Aplicaciones

A modo de ejemplo comparativo se tienen los requerimientos del filtro interpolador en la Tabla 2.3 y se realiza una comparación de utilización de recursos al implementar el filtro interpolador mediante la arquitectura CIC y mediante la arquitectura polifásica. Para poder comparar ambas arquitecturas en igualdad de condiciones, se agrega una etapa compensadora al filtro CIC, que actúe mejorando, según los requerimientos, la banda de paso deseada del filtro.

Parámetro	Valor
Ripple en la banda de paso	$< 0{,}2$dB
Frecuencia de paso	5,712MHz
Frecuencia de rechazo	11MHz
Atenuación de rechazo	60dB
Frecuencia de muestreo alta (f_s)	913,92MHz

Tabla 2.3: Requerimientos para el filtro interpolador

Por ello se compara la respuesta del filtro interpolador CIC en conjunto con su filtro compensador y la respuesta del filtro interpolador polifásico. Para cumplir con los requerimientos planteados se diseña un filtro interpolador CIC de N=3,M=1 y R=10 con su filtro compensador de 48 coeficientes y un filtro interpolador polifásico de 430 coeficientes.

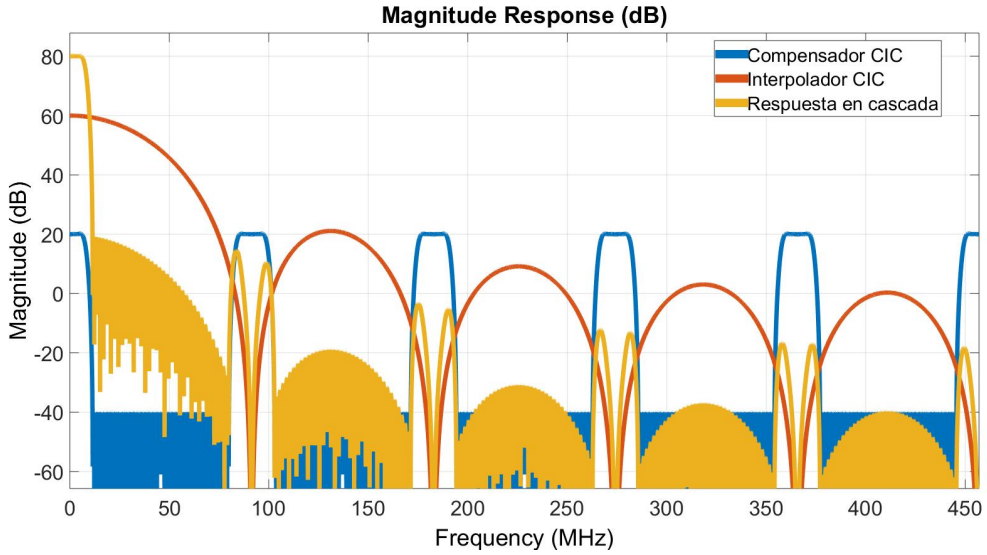


Figura 2.16: Respuesta del filtro Interpolador CIC compensado

La Figura 2.16 muestra la respuesta en frecuencia del filtro interpolador CIC, su filtro compensador y la respuesta en cascada de ambos filtros implementados. Los resultados de la respuesta en cascada de ambos filtros se especifican en la Tabla 2.4. Todas las multiplicaciones y 68 de las sumas necesarias por muestra de entrada son realizadas a la tasa baja de muestreo ($\frac{f_s}{R}$) y tan sólo 3 sumas son realizadas a la tasa alta de muestreo (f_s).

CIC compensado	
Ripple en la banda de paso	0,17dB
Frecuencia de paso	5,712MHz
Frecuencia de rechazo	11,424MHz
Atenuación de rechazo	60,07dB
Multiplicaciones por muestra de entrada	48
Sumas por muestra de entrada	71

Tabla 2.4: Resultados de implementación del Filtro Interpolador CIC N=3,M=1 y R=10 compensado con un filtro FIR de 48 coeficientes

En contraposición, en la Tabla 2.5 se especifican los resultados de la implementación de un filtro polifásico R=10 de 430 coeficientes. Se puede apreciar que mejora el rechazo a las imágenes notablemente pero todas las operaciones son realizadas a la tasa alta de muestreo (f_s).

Polifásico	
Ripple en la banda de paso	0,1976dB
Frecuencia de paso	5,857MHz
Frecuencia de rechazo	11,43MHz
Atenuación de rechazo	61,19dB
Multiplicaciones por muestra de entrada	430
Sumas por muestra de entrada	420

Tabla 2.5: Resultados de implementación del Filtro Interpolador Polifásico R=10 de 430 coeficientes

Finalmente, la Figura 2.17 se tiene la respuesta de ambas arquitecturas superpuestas.

A nivel energético es mucho más demandante mantener un circuito trabajando a una frecuencia alta que a una frecuencia baja. Se necesita más corriente para mantener alimentados los circuitos y que respondan rápidamente. Por ello a nivel de implementación en punto fijo en una FPGA el filtro interpolador CIC es óptimo. Tan sólo las operaciones de suma necesarias en la etapa de integración son las que trabajan a una tasa alta.

Si aumentamos el factor de interpolación a un valor 10 veces mayor y disminuimos el ancho de banda de paso del filtro en la misma medida se acentúan más las ventajas que tiene el filtro de interpolación CIC con respecto al filtro polifásico.

Éste último requerirá aún más coeficientes en contraposición al filtro interpolador CIC que con la misma cantidad de etapas puede cumplir con los requerimientos planteados. Incluso al disminuir en ancho de banda de la banda de paso, las imágenes se acercan aún más a los ceros de la función de transferencia, atenúandose aún más.

El filtro interpolador CIC es óptimo para aplicaciones que requieren grandes cambios de tasa y la banda de paso de interés del filtro es acotada a unos cuantos Khz.

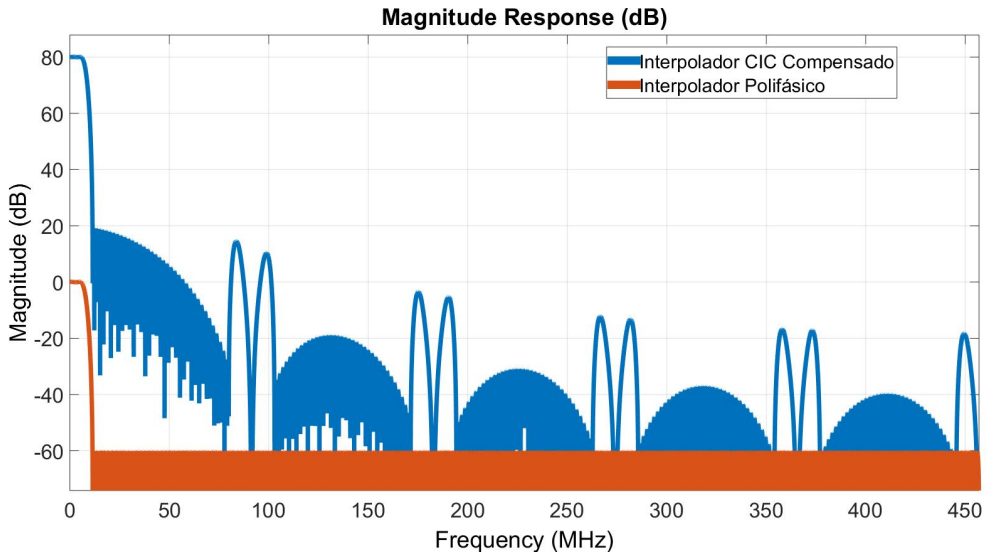


Figura 2.17: Respuesta de ambas arquitecturas de filtro interpolador

La aplicación de los filtros CIC es en áreas donde se requiere grandes frecuencias de muestreo, lo que vuelve a la implementación de multiplicadores en una opción antieconómica, y en áreas donde grandes factores de cambio de tasa requerirían demasiado almacenamiento en la cantidad de coeficientes para almacenar o generación rápida de respuesta impulsiva.

Adicionalmente, los filtros CIC pueden ser típicamente reconfigurados para diferentes cambios de tasa cambiando solamente la sección de interpolación asumiendo que el ancho de la palabra de las secciones comb e integradora se calculan para el máximo cambio de tasa posible.

Capítulo 3

Diseño en punto flotante

Para realizar el diseño en punto flotante se hace uso de scripts de MATLAB. Los mismos se encargan adicionalmente de automatizar la simualción en punto fijo con los parámetros configurados desde punto flotante. Principalemente cuenta con un archivo *main* y un único archivo de configuración llamado *config* donde se setea la entrada, la resolución, los parámetros del filtro CIC y las variables para graficar los resultados.

En el presente capítulo se presentan los algoritmos implementados en MATLAB y se verifican los mismos por simulación para entrada impulsiva, senoidal y de ruido uniforme. En el apéndice A se encuentra el código escrito de lo que en este capítulo se explicará en detalle.

3.1. Algoritmos de implementación

Los algoritmos de implementación de cada sección fueron deducidos desde la función de transferencia correspondiente y del razonamiento teórico presentado en el capítulo anterior.

3.1.1. Filtro comb

El sistema visto desde la entrada, trabaja a la frecuencia baja de muestro ($\frac{f_s}{R}$). Por lo que la ecuación 2.15 a la entrada del sistema tiene la función de transferencia:

$$H_C(z) = 1 - z^{-M} \tag{3.1}$$

Reemplazando $H_C(z) = \frac{Y_C(z)}{X_C(z)}$:

$$\begin{aligned}\frac{Y_C(z)}{X_C(z)} &= 1 - z^{-M} \\ Y_C(z) &= X_C(z) - z^{-M} X_C(z)\end{aligned} \tag{3.2}$$

Ahora llevado al dominio del tiempo:

$$Y_C(n) = X_C(n) - X_C(n - M) \tag{3.3}$$

La salida actual del filtro comb es en realidad la entrada actual menos la entrada M muestras anterior.

Los primeros M valores del vector de salida son iguales a la entrada, ya que hasta llegar al índice M+1, la historia del buffer es cero.

El filtrado se realiza mediante una función que tiene como parámetro de entrada un vector x. Se cargan los primeros M valores del vector de entrada y luego se recorre desde M+1 hasta la longitud total del vector x, realizando en cada iteración la resta de las muestras pertinententes, y así conformando punto a punto cada valor de su vector de salida *output vector*. El algoritmo implementado se encuentra en el Anexo A.6.6.

3.1.2. Cambio de tasa

Interpolar significa insertar R-1 ceros entre muestra y muestra. Para realizar un cambio de tasa, dado un vector de entrada x con longitud L, se obtiene un vector de salida de longitud RL. En este vector de salida, cada iR muestras, siendo $i = 1, 2, 3, .., L$, la muestra en el vector de salida se corresponde con la muestra del vector de entrada suministrado con el índice i.

El algoritmo se implementa a través de una función llamada *rate change factor*. En una primera instancia se guardan RL espacios de memoria con ceros. Luego, se recorre el vector de entrada x a través de un ciclo for y dentro de cada iteración se le asigna al vector de salida, *output vector*, en los valores de índice iR, el valor del vector de entrada correspondiente.

El algoritmo implementado se encuentra en el Anexo A.6.8

3.1.3. Filtro integrador

Se toma la ecuación 2.14 y se resuelve:

$$\begin{aligned} \frac{Y_I(z)}{X_I(z)} &= \frac{1}{1-z^{-1}} \\ (1-z^{-1})Y_I(z) &= X_I(z) \\ Y_I(z) - Y_I(z)z^{-1} &= X_I(z) \\ Y_I(z) &= X_I(z) + Y_I(z)z^{-1} \end{aligned} \tag{3.4}$$

Llevada a una expresión temporal:

$$Y_I(n) = X_I(n) + Y_I(n-1) \tag{3.5}$$

La salida actual del filtro integrador es igual a la entrada actual sumada a la entrada anterior. Por ello también se lo conoce como acumulador.

El algoritmo implementado se encuentra en el Anexo A.6.7. Consiste en una función con un vector de entrada x y un vector de salida llamado *output vector*. Primero se asigna la memoria del vector de salida. Luego, para que el primer valor del vector de entrada no se pierda, se lo asigna como primer valor del vector de salida. Después, se procede a recorrer el vector de entrada a través de un ciclo for, calculando punto a punto el valor del vector de salida con la ecuación 3.5.

3.1.4. Multietapa

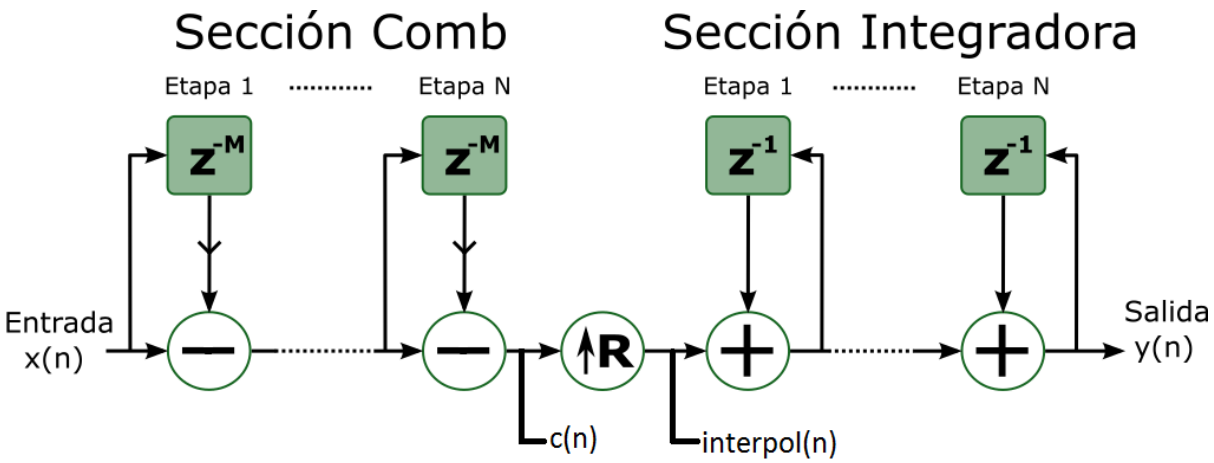


Figura 3.1: Filtro CIC Interpolador de N etapas

Los algoritmos anteriores se encuentran integrados en un único script de simulación en punto flotante llamado *floating point* y se encuentra en el Anexo A.6.9.El script consiste en una función que tiene sólo un parámetro de entrada, el vector de entrada de la señal llamado x. Como salida devuelve 3 vectores:

- c: Vector de salida de la sección comb del filtro CIC

- *interpol*: Vector de salida luego que se realiza la interpolación por el factor de cambio de tasa R
- y: Vector de salida de la sección integradora del filtro CIC

Como primera instancia dentro de la función se cargan los parámetros de configuración del filtro CIC (configurados en el archivo *config*). Seguidamente, se comienza con la sección comb del filtro CIC. En ella primero se asigna el valor del vector de entrada a c, para no perder el vector de entrada, y a través de un ciclo for se llama la función *comb filter* tantas veces como etapas del filtro CIC se hayan configurado (el número de etapas N en el diseño de punto flotante es llamado *stages*).

Posteriormente, se le asigna a *interpol* la salida de la función *rate change factor* que se encarga de realizar el cambio de tasa.

Finalmente, la sección integradora algoritmicamente es igual a la sección comb del filtro CIC. Se asigna primero el valor del vector *interpol* a y y luego se procede a realizar el filtrado de la señal, a través de la función *integrator filter*, tantas veces como etapas del filtro se hayan configurado.

La Figura 3.1 ilustra en qué punto del filtro de interpolación CIC se toma la salida para cada vector.

3.2. Verificación por simulación

Para corroborar el correcto funcionamiento del algoritmo implementado en punto flotante se procede a verificar el diseño mediante simulación para entrada impulsiva, senoidal y de ruido uniforme.

En el script *config*, ubicado en el Anexo A.2, se configuran las siguientes variables:

- R: Entero mayor a 2 que especifica el factor de interpolación del filtro CIC
- M: Entero de valor 1 o 2 que especifica el retardo diferencial a la tasa baja $\frac{F_s}{R}$ de la sección comb del filtro de interpolación CIC
- *stages*: Entero mayor o igual a 1 y menor o igual 6 que especifica la cantidad de etapas N del filtro de interpolación CIC
- F_s: Entero positivo que especifica la frecuencia en Hz de la tasa alta del filtro de interpolación CIC
- *Signal Type*: String que especifica el tipo de señal a generar
- *Noise Add*: Boolean que determina si se agrega ruido o no a la señal generada
- *Noise Source*: Booelan que especifica la fuente de la generación de ruido. Puede ser tanto de punto flotante(0) como de punto fijo(1).

- *Sin Source*: Boolean que especifica la fuente de la generación de onda senoidal. Puede ser tanto de punto flotante(0) como de punto fijo(1).
- *Total Samples*: Entero positivo que determina la cantidad de muestras del vector de entrada
- *Impulse Amplitude*: Entero positivo que determina la amplitud del impulso a generar en punto flotante y fijo
- *A*: Flotante positivo que determina la amplitud de la onda senoidal a generar en punto flotante y fijo
- *f*: Entero positivo que determina la frecuencia en Hz de la onda senoidal a generar en punto flotante. Siempre que se elija también realizar la simulación en punto fijo esta variable se modifica por cuestiones de implementación
- *phi*: Flotante que determina el desfasaje en radianes de la onda senoidal a generar
- *Offset*: Flotante que determina el offset de continua de la onda senoidal a generar
- *Noise Var*: Flotante positivo que determina la varianza del ruido uniforme a generar en punto flotante
- *Noise mean*: Flotante que determina la media del ruido uniforme a generar en punto flotante
- *Sim Type*: String que puede tomar como valor "floating", "fixed" o "floatingfixed" dependiendo si se pretende que se realice la simulación en punto flotante, fijo o ambas respectivamente
- *Input Bits*: Entero positivo que especifica la cantidad de bits a la entrada del filtro de interpolación CIC en punto fijo
- *Fractional Bits*: Entero positivo que especifica la cantidad de bits de los bits de entrada a tomar para representar la parte fraccional de la señal de entrada en punto fijo
- *LUT address width*: Entero positivo que especifica el ancho de la palabra de la LUT a utilizar en punto fijo para la generación de onda senoidal en punto fijo
- *Output Bits*: Entero positivo que especifica la cantidad de bits a tomar a la salida del filtro de interpolación CIC en punto fijo. Este valor debe ser igual o menor a la resolución efectiva del filtro a la salida, tomando en cuenta el crecimiento del ancho de la palabra a lo largo del filtro
- *Output Frac bits*: Entero positivo que especifica la cantidad de bits de los bits de salida a tomar para representar la parte fraccional de la señal de salida del filtro de interpolación CIC en punto fijo
- *Plot All Signals*: Boolean que setea si se realiza la gráfica de las señales de punto flotante *x*, *c*, *interpol* y *y* en el tiempo
- *Plot Phase*: Boolean que setea si se realiza la gráfica de la respuesta en fase de la señal de salida del filtro de interpolación CIC en punto flotante

- *Plot tf*: Boolean que setea si se realiza la gráfica de zeros y polos de la función de transferencia del filtro de interpolación CIC
- *Plot I O*: Boolean que setea si se realiza la gráfica de la respuesta en frecuencia de la entrada y la salida del filtro de interpolación CIC
- *Plot Error*: Boolean que setea si se realiza la gráfica de la respuesta en frecuencia de la respuesta impulsiva del filtro de interpolación CIC, y la salida efectiva de la/s simulación/es que se han configurado previamente

La entrada se genera mediante el script *input config*, mostrado en el Anexo A.6.2. Es una función que toma los parámetros de las señales de entrada configurados en el archivo *config* y de acuerdo al parámetro seteado en *Signal Type*, genera la correspondientes muestras del vector de entrada x a tomar como entrada para la simulación en punto flotante.

A continuación se realiza el recorrido de cada señal generada desde la entrada a la sección comb hasta la salida de la sección integradora. Se muestran tanto las gráficas temporales para cada tipo de señal generada como asi tambien la respuesta en frecuencia.

3.2.1. Entrada impulsiva

Para la generación de la entrada impulsiva se utiliza el script del Anexo A.6.3. El mismo consiste en una función que tiene como parámetros de entrada la amplitud del impulso, la cantidad total de muestras a generar y el delay diferencial M configurados en el archivo *config*. Tiene como salida un vector de ceros x de la longitud configurada, y en la posición $M + 1$ se encuentra el impulso con la ampltiud configurada.

En el dominio temporal, al ingresar el impulso a la sección comb, el filtro comb se encarga de reproducir el mismo impulso y luego de M muestras invertirlo. Al hacer el cambio de tasa, se interpola la salida del filtro comb con R-1 ceros intermedios. El integrador se comporta como un acumulador. El primer impulso proveniente del filtro comb eleva al valor del impulso, la entrada del integrador y, al llegar el impulso invertido, se cancela, llevando a cero la salida. Finalmente, a la salida del filtro se conforma una onda rectangular de ancho RM. Sea R=4, M=2, N=1 los parámetros del filtro de interpolación CIC, se ilustra en la Figura 3.2 el comportamiento a través de las etapas descriptas anteriormente.

En la Figura 3.3 se ilustra la salida en frecuencia normalizada del caso de uso R=4, M=2, N=1 y la curva teórica de la salida del filtro para ese caso de uso. Se observa que ambas curvas se superponen, concluyendo en que la respuesta impulsiva simulada en punto flotante concuerda con la salida teórica del filtro.

3.2.2. Entrada de onda senoidal

Para la generación de la entrada senoidal se utiliza el script del Anexo A.6.4. El mismo consiste en una función que tiene como parámetros de entrada el periodo de

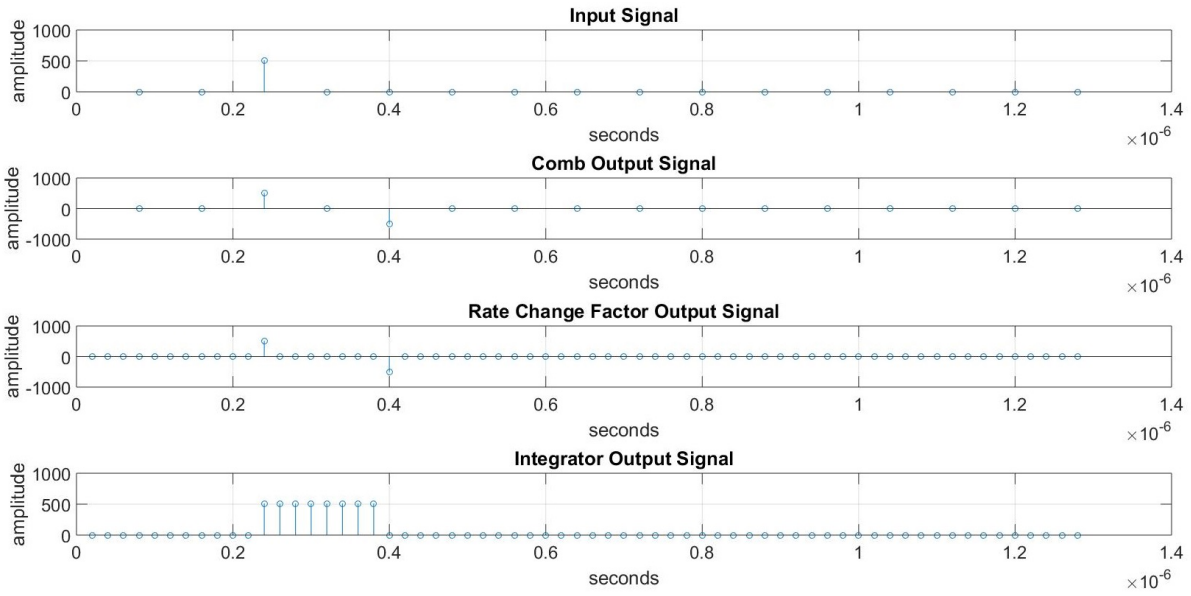


Figura 3.2: Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada impulsiva de amplitud 511, simulada en punto flotante

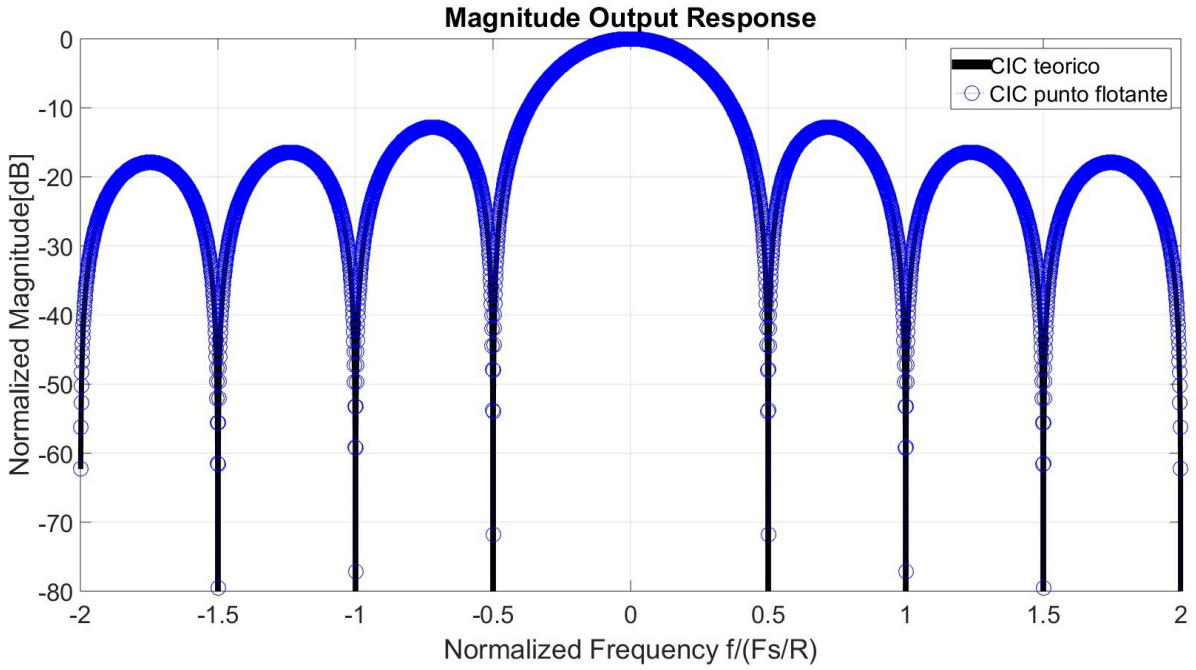


Figura 3.3: Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada impulsiva de amplitud 511, simulada en punto flotante

muestreo de la tasa alta, la amplitud de la onda senoidal, la frecuencia de la onda senoidal, la cantidad de muestras a generar, el desfasaje y el offset de continua de la señal senoidal. Todo estos parámetros se configuran en el archivo *config*. Tiene como salida un vector x de la longitud configurada, construido con la función *sin* de MATLAB de acuerdo al vector de tiempo previamente generado.

Siguiendo con el mismo caso de uso planteado que para entrada impulsiva, se genera una onda senoidal de f=781,25 Khz y A=0.5. La Figura 3.4 muestra el recorrido de la señal generada a lo largo de las secciones del filtro. Se verifica que la salida sea de amplitud 1, de acuerdo a la ecuación 2.23.

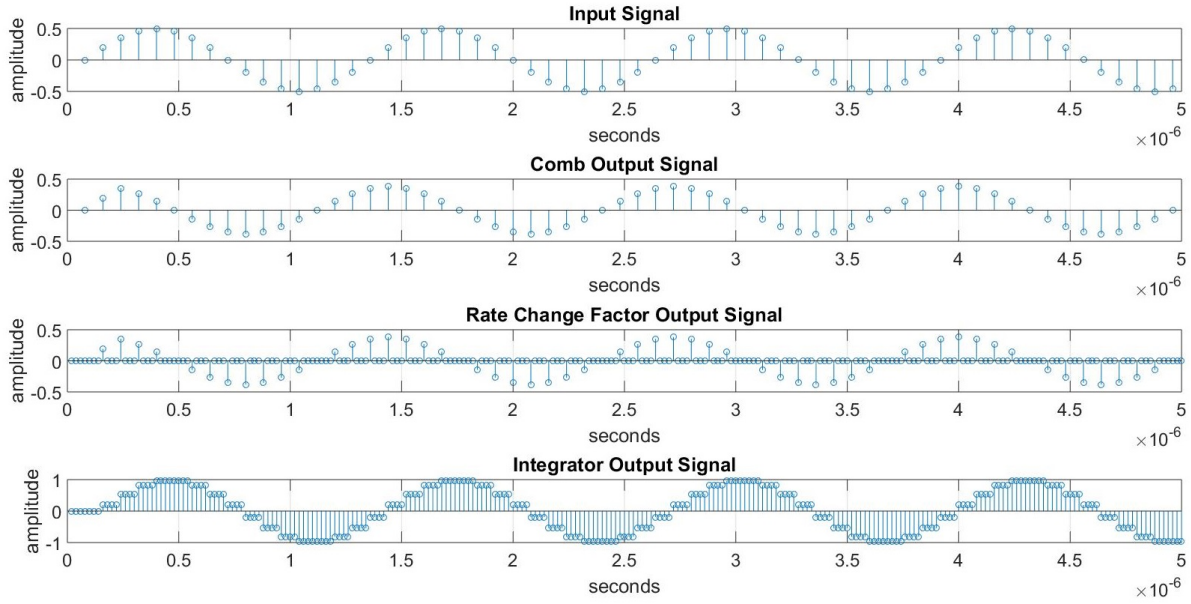


Figura 3.4: Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada senoidal de f=781.25Khz, A=0.5, phi=0,offset=0, simulada en punto flotante

En el dominio de la frecuencia se puede observar en la Figura 3.5 los tonos de la señal generada en el punto +/- 0.0625 de frecuencia normalizada y la aparición de las imagenes centradas a múltiplos de $\frac{f_s}{R}$. Estas imágenes se encuentran en los puntos:

- $1 - 0{,}0625 = +/- 0{,}9375$
- $1 + 0{,}0625 = +/- 1{,}0625$
- $2 - 0{,}0625 = +/- 1{,}9375$

Se observa además una diferencia de apróximadamente 20 dB entre el tono central de la onda senoidal y la primer imagen. Ésta atenuación se puede mejorar con el incremento de etapas del filtro interpolador CIC.

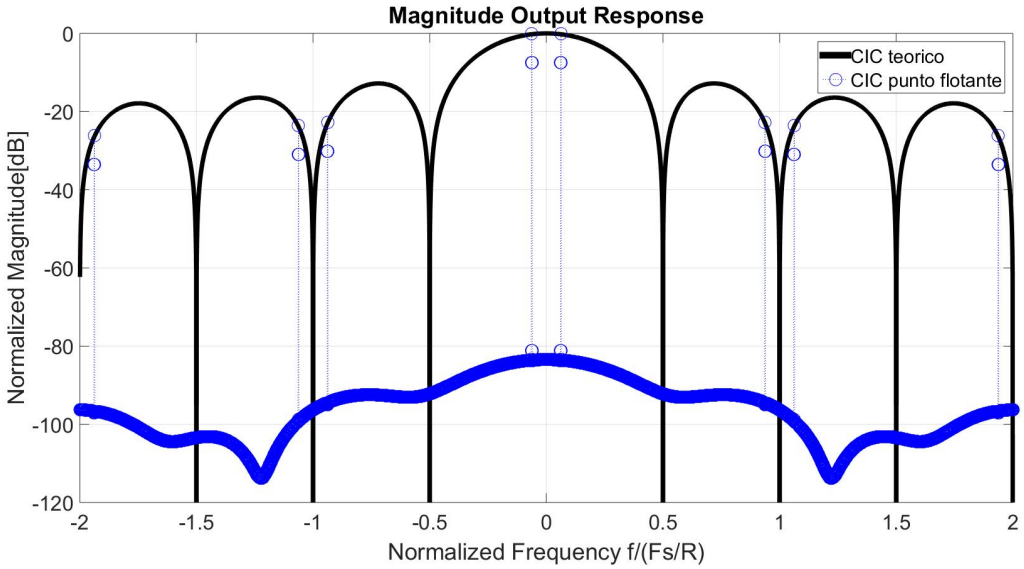


Figura 3.5: Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada senoidal de f=781.25Khz, A=0.5, phi=0,offset=0, simulada en punto flotante

3.2.3. Entrada de ruido de distribución uniforme

Para la generación de la entrada de ruido uniforme se utiliza el script del Anexo A.6.5. El mismo consiste en una función que tiene como parámetros de entrada la varianza y la media del ruido uniforme y la cantidad total de muestras a generar, configurados en el archivo *config*. Tiene como salida un vector x de la longitud configurada, construido con la función *rand* de MATLAB. Esta función genera valores aleatorios entre 0 y 1 por ello se le resta una media de 0.5 ya que nos interesa tener tanto valores positivos como negativos. Luego se lo multiplica por la raíz cuadrada de la varianza y se le suma la media.

Las gráficas de la señal de ruido uniforme siguiendo el caso de uso R=4, M=2 y N=1 a lo largo de todas las secciones del filtro se lo puede observar en la Figura 3.6.

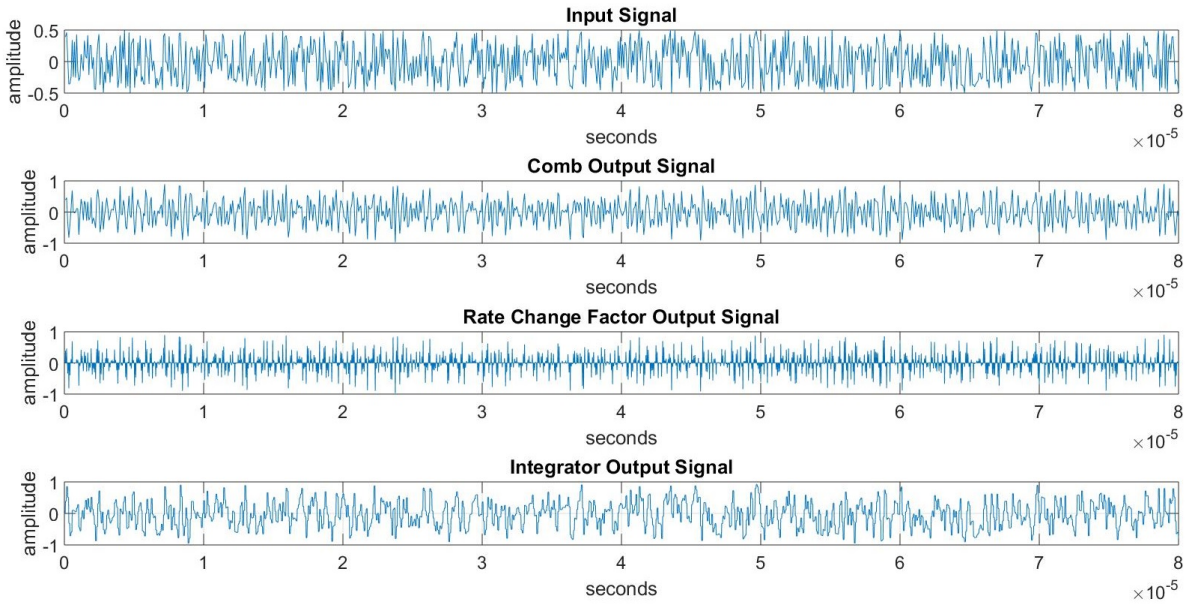


Figura 3.6: Respuestas en el tiempo de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada de ruido de media 0 y varianza 1, simulada en punto flotante

En el dominio de la frecuencia, en la Figura 3.7, se verifica que la salida del filtro simulada concuerda con la respuesta impulsiva del filtro teórico.

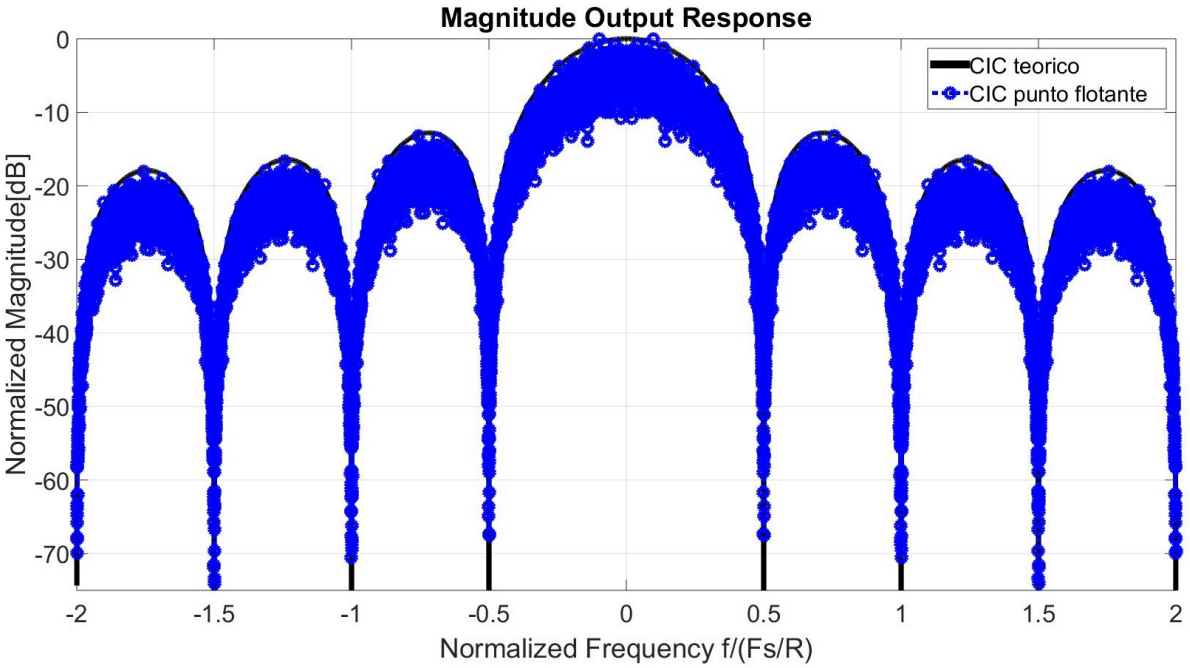


Figura 3.7: Respuestas en frecuencia de un filtro de interpolación CIC R=4, M=2 y N=1 para una entrada de ruido de media 0 y varianza 1, simulada en punto flotante

Capítulo 4

Diseño en punto fijo

Para el diseño de la arquitectura digital se utiliza DSP Builder. DSP Builder para Intel® FPGAs es una herramienta de diseño para DSP que permite la generación de código HDL de algoritmos de DSP directamente desde un entorno de SIMULINK [8].

SIMULINK es un paquete de software para modelado, simulación y análisis de sistemas dinámicos. Soporta sistemas lineales y no lineales modelados en tiempo continuo, por muestra, o un híbrido de los dos. Para el modelado, SIMULINK provee una interfaz gráfica de usuario (GUI) para construir modelos como diagramas de bloques.

DSP Builder de Intel Corporation incorpora modelos de bloques directamente sintetizables en una FPGA. Permitiendo la parametrización mediante variables del entorno de trabajo MATLAB.

En el presente capítulo se presentan los algoritmos implementados en DSP Builder y se verifican los mismos por simulación para entrada impulsiva, senoidal y de ruido de distribución uniforme.

4.1. Algoritmos de implementación

La implementación algoritmica en punto fijo se realiza mediante el diagramado de bloques de las ecuaciones desarrolladas en el capítulo anterior.

4.1.1. Filtro comb

La implementación del filtro comb es simplemente llevar la ecuación 3.3 a diagrama de bloques como se ilustra en la Figura 4.1.

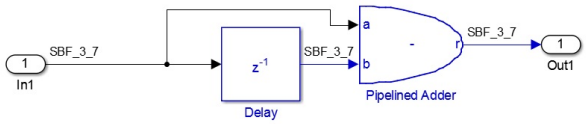


Figura 4.1: Diagrama de bloques de un filtro comb

4.1.2. Cambio de tasa

Se realiza el cambio de tasa mediante la utilización de dos bloques ilustrados en la Figura 4.2. El primer bloque es el encargado de especificar la tasa de sobremuestreo (20 ns), *Multi-Rate DFF(Flip Flop D)*, y el bloque subsiguiente (*Up Sampling*), se encarga de realizar la inserción de ceros, entre muestra y muestra a la nueva tasa especificada.

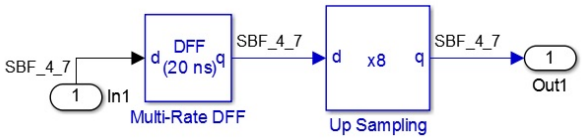


Figura 4.2: Diagrama de bloques del cambio de tasa

4.1.3. Filtro integrador

De la misma forma que el filtro comb, la implementación del filtro integrador es llevar la ecuación 3.5 a diagrama de bloques como se ilustra en la Figura 4.3.

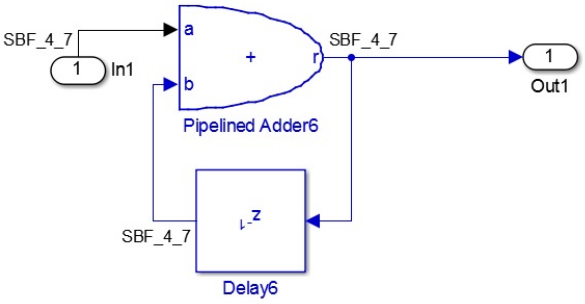


Figura 4.3: Diagrama de bloques de un integrador

4.1.4. Multietapa

La arquitectura de filtro de interpolación CIC se diseña para que pueda ser reconfigurado hasta un máximo de 6 etapas mediante la utilización de compuertas lógicas.

En la Figura 4.4 se ilustra el diseño de entrada del filtro interpolador CIC para N=5 y N=6 mediante el uso de un demultiplexor con 6 salidas seleccionada de acuerdo

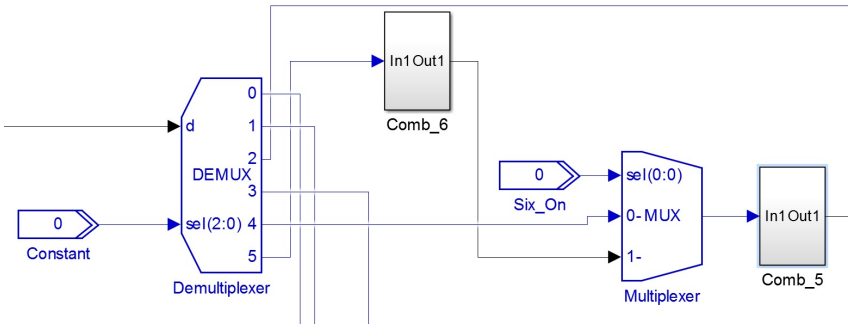


Figura 4.4: Modularización del la sección comb del filtro interpolador CIC

al parámetro N configurado del filtro. Para cada etapa del filtro subsiguiente se utiliza un multiplexor de 2 entradas que selecciona si el flujo de datos proviene de la etapa comb anterior o directamente de la salida del demultiplexor.

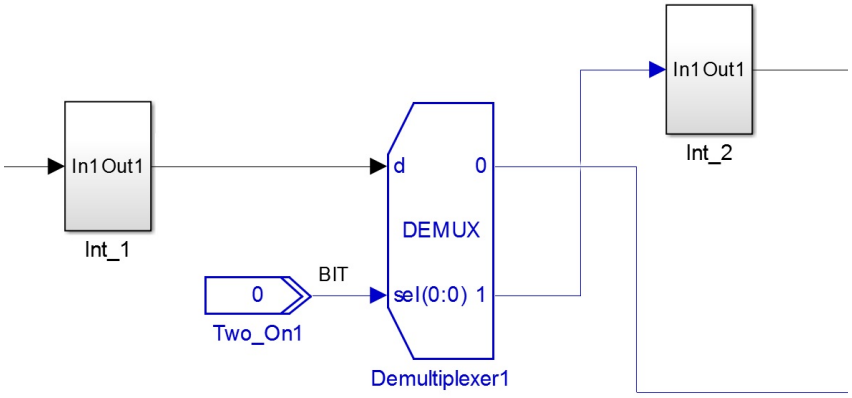


Figura 4.5: Modularización de la sección integradora del filtro interpolador CIC

Para la etapa integradora del filtro interpolador CIC se utiliza demultiplexores de 2 salidas que seleccionan si los datos provenientes de la etapa anterior deben insertarse al integrador siguiente o ser tomadados como salida del filtro. En la Figura 4.5 se ilustra el diseño para N=1 y N=2.

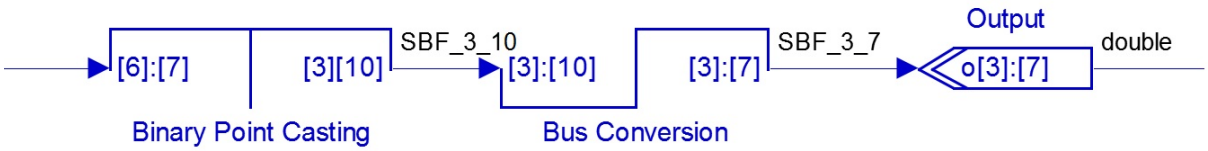


Figura 4.6: Diagrama de bloques de la salida del filtro interpolador CIC

La representación numérica elegida a lo largo de todo el diseño es fraccional con signo y se procede a realizar un truncamiento a la salida de acuerdo a la resolución configurada desde el entorno MATLAB. El truncamiento se realiza mediante 3 bloques

concatenados como ilustra la Figura 4.6. El primer bloque se encarga de adecuar la representación binaria a la resolución de la parte entera configurada a la salida (*Binary Point Casting*). Después se procede a tomar los bits más significativos configurados (*Bus Conversion*) y finalmente el último bloque se encarga de mapear los pines de salida (*Output*).

El diseño contempla el crecimiento de la palabra etapa a etapa calculado según [3]. El incremento se realiza aumentando la cantidad de bits pertenecientes a las unidades enteras del formato de salida, conservando la misma resolución para la parte fraccional de su representación.

4.2. Verificación por simulación

Para la verificación del diseño por simulación se cuenta con bancos de prueba para entrada impulsiva, senoidal y de ruido de distribución uniforme. Los mismos permiten la parametrización de su configuración mediante el entorno de trabajo de MATLAB.

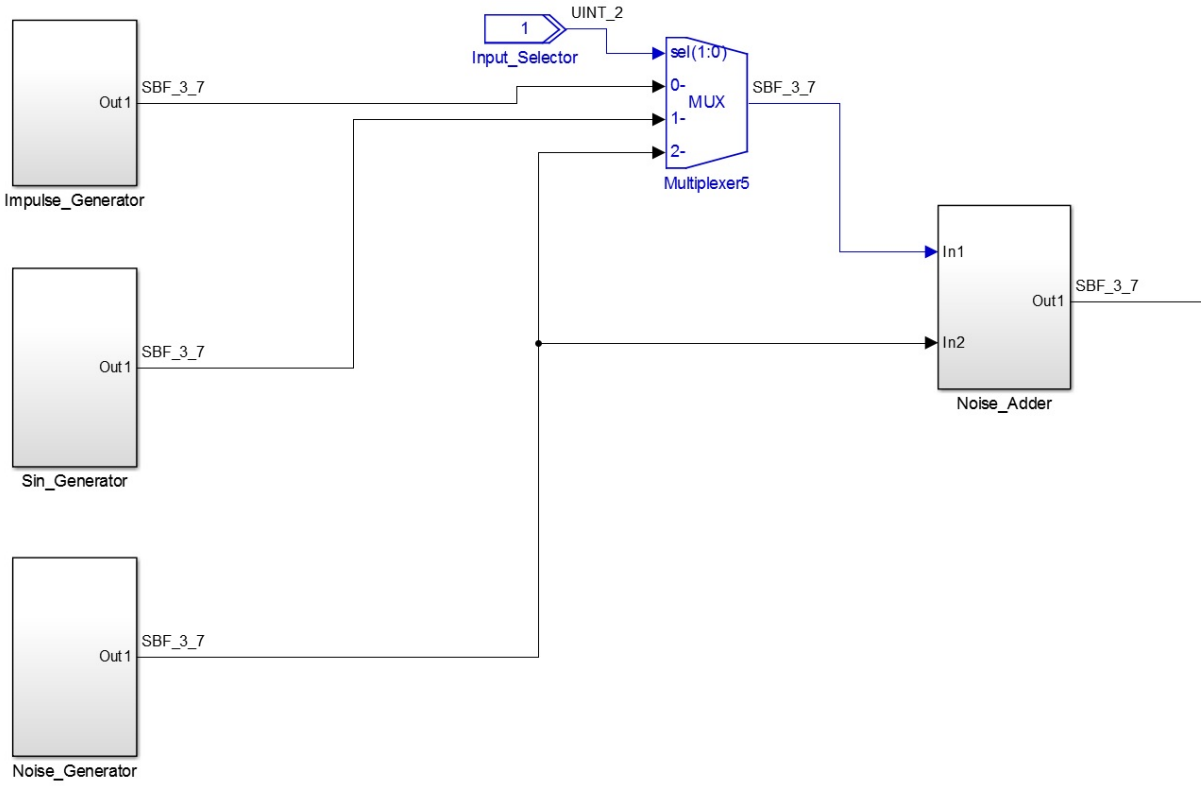


Figura 4.7: Diagrama de bloques de entradas del filtro interpolador CIC

En la Figura 4.7 se ilustra el diagrama de bloques de la entrada al filtro interpolador CIC. El mismo cuenta con un multiplexor que selecciona el tipo de entrada y luego cuenta con un bloque encargado de agregar ruido a la señal seleccionada de acuerdo al parámetro seteado *Noise Add* del archivo *config* de MATLAB.

El diseño cuenta con 2 relojes, siendo uno el reloj principal del sistema correspondiente a $\frac{1}{f_s}$ y otro un reloj derivado mediante un PLL que corresponde a $\frac{R}{f_s}$.

4.2.1. Entrada impulsiva

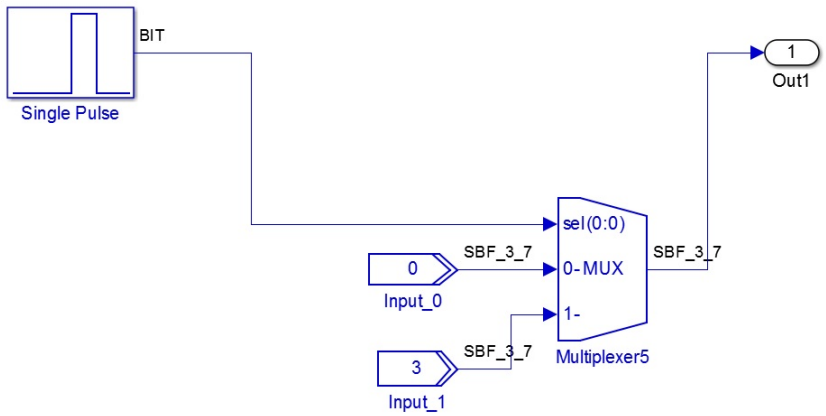


Figura 4.8: Diagrama de bloques del generador de entrada impulsiva

Como ilustra la Figura 4.8, la generación de la entrada impulsiva utiliza un multiplexor con 2 entradas. Una de las entradas es una constante de 0 y la otra es una constante configurada al valor seteado en *Impulse Amplitude* en el archivo *config* de MATLAB. La selección del multiplexor se realiza mediante el bloque *Single Pulse* que genera un sólo impulso luego de $M + 1$ muestras.

Se procede a verificar el diseño para el caso de uso R=4, M=2 y N=1 obteniendo un resultado satisfactorio como se ilustra en la Figura 4.9. Los puntos de simulación en punto fijo coindicen plenamente con los simulados en punto flotante. Para la resolución en punto fijo se utilizan 10 bits de entrada sin parte fraccional y 10 bits de salida sin parte fraccional y un impulso de amplitud 511 para conseguir la máxima excursión dada la resolución elegida.

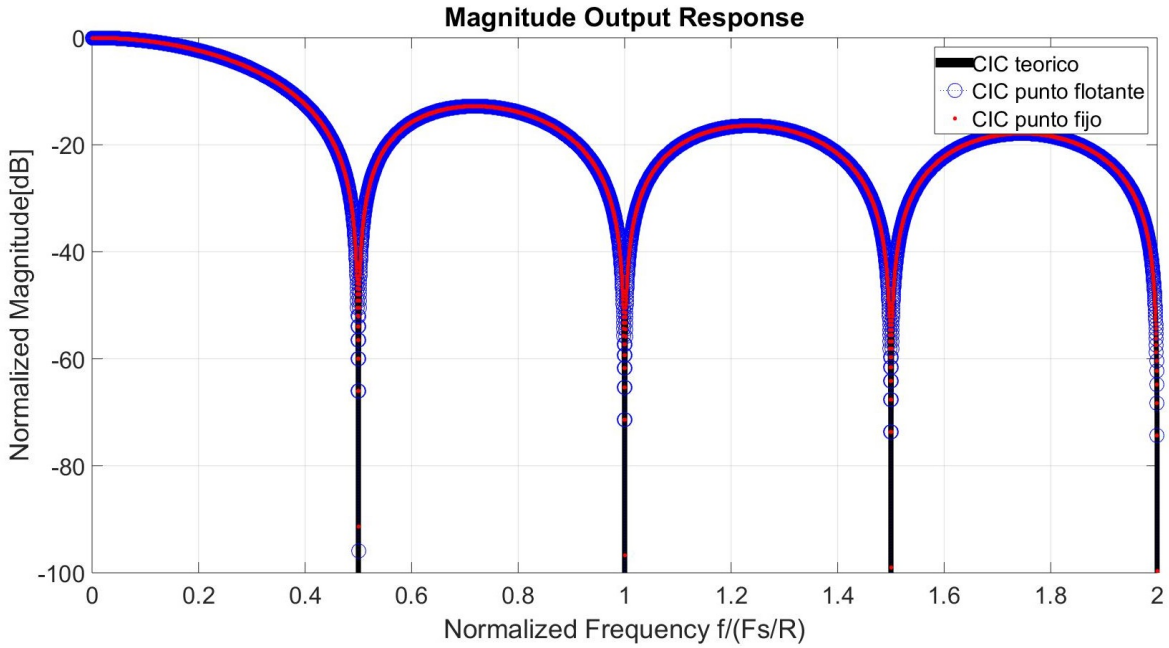


Figura 4.9: Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada impulsiva de amplitud 511 simulada en punto fijo

4.2.2. Entrada de onda senoidal

Como entrada de señal senoidal se cuenta con 2 tipos de entradas configurables, como se ilustra en la Figura 4.10. El seleccionado entre las 2 entradas es mediante un multiplexor. La primera es la digitalización de la señal de onda senoidal generada en punto flotante y la segunda es un generador de onda senoidal íntegramente sintetizable en la FPGA mediante una LUT que se explicará en detalle en la sección siguiente.

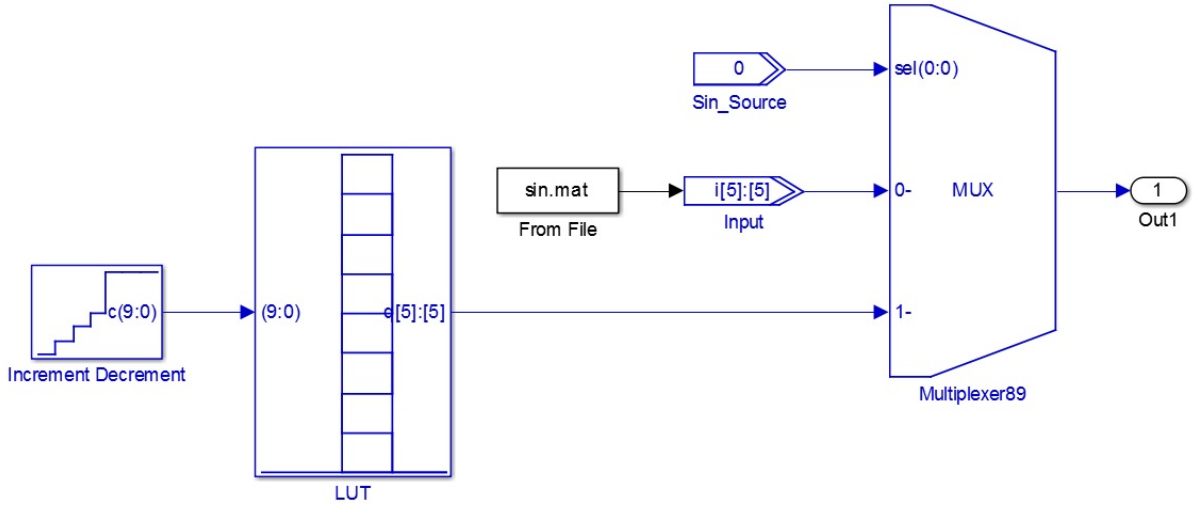


Figura 4.10: Diagrama de bloques del generador de onda senoidal

Se procede a verificar el diseño para el caso de uso R=4, M=2 y N=1 obteniendo un resultado satisfactorio como se ilustra en la Figura 4.11. Los puntos de simulación de interés en punto fijo, correspondientes al tono central de la onda senoidal y las imágenes, coindicen plenamente con los simulados en punto flotante. Se utiliza la fuente de generación de onda senoidal de punto flotante para la simulación y para la resolución en punto fijo se utilizan 10 bits de entrada siendo 5 para la parte fraccional y 5 para la parte entera. Como salida se utiliza la misma resolución de entrada. La onda senoidal generada es de amplitud 15 para conseguir la máxima excursión dada la resolución elegida. Su frecuencia es de 781,25 KHz y no posee desfasaje ni offset de continua.

Nótese la aparición de una pequeña componente de continua de $-52{,}45$ dB. Esta componente de continua que aparece a la salida del filtro interpolador CIC se debe a la discretización de la señal a la entrada, ruido propio inherente al filtro y el truncamiento que se realiza a la salida del filtro.

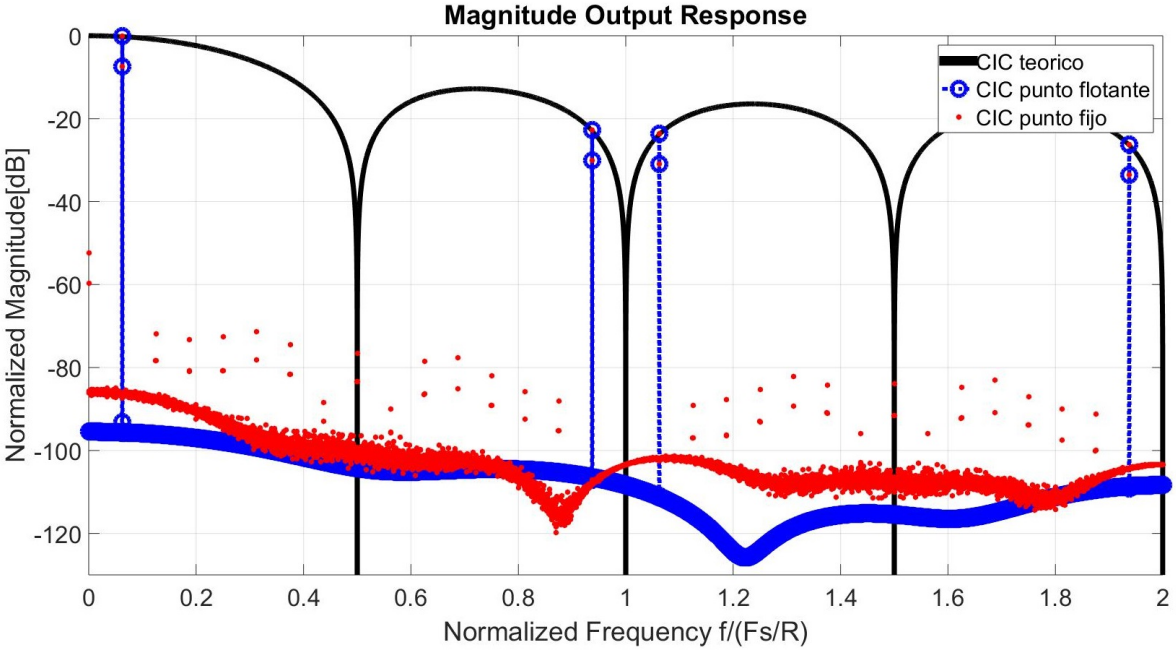


Figura 4.11: Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada senoidal de amplitud 15, frecuencia 781,25 KHz, sin desfasaje ni offset de continua, simulada en punto fijo

4.2.3. Entrada de ruido de distribución uniforme

Como entrada de ruido de distribución uniforme se cuenta con 2 tipos de entradas configurables, como se ilustra en la Figura 4.13. El seleccionado entre las 2 entradas es mediante un multiplexor. La primera es la digitalización de la señal de ruido generada en punto flotante y la segunda es un generador de ruido de distribución uniforme íntegramente sintetizable en la FPGA, llamado generador de Tausworthe [18].

El generador de Tausworthe genera un número aleatorio de 32 bits de los cuales se extraen la cantidad de bits especificada en el archivo *config* y, mediante el mismo proceso de extracción de bits aplicado a la salida del filtro, se le configura el punto fraccional. Una explicación más detallada de este generador se encuentra en la sección siguiente.

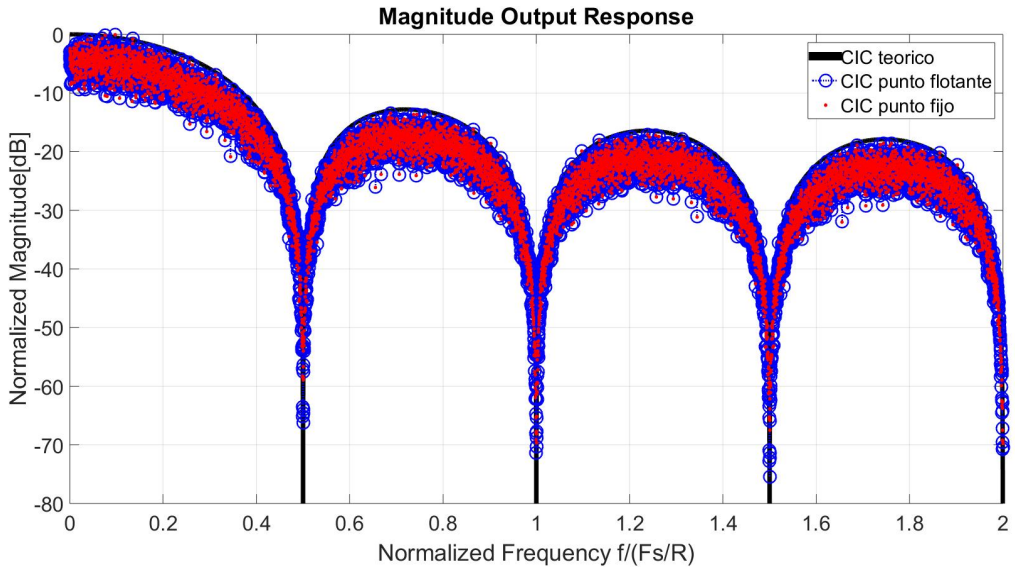


Figura 4.12: Salida en frecuencia normalizada de un filtro interpolador CIC R=4, M=2, N=1 con entrada de ruido con media 0 y varianza 1 simulada en punto fijo

Se procede a verificar el diseño para el caso de uso R=4, M=2 y N=1 obteniendo un resultado satisfactorio como se ilustra en la Figura 4.12. Los puntos de simulación en punto fijo, coindicen plenamente con los simulados en punto flotante. Se utiliza la fuente de generación de ruido de distribución uniforme de punto flotante para la simulación y para la resolución en punto fijo se utilizan 10 bits de entrada siendo 8 para la parte fraccional y 2 para la parte entera. Como salida se utiliza la misma resolución de entrada.

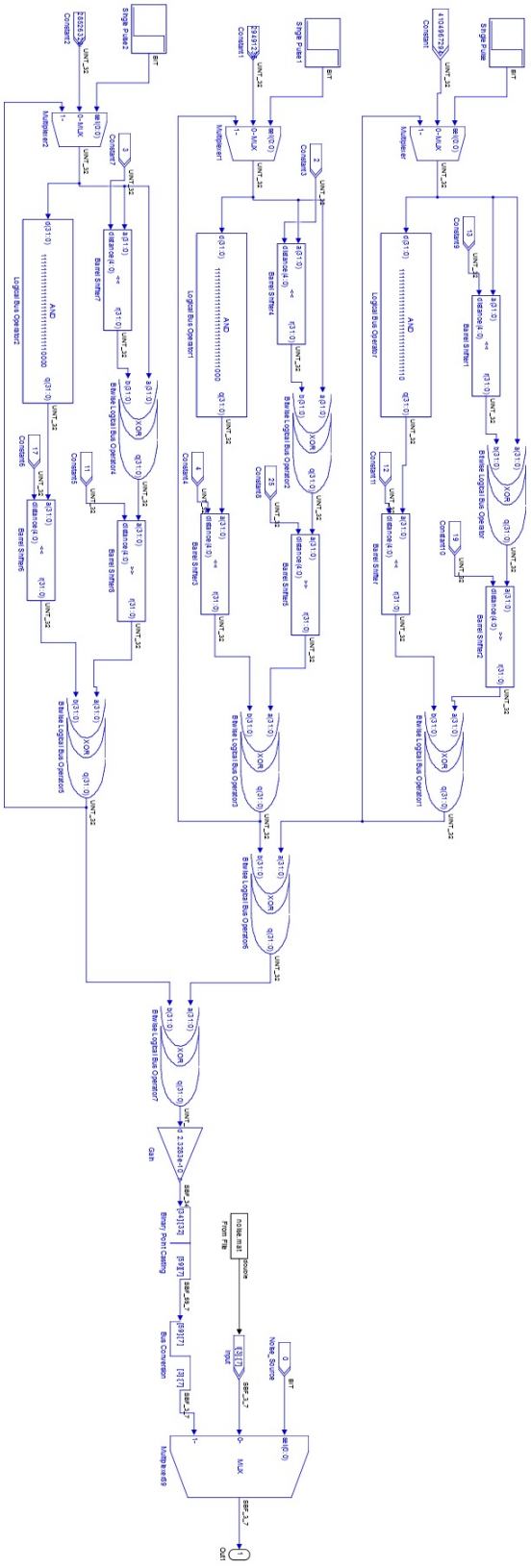


Figura 4.13: Diagrama de bloques del generador de ruido de distribución uniforme

4.2.4. Verificación comparativa

Se realiza una verificación comparativa del filtro interpolador CIC diseñado con un ejemplo diseñado por Altera con los parámetros R=75, M=1 y N=3. Se reconfigura el filtro diseñado con esos parámetros para realizar la comparación de ambas salidas para diferentes señales de entrada: impulsiva, de onda senoidal y de ruido de distribución uniforme. Se mantiene el banco de prueba diseñado para las verificaciones anteriores y se contrasta con la curva de salida en punto flotante. Para todos los casos se mantuvo una f_s de 50 Mhz y se toma una resolución, tanto a la salida como a la entrada del filtro, de 10 bits.

El diseño de Altera utilizado para la comparación se encuentra en la Figura 4.14. En él, se realiza el cálculo del ancho de la palabra para el máximo crecimiento a través del filtro y se configura el ancho de la palabra de todas las secciones del filtro con este valor.

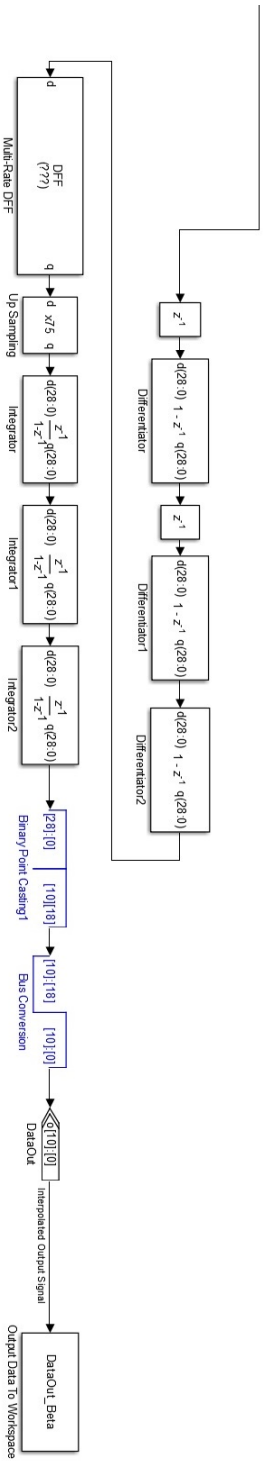


Figura 4.14: Diseño de Altera de un filtro interpolador CIC R=75, M=1 y N=3 utilizado para la verificación el filtro interpolador CIC propio diseñado

Entrada Impulsiva

Se simula una entrada impulsiva de 10 bits sin parte fraccional con una amplitud de 511, obteniendo la gráfica de la Figura 4.15. Se observa una separación de las curvas simuladas en punto fijo con la curva de punto flotante debido al truncamiento que se realiza a la salida de ambos filtros.

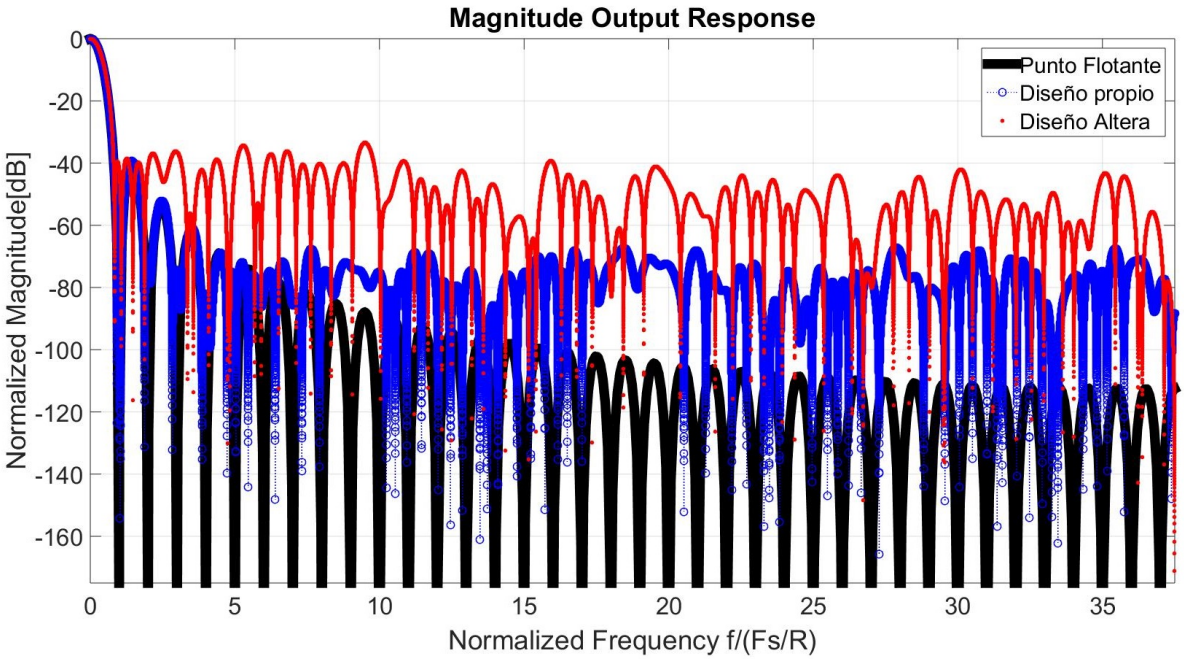


Figura 4.15: Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada impulsiva de amplitud 511

Se observa una notable mejor respuesta en el filtro diseñado. Pudiendo seguir la curva de punto flotante hasta el 4to lóbulo lateral, mientras que el diseño de altera sólo logra seguir hasta el 1er lóbulo lateral, obteniendo una diferencia de más de 20dB de atenuación.

Entrada Senoidal

Se simula una entrada senoidal de 10 bits, siendo 5 bits para la parte fraccional y 5 bits para la parte entera, obteniendo la gráfica de la Figura 4.16. La onda senoidal generada es de amplitud 15, de frecuencia 390,625 Khz y de offset de fase de $\frac{\pi}{4}$. Representa el punto normalizado 0,0625 en relación a la frecuencia más baja. A la salida de ambos filtros se utiliza la misma resolución adoptada para la entrada del mismo.

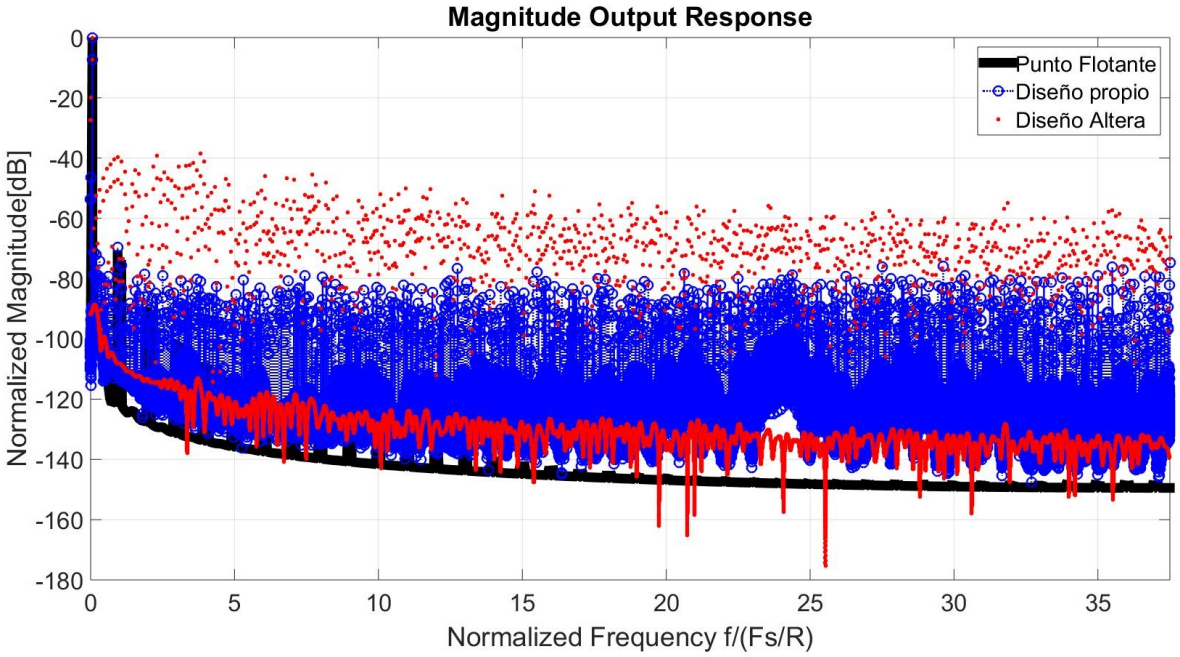


Figura 4.16: Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada senoidal de amplitud 15, de frecuencia 390,625 Khz y de offset de fase de $\frac{\pi}{4}$

En la Figura 4.17, se realiza un acercamiento en las frecuencias más bajas para observar mejor la magnitud de las componentes de continua.

Nuevamente se obtuvo una mejor respuesta en el diseño propio. Resultando una componente de continua de $-19{,}89$ dB en el diseño de Altera y una componente de continua de $-46{,}35$ dB en el diseño propio.

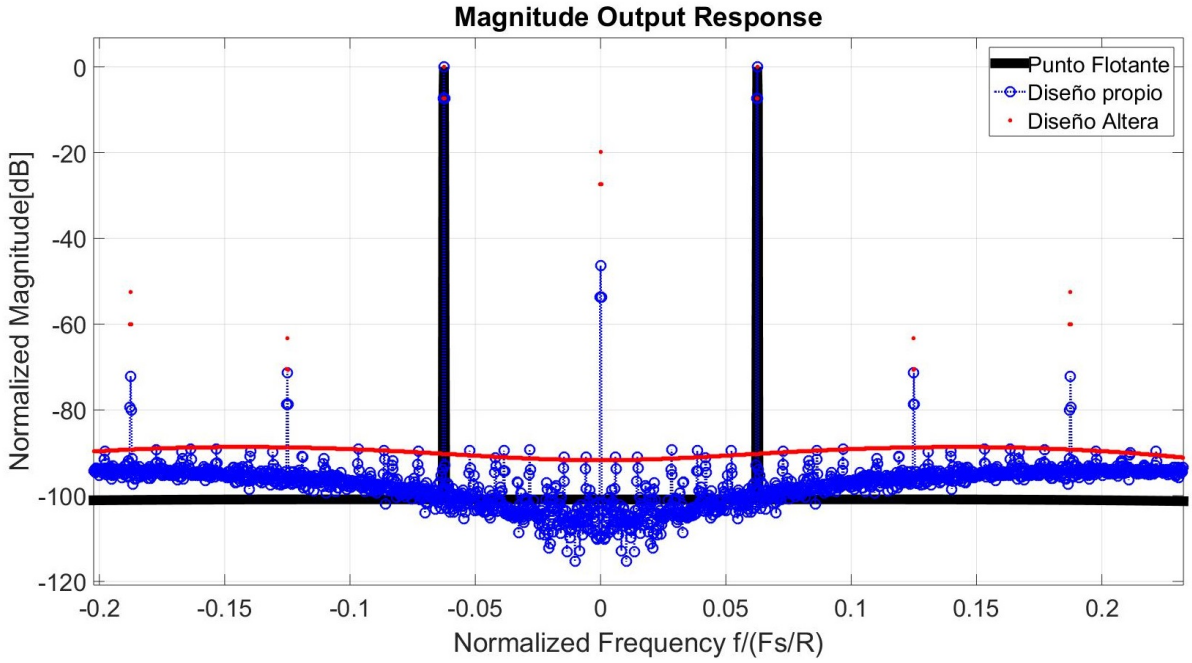


Figura 4.17: Acercamiento de la salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada senoidal de amplitud 15, de frecuencia 390,625 Khz y de offset de fase de $\frac{\pi}{4}$

Entrada de ruido de distribución uniforme

Se simula una entrada de ruido de 10 bits, siendo 8 bits para la parte fraccional y 2 bits para la parte entera, obteniendo la gráfica de la Figura 4.18. El ruido generado es de media 0 y varianza 1. A la salida de ambos filtros se utiliza la misma resolución adoptada para la entrada del mismo.

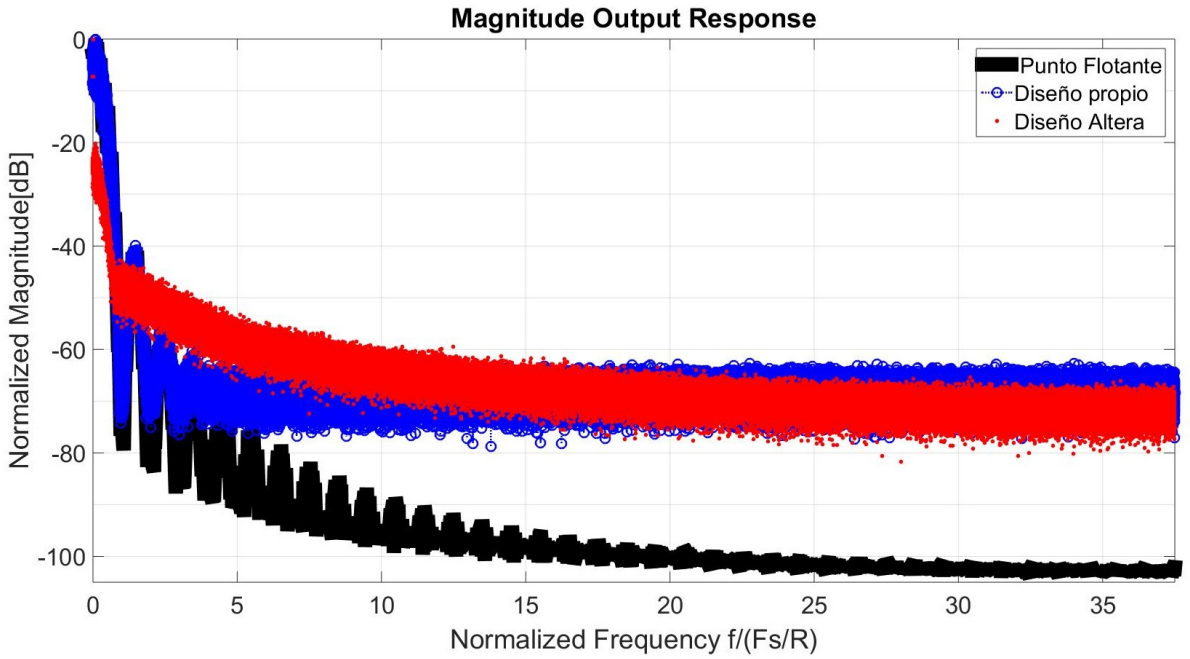


Figura 4.18: Salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada de ruido de media 0 y varianza 1

Se puede observar, en la Figura 4.19, la aparición de una componente de continua en el diseño de altera de aproximadamente -25 dB.

De esta forma podemos concluir que el diseño propio del filtro de interpolación CIC realizado en DSP Builder cuenta con una mejor respuesta tomando como referencia el filtro diseñado por Altera para los valores de R=75, M=1 y N=3.

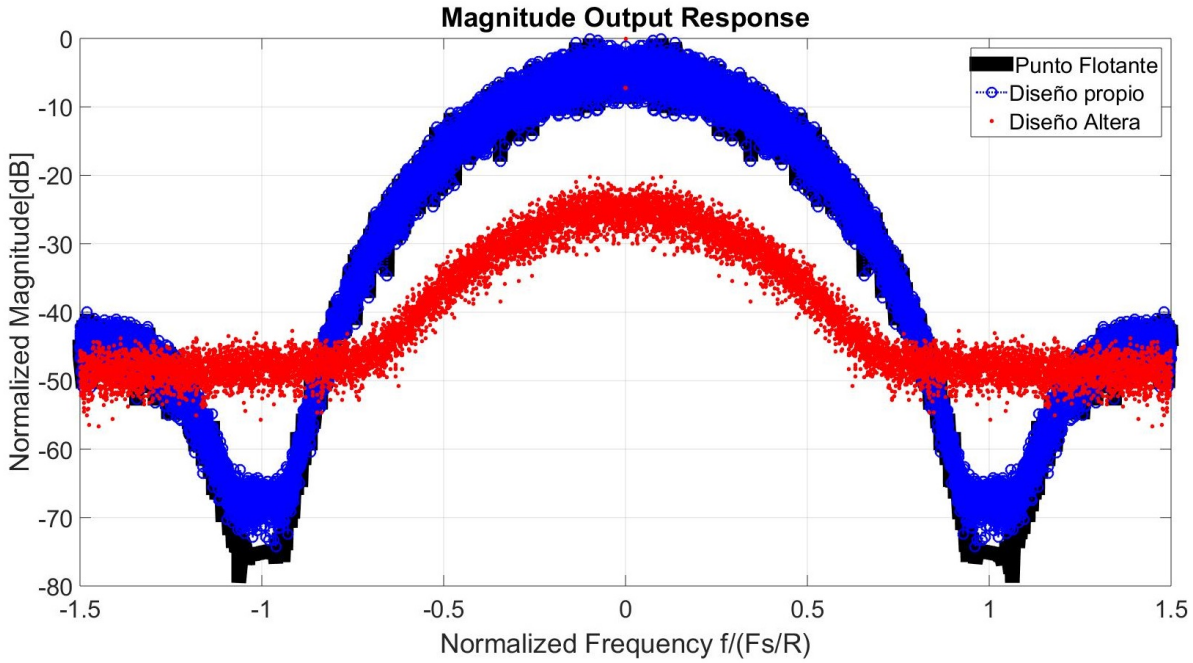


Figura 4.19: Acercamiento de la salida comparativa en frecuencia normalizada de un filtro interpolador CIC R=75, M=1, N=3 con entrada de ruido de media 0 y varianza 1

Capítulo 5

Validación en FPGA

En el presente capítulo se realiza la validación del diseño mediante sintetizado en FPGA de la arquitectura multietapa digital diseñada y se toman mediciones mediante osciloscopio y analizador de espectro. Luego se analizan los resultados obtenidos y se comparan con los resultados de simulación en punto fijo.

5.1. Entrada Impulsiva

5.1.1. Hardware utilizado

Se procede a validar el diseño mediante entrada impulsiva en una placa DE2 con una FPGA Cyclone II EP2C35F672C6 y se utiliza un osciloscopio Yokogawa DL1520L para medir la respuesta temporal del filtro interpolador CIC R=8, M=2 para valores N=1 y N=3, a una f_s de 50Mhz. La salida de la FPGA se toma a través de la salida VGA de la placa utilizando el DAC del color rojo.

5.1.2. Generación de tren de impulsos

Debido a que un sólo impulso resulta inviable para tomar una medición, se diseña una arquitectura para obtener un tren de impulsos que permita darle el tiempo necesario de respuesta al filtro entre pulso y pulso, y, a su vez, poder observar su respuesta en el osciloscopio.

La Figura 5.1 ilustra la resolución adoptada. Se utiliza un contador de 5 bits a la frecuencia f_s y se procede a dividir el bus del contador bit a bit. De esa forma el bit 3 cuenta con un impulso de 160 ns y mediante una operación AND entre los bits 3, 4 y 5 se obtiene una separación de 1,12 us entre pulso y pulso.

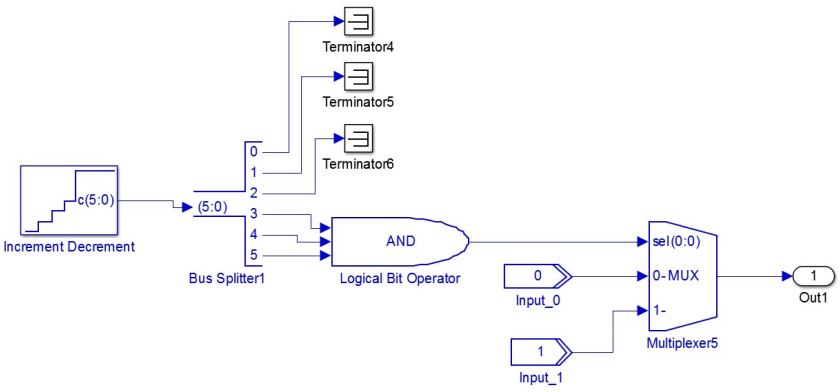


Figura 5.1: Generación de tren de impulsos para validación en FPGA Cyclone II EP2C35F672C6

5.1.3. Mediciones

Como primera instancia, se verifica que la entrada del filtro interpolador CIC sea la configurada. Se debe obtener un impulso de período 1,28 us, con una duración de 160 ns y una separación entre pulso y pulso de 1,12 us, como se verifica en las Figuras 5.2 y 5.3.

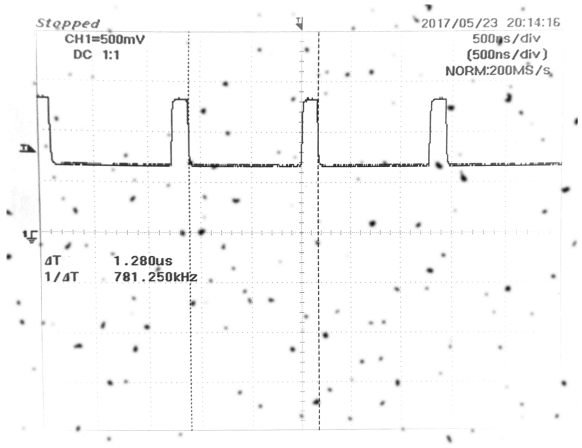


Figura 5.2: Medición del período de la señal de entrada del tren de impulsos

Luego, se procede a verificar que la salida del filtro interpolador CIC concuerde con los parámetros configurados.

Para R=8, M=2 y N=1, se debe observar a la salida del filtro un pulso cuadrado del doble del ancho del impulso de entrada, es decir, de 320 ns. Verificado correctamente en la Figura 5.4.

La frecuencia de la señal se debe mantener, ya que sólo trabajamos en las muestras. Por ello, la separación entre los pulsos de la señal de salida del filtro debe ser de 960 ns, como se verifica en la Figura 5.5.

Por último, elevando el número de etapas del filtro, para R=8, M=2 y N=3,

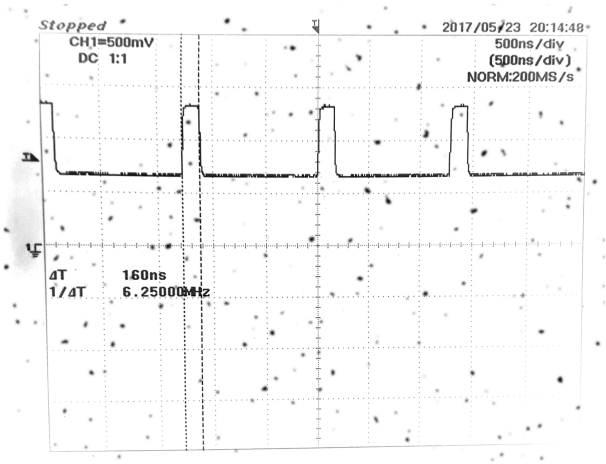


Figura 5.3: Medición del ancho del pulso de la señal de entrada del tren de impulsos

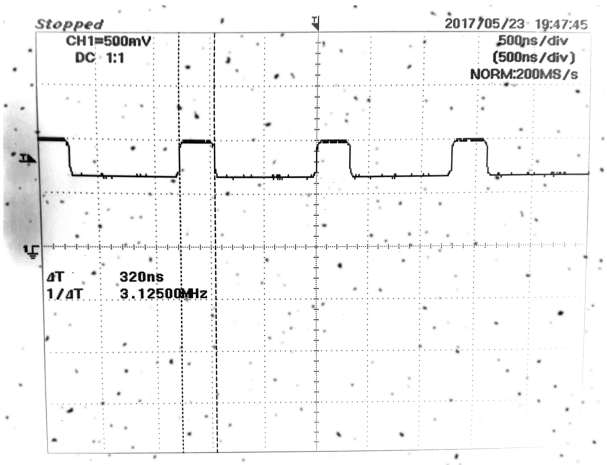


Figura 5.4: Medición del ancho del pulso de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada de tren de impulsos

se debe obtener a la salida del filtro una señal de la misma frecuencia que para N=1. La única salvedad es que, a medida que se aumentan las etapas del filtro, el pulso conformado a la salida se suaviza en forma de campana. Se verifica la correcta respuesta del filtro diseñado en la Figura 5.6.

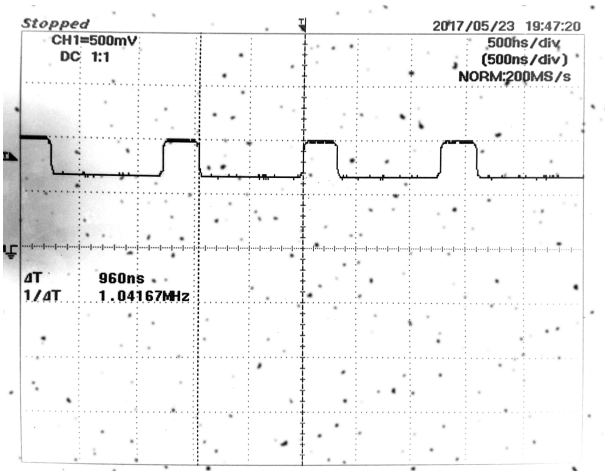


Figura 5.5: Medición de la separación entre pulsos de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada de tren de impulsos

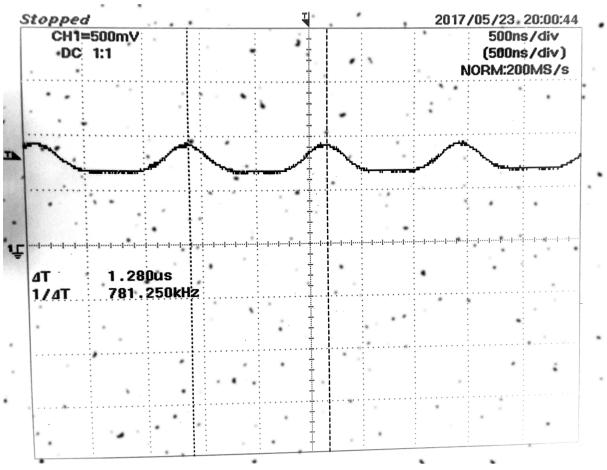


Figura 5.6: Medición del periodo de la señal de salida del filtro de interpolación CIC para los parámetros R=8, M=2 y N=3 y entrada de tren de impulsos

5.2. Entrada Senoidal

5.2.1. Hardware utilizado

Se procede a validar el diseño mediante entrada senoidal en una placa DE2 con una FPGA Cyclone II EP2C35F672C6 y se utiliza un osciloscopio Yokogawa DL1520L para medir la respuesta temporal y en frecuencia del filtro interpolador CIC R=8, M=2 y N=1, a una f_s de 50Mhz. La salida de la FPGA se toma a través de la salida VGA de la placa utilizando el DAC del color rojo.

5.2.2. Generación de onda senoidal

Para la generación de la onda senoidal se utliza una Look Up Table (LUT) que almacena los valores de muestras de la señal senoidal generada en punto flotante y luego se la recorre mediante un contador incremental. Al fin de evitar discontinuidades de la onda senoidal generada, se deben almacenar períodos enteros de la onda senoidal para variar su frecuencia. Por ello, existe una limitación en las frecuencias de la onda senoidal que se pueden generar.

Se automatiza el cálculo de la onda senoidal mediante un script de MATLAB encontrado en el Anexo A.5. En él se define la resolución de la señal de entrada y el punto de salida en frecuencia normalizada que se desea obtener. En base a estos parámetros calcula la frecuencia de la onda senoidal más próxima al punto deseado y reconfigura algunos parámetros de simulación para que concidan con los nuevos valores.

5.2.3. Mediciones

Se realizan las mediciones de la atenuación de la señal de salida temporal y de la atenuación de las imágenes espectrales propias del proceso de sobremuestreo. Los resultados se muestran en las Tablas 5.1, 5.2, 5.3 y 5.4.

Tabla 5.1: Resultados de medición temporal de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal

LUT Periods	Sin Freq [KHz]	Vin [V]	Vout [V]	Attenuation [dB]	Relative Attenuation [dB]	Normalized Point	Power Response [W]	Desired Attenuation [dB]	Error [dB]	Figure
1	3.05	1,312	1,312	0	0,00	0,000488482	255,9991993	0,00	0,00	5.7
2	6.10	1,312	1,312	0	0,00	0,000976964	255,9967972	0,00	0,00	5.9
4	12.21	1,312	1,312	0	0,00	0,001953927	255,9871889	0,00	0,00	5.11
8	24.42	1,312	1,312	0	0,00	0,003907855	255,9487586	0,00	0,00	5.13
16	48.85	1,312	1,312	0	0,00	0,007815709	255,7950836	0,00	0,00	5.15
32	97.69	1,312	1,312	0	0,00	0,015631418	255,1811202	-0,01	0,01	5.17
64	195.39	1,312	1,312	0	0,00	0,031262837	252,7370185	-0,06	0,06	5.19
128	390.78	1,312	1,312	0	0,00	0,062525673	243,1466319	-0,22	0,22	5.21
256	781.57	1,312	1,216	-0,660005202	-0,66	0,125051346	207,6360498	-0,91	0,25	5.23
512	1563.14	1,312	0,944	-2,859236815	-2,86	0,250102692	104,0016416	-3,91	1,05	5.25

Tabla 5.2: Resultados de medición de atenuación de la 1º,2º y 3º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal

LUT Periods	Sin Freq [KHz]	1º Image			2º Image			3º Image			Figure
		Freq [MHz]	Simulation [dB]	Measure [dB]	Freq [MHz]	Simulation [dB]	Measure [dB]	Freq [MHz]	Simulation [dB]	Measure [dB]	
1	3.05	6,25	-66,52	-62,8	6,25	-66,52	-62,8	12,5	-73,54	-64,9	5.8
2	6.10	6,24375	-60,49	-56,3	6,25625	-60,51	-56,3	12,49	-67,42	-59,8	5.10
4	12.21	6,2375	-54,46	-46,4	6,2625	-54,5	-46,4	12,4875	-61,4	-50,4	5.12
8	24.42	6,22	-48,42	-48,4	6,275	-48,56	-48,4	12,475	-55,37	-53,7	5.14
16	48.85	6,2	-42,37	-41,8	6,3	-42,52	-42	12,45	-49,32	-48,7	5.16
32	97.69	6,15	-36,26	-36,2	6,35	-36,63	-36,4	12,4	-43,25	-43,1	5.18
64	195.39	6,055	-30,08	-30,3	6,44	-30,74	-30,9	12,29	-37,13	-38,3	5.20
128	390.78	5,8	-23,77	-23,6	6,6	-24,98	-24,8	12,1	-30,9	-30,5	5.22
256	781.57	5,46	-17,12	-17,4	7	-19,72	-19,6	11,7	-24,86	-25,2	5.24
512	1563.14	4,7	-9,73	-9,8	7,8	-14,46	-14,5	10,9	-17,83	-17,9	5.26

Tabla 5.3: Resultados de medición de atenuación de la 4º,5º y 6º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal

LUT Periods	Sin Freq [KHz]	4º Image			5º Image			6º Image			Figure
		Freq [MHz]	Simulation [dB]	Measure [dB]	Freq [MHz]	Simulation [dB]	Measure [dB]	Freq [MHz]	Simulation [dB]	Measure [dB]	
1	3.05	12,5	-73,54	-64,9	18,75	-78,64	-64,4	18,75	-78,64	-64,4	5.8
2	6.10	12,51	-67,43	-59,8	18,74	-72,67	-59,8	18,76	-72,67	-59,8	5.10
4	12.21	12,5125	-61,42	-50,4	18,7375	-66,47	-52,9	18,7625	-66,48	-52,9	5.12
8	24.42	12,525	-55,52	-53,7	18,725	-60,43	-57,7	18,775	-60,64	-57,7	5.14
16	48.85	12,55	-49,52	-48,5	18,7	-54,4	-52	18,8	-54,64	-54	5.16
32	97.69	12,6	-43,54	-43,2	18,65	-48,33	-46,1	18,85	-48,64	-45,7	5.18
64	195.39	12,69	-37,59	-38,6	18,55	-42,24	-43,3	18,94	-42,65	-42,3	5.20
128	390.78	12,89	-31,67	-31	18,36	-36,05	-33,8	19,14	-36,85	-34,3	5.22
256	781.57	13,3	-26,15	-26,3	17,9	-29,92	-29,8	19,5	-31,26	-30,1	5.24
512	1563.14	14,1	-20,61	-20,2	17,2	-23,11	-22,1	20,3	-25,5	-23,2	5.26

Tabla 5.4: Resultados de medición de atenuación de la 7º imagen de salida del filtro interpolador CIC R=8, M=2 y N=1 para entrada senoidal

LUT Periods	Sin Freq [KHz]	7º Image			Figure
		Freq [MHz]	Simulation [dB]	Measure [dB]	
1	3.05	25	-83,24	-62,7	5.8
2	6.10	24,99	-77,27	-63,1	5.10
4	12.21	24,9875	-71,24	-61,5	5.12
8	24.42	24,975	-65,22	-62,7	5.14
16	48.85	24,95	-59,2	-54	5.16
32	97.69	24,9	-53,14	-47,9	5.18
64	195.39	24,8	-47,05	-45,8	5.20
128	390.78	24,6	-40,88	-38,2	5.22
256	781.57	24,2	-34,77	-31,3	5.24
512	1563.14	23,4	-27,9	-26,5	5.26

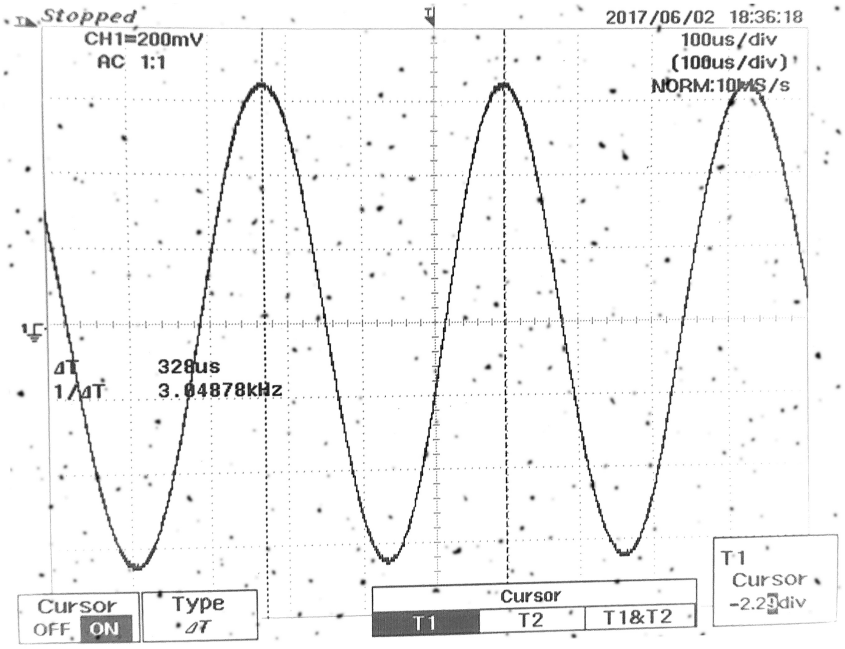


Figura 5.7: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 1 período en LUT

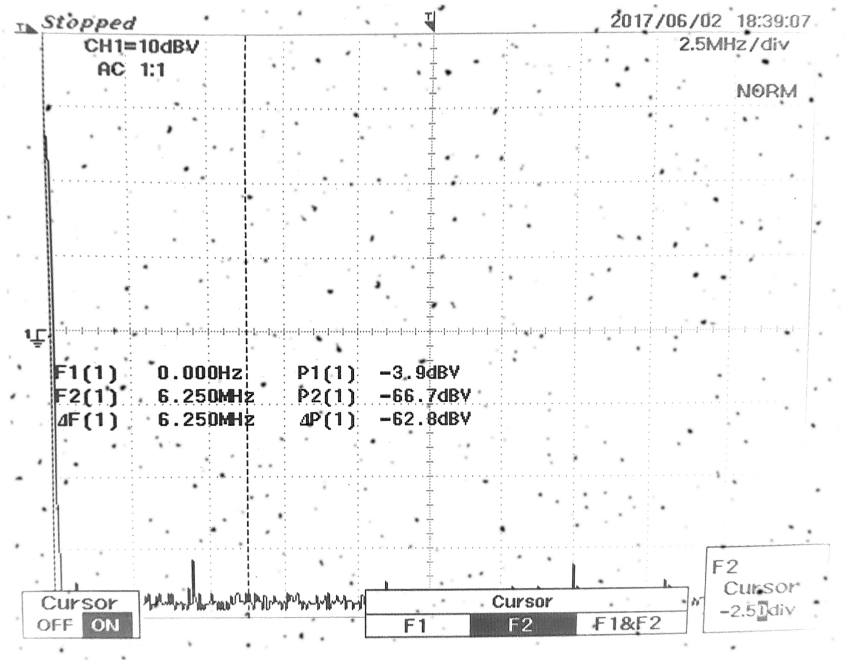


Figura 5.8: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 1 período en LUT

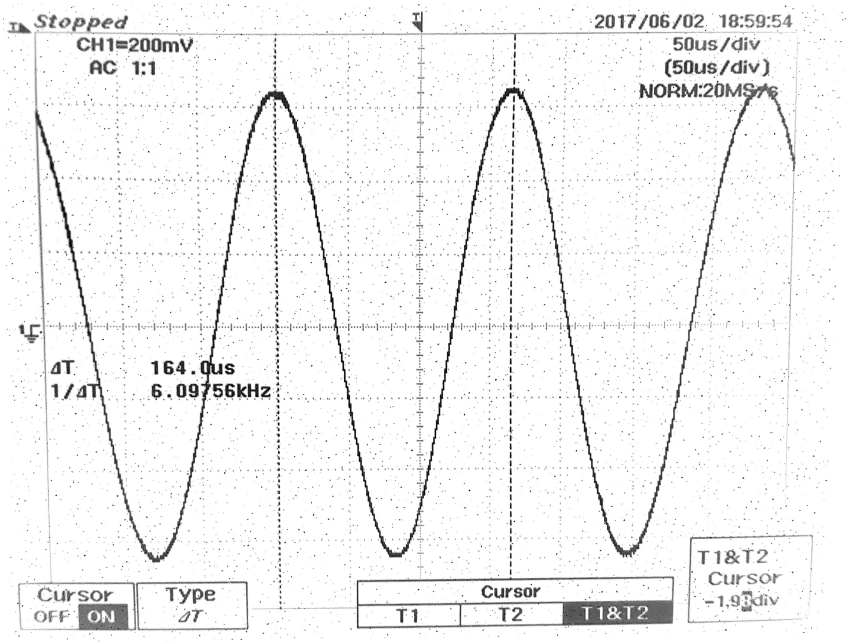


Figura 5.9: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 2 períodos en LUT

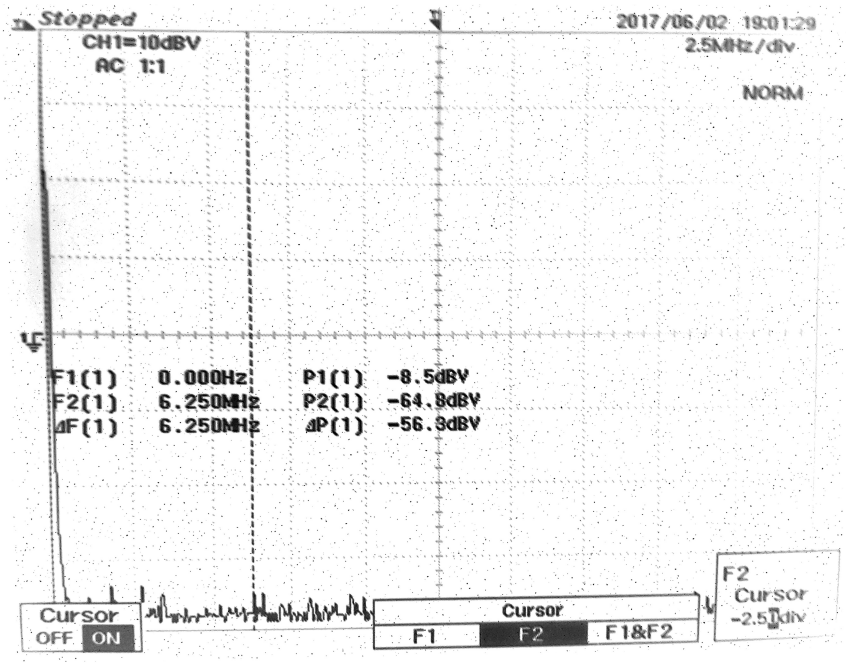


Figura 5.10: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 2 períodos en LUT

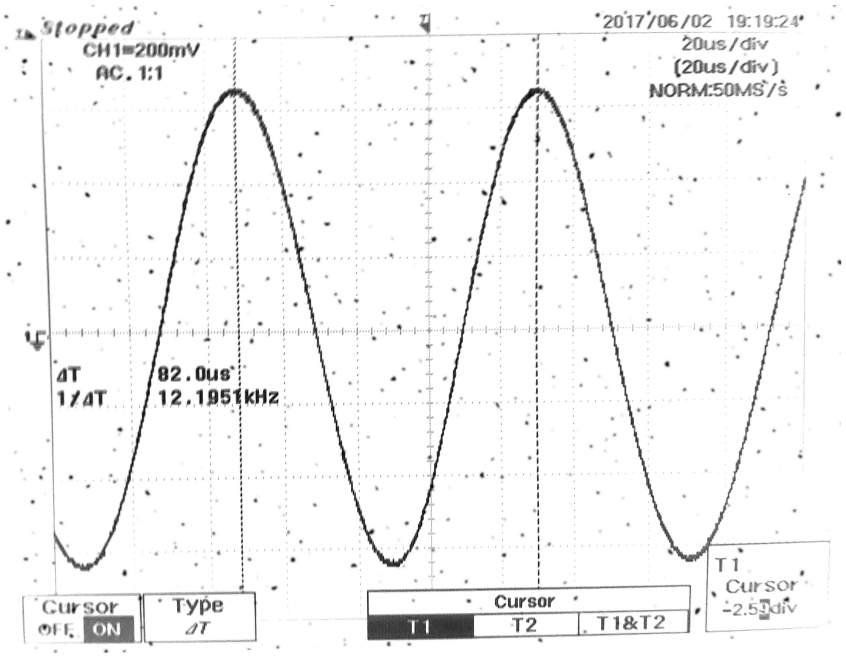


Figura 5.11: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 4 períodos en LUT

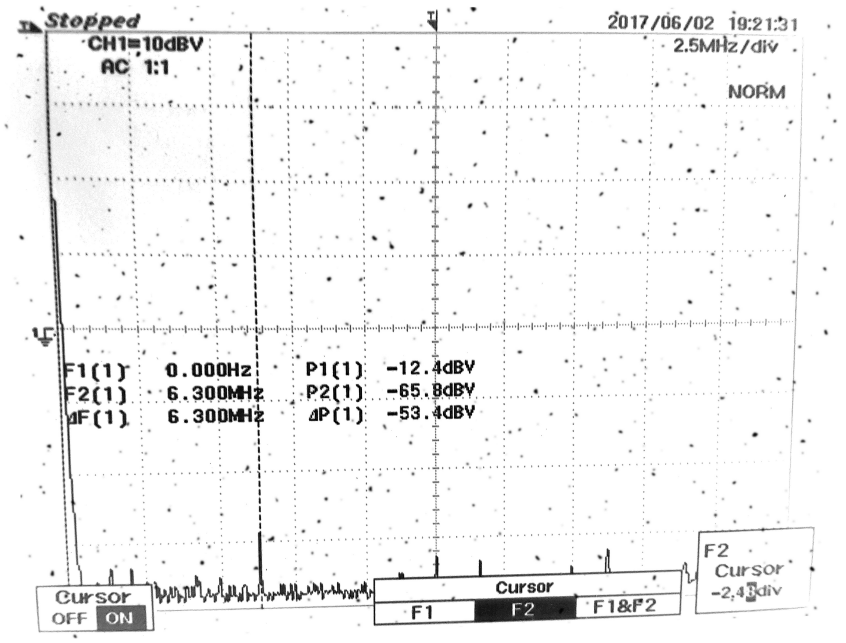


Figura 5.12: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 4 períodos en LUT

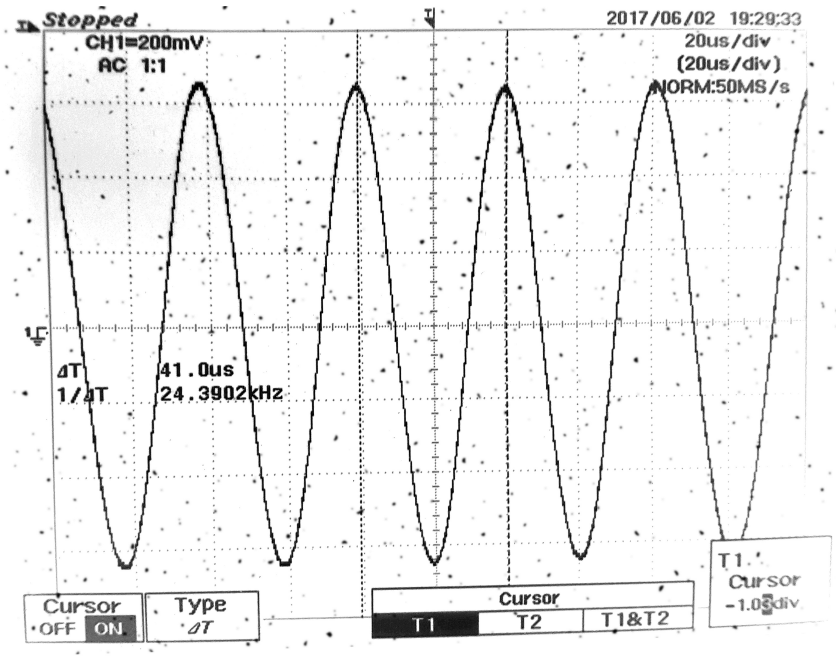


Figura 5.13: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 8 períodos en LUT

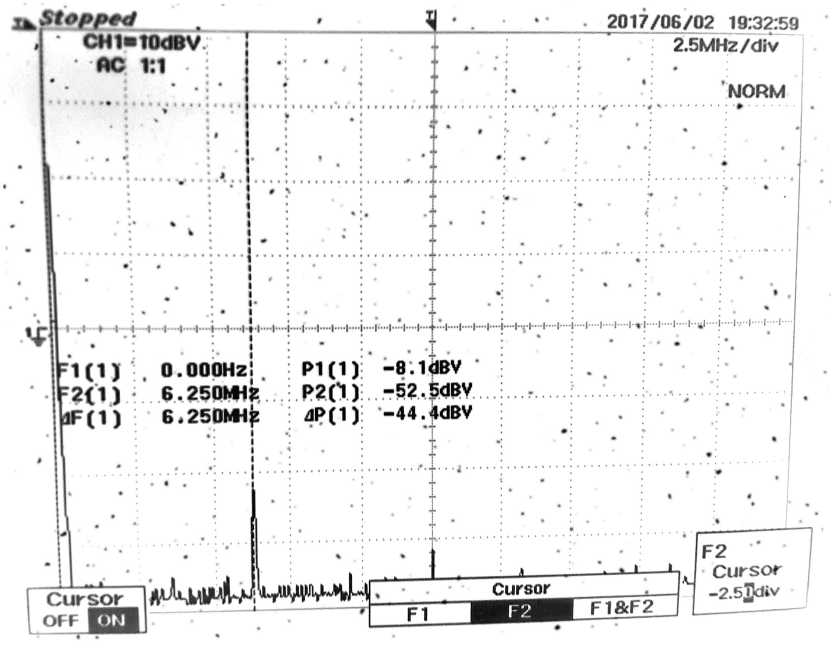


Figura 5.14: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 8 períodos en LUT

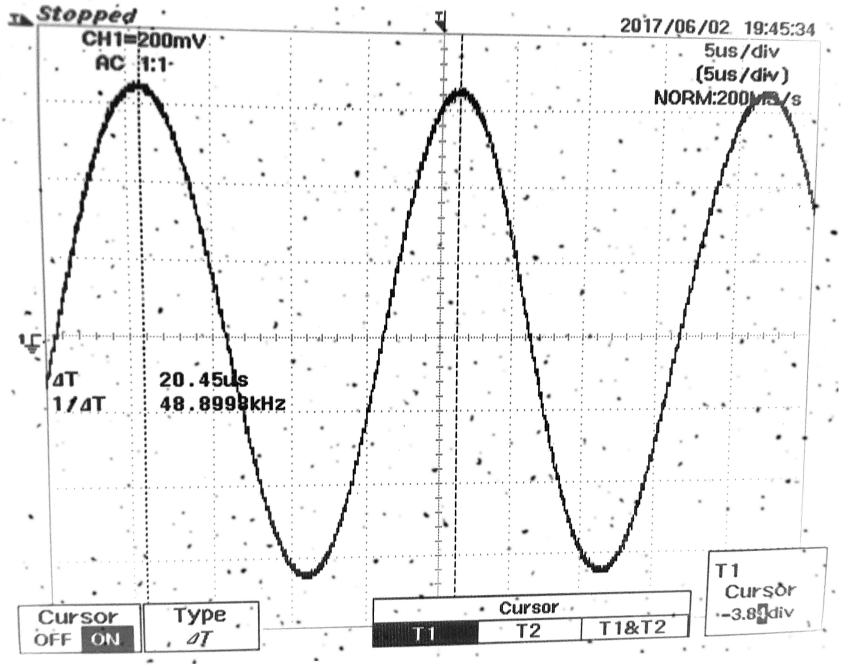


Figura 5.15: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 16 períodos en LUT

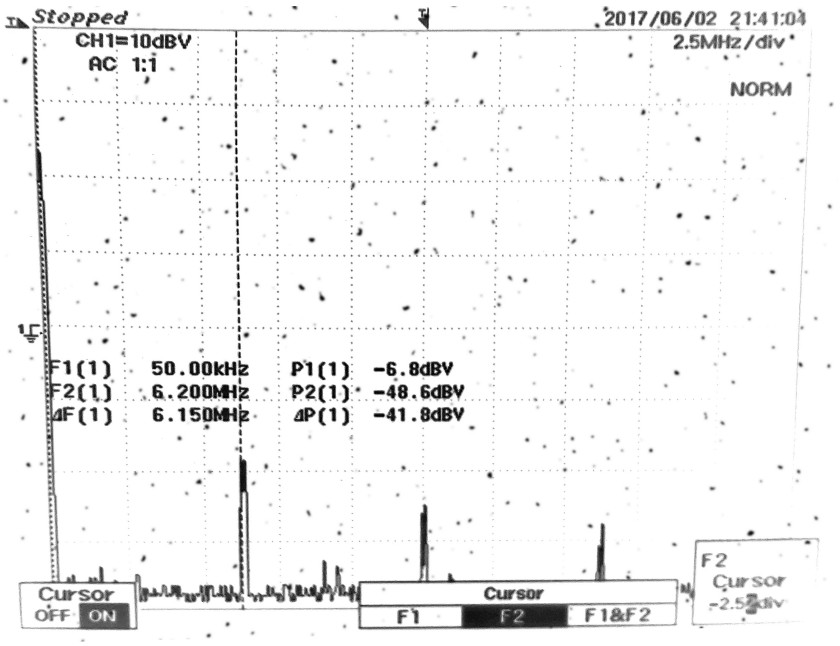


Figura 5.16: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 16 períodos en LUT

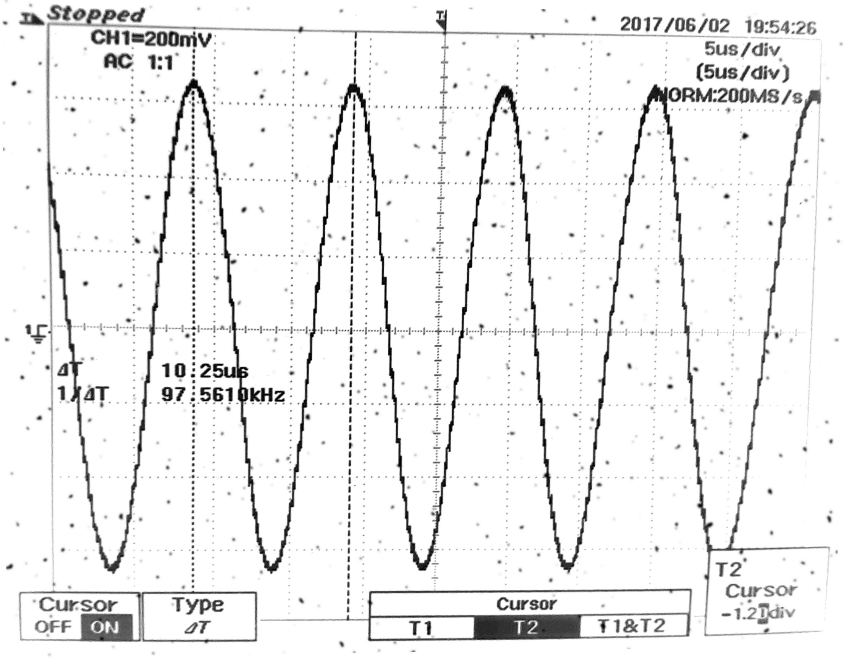


Figura 5.17: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 32 períodos en LUT

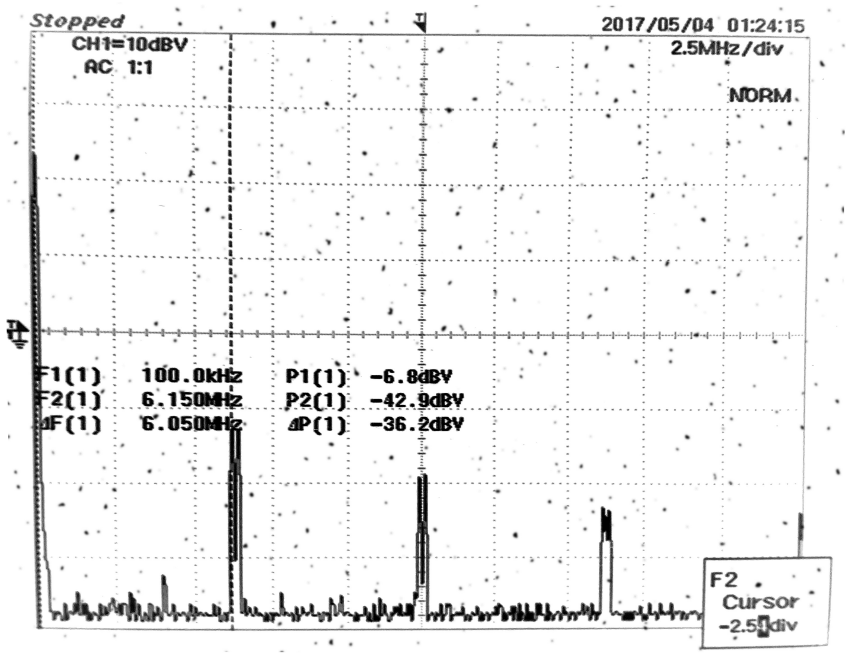


Figura 5.18: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 32 períodos en LUT

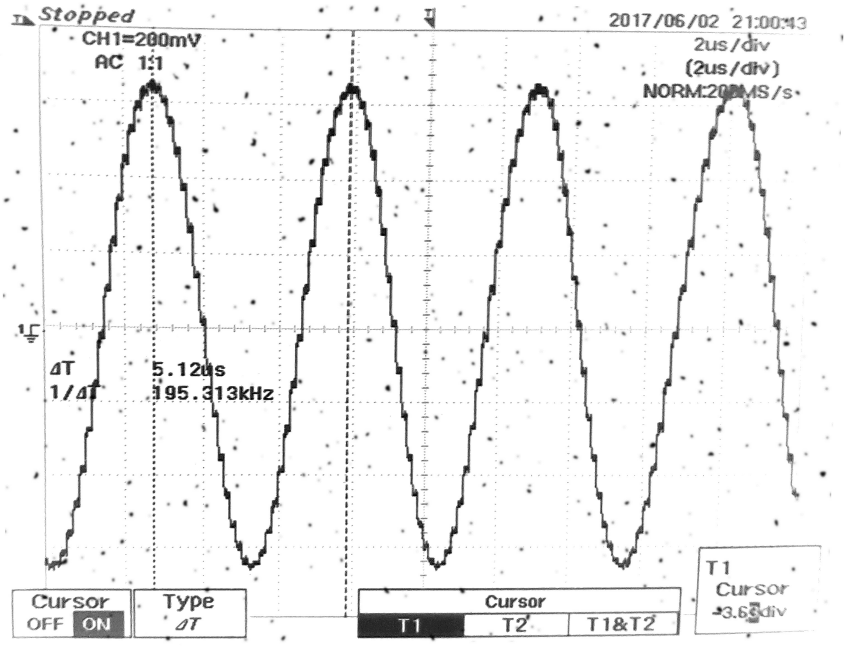


Figura 5.19: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 64 períodos en LUT

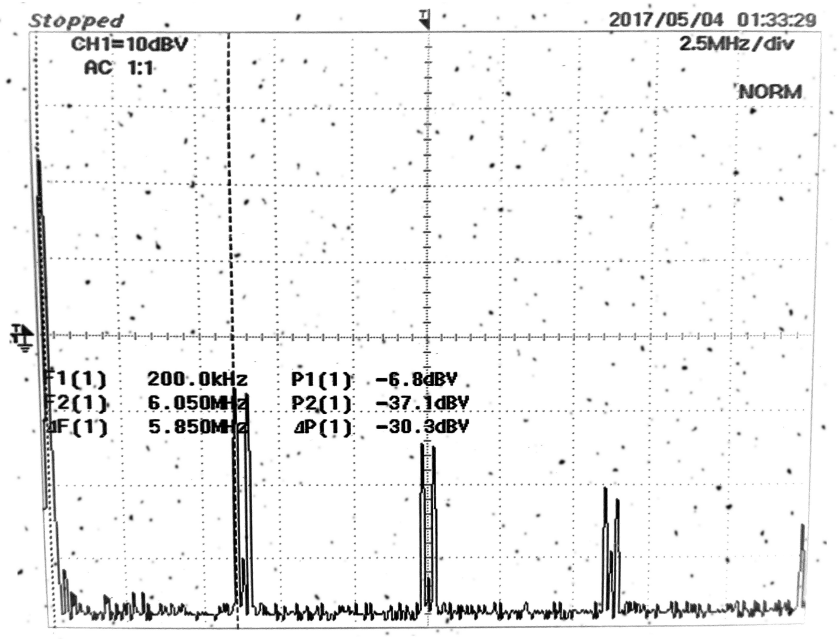


Figura 5.20: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 64 períodos en LUT

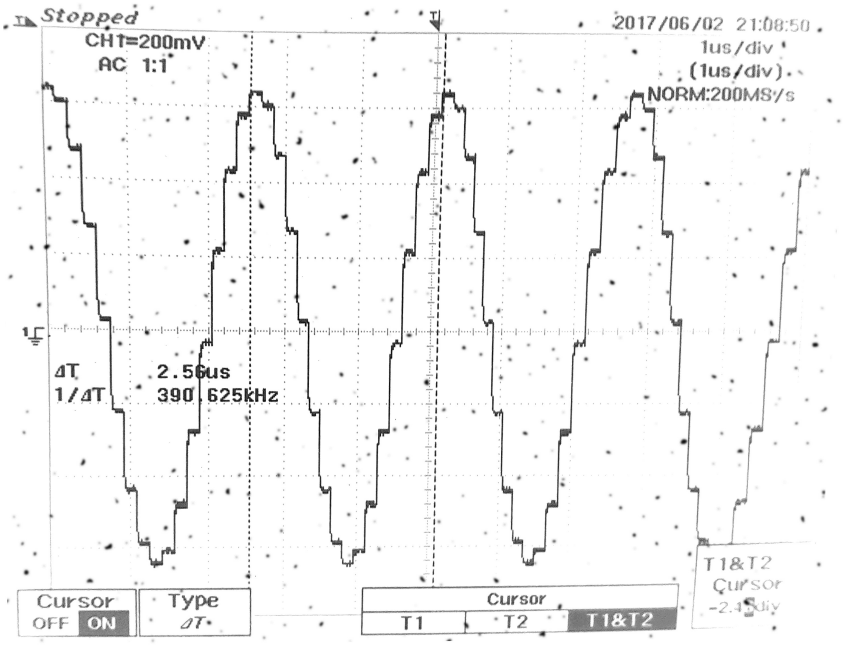


Figura 5.21: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 128 períodos en LUT

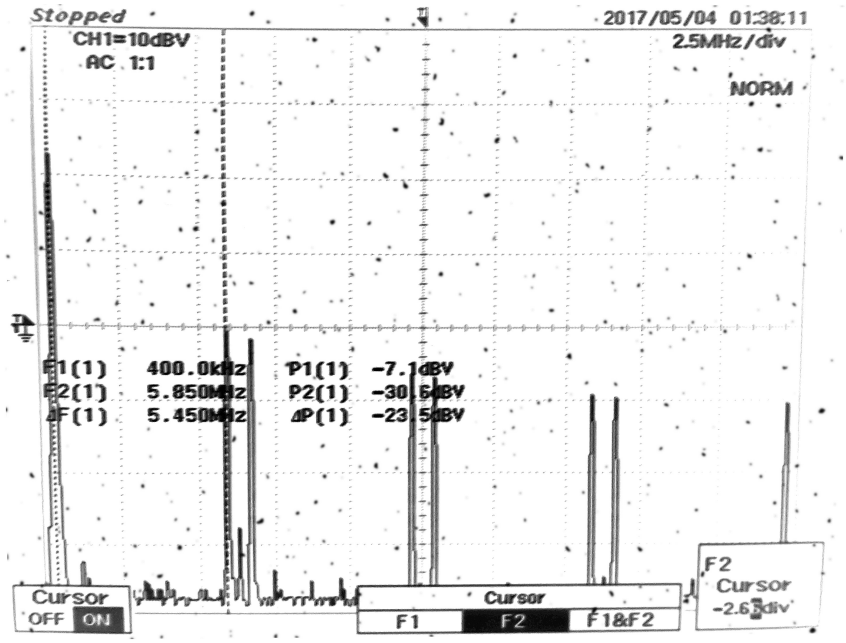


Figura 5.22: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 128 períodos en LUT

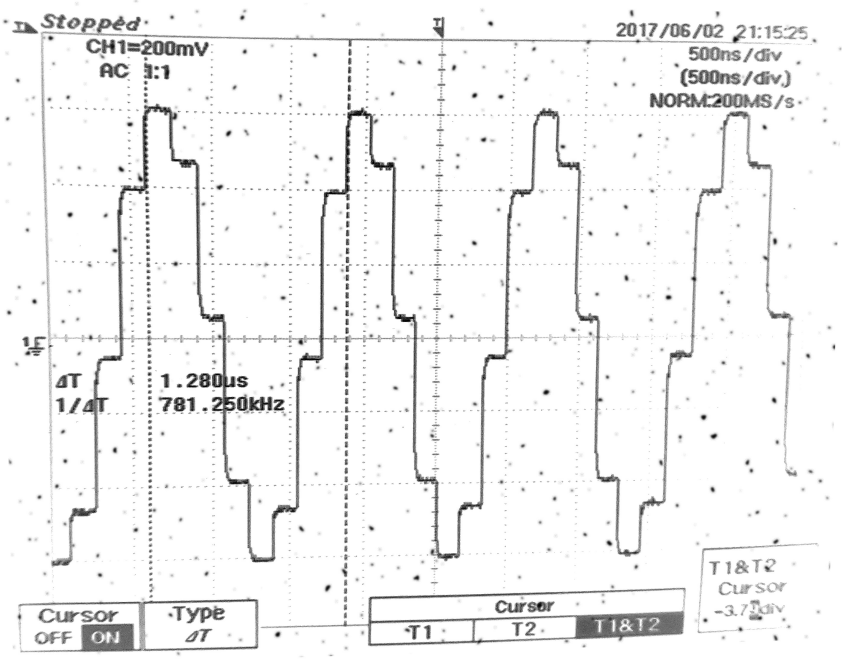


Figura 5.23: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 256 períodos en LUT

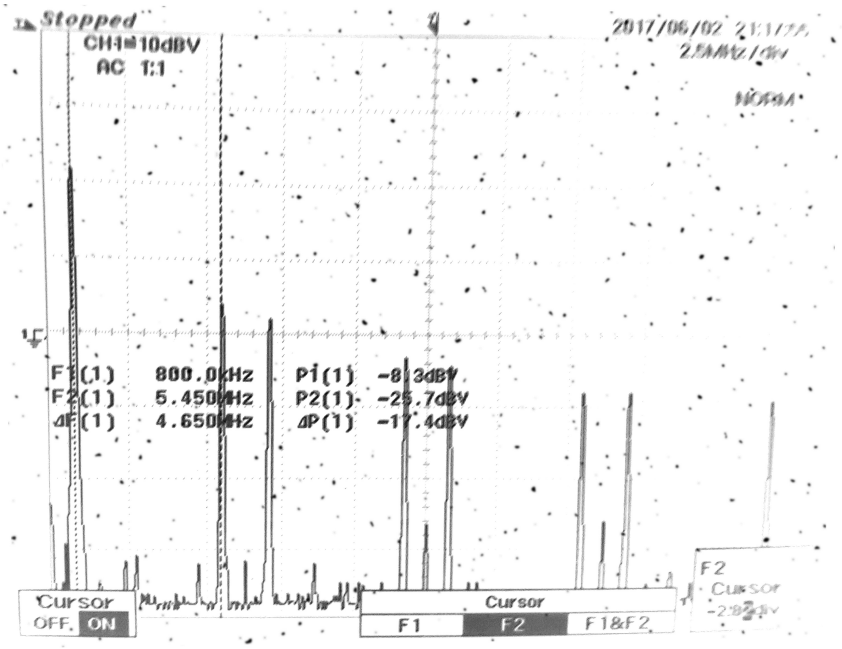


Figura 5.24: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 256 períodos en LUT

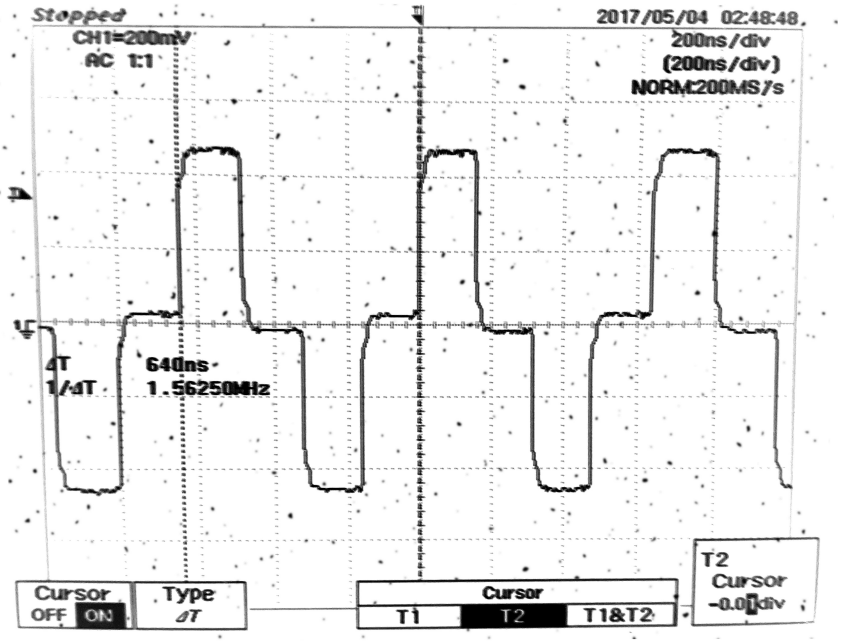


Figura 5.25: Medición de la salida temporal del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 512 períodos en LUT

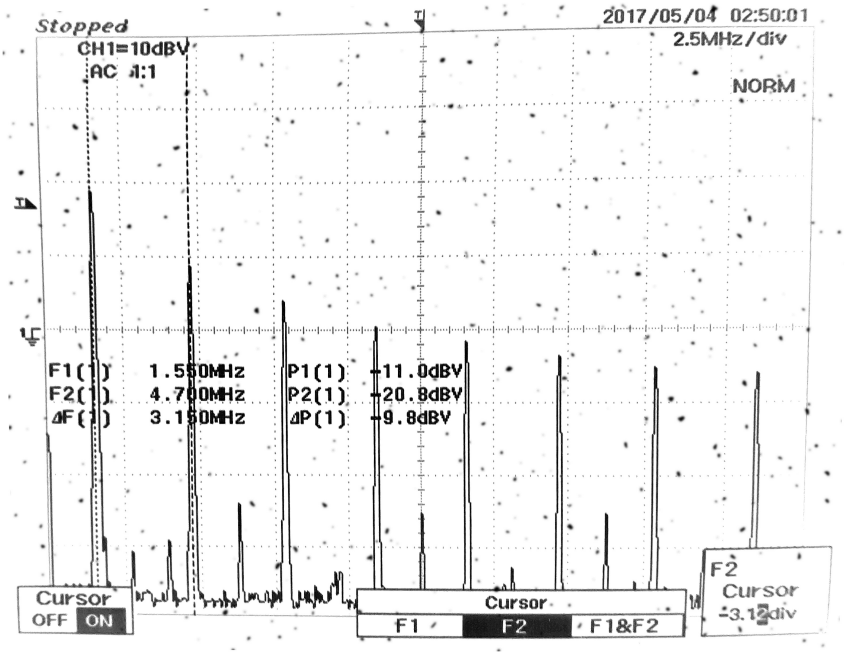


Figura 5.26: Medición de la salida en frecuencia del filtro de interpolación CIC para los parámetros R=8, M=2 y N=1 y entrada senoidal de 512 períodos en LUT

5.3. Entrada de Ruido de distribución uniforme

5.3.1. Hardware utilizado

Se procede a validar el diseño mediante entrada de ruido de distribución uniforme en una placa DE2 con una FPGA Cyclone II EP2C35F672C6 y se utiliza un analizador de espectro Signal Hound USB-SA44B para medir la respuesta en frecuencia del filtro interpolador CIC R=8, N=1 para valores M=1 y M=2 y una f_s de 50Mhz. La salida de la FPGA se toma a través de la salida VGA de la placa utilizando el DAC del color rojo. Los datos se procesan mediante el software *Spike: Signal Hound Spectrum Analyzer Software* [19].

5.3.2. Generación de ruido de distribución uniforme

El generador de ruido de distribución uniforme es implementado internamente en la FPGA utilizando un generador de Tausworthe [18] ilustrado en la Figura 4.13. El generador de ruido de Tausworthe fue implementado con 3 LFSR en paralelo con una operación XOR entre sus resultados. Para la implementación de este generador se utilizó el uso de un tipo de shifter conocido como Barrel Shifter. La particularidad de este módulo es que realiza la operación de corrimiento en un solo ciclo de reloj.

5.3.3. Mediciones

Los resultados de las mediciones se encuentran en las Figuras 5.27 y 5.28.

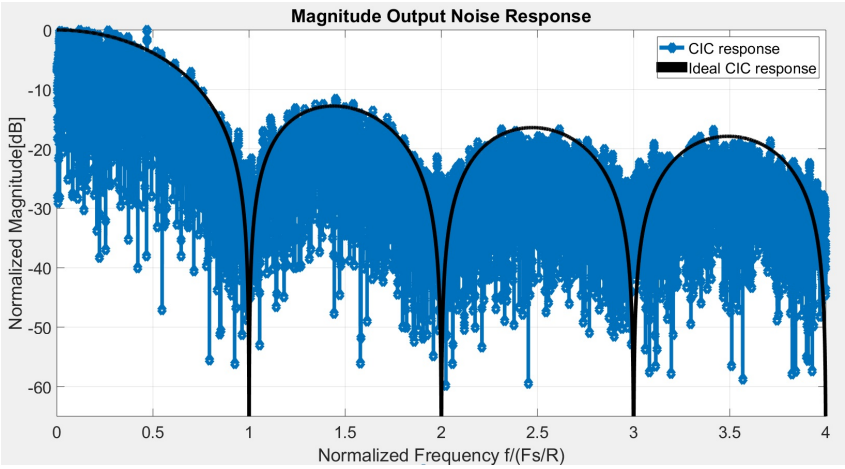


Figura 5.27: Salida del filtro interpolador CIC sintetizada para R=8, M=1 y N=1

Se observa una mejor atenuación de la señal en los nulos para entrada de ruido cuando M=2.

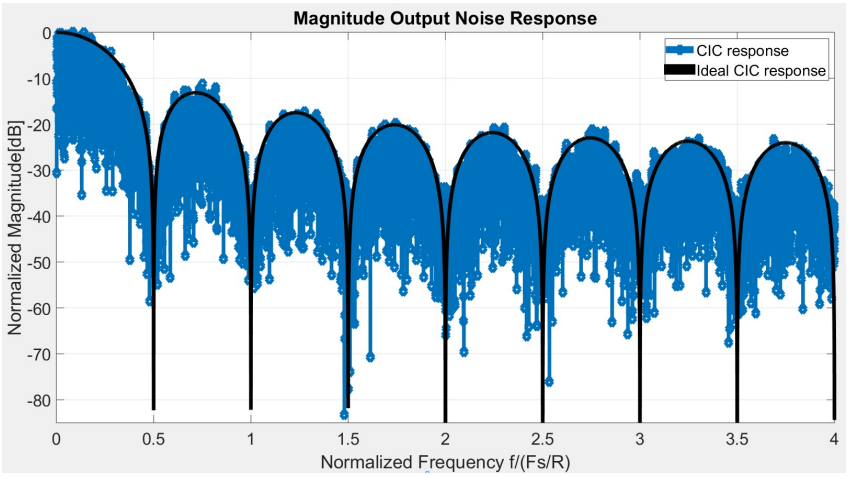


Figura 5.28: Salida del filtro interpolador CIC sintetizada para R=8, M=2 y N=1

5.4. Recursos utilizados

Para conocer el máximo timing y la utilización de recursos del diseño se procede a sintetizar un filtro de N=6, M=2 y R=8, con entrada de ruido de distribución uniforme. Se presenta el resumen de la utilización de recursos en la tabla 5.5.

Tabla 5.5: Recursos utilizados

Lógica	1 %
Celdas Lógicas	436
Registros Lógicos Dedicados	425
Registros E/S	0
Bits de Memoria	0
M4Ks	0
Pines	27
Pines Virtuales	0
LUT - Sólo celdas lógicas	11
Registros - Sólo celdas lógicas	86
LUT/Registro celdas lógicas	339

La máxima frecuencia a la que puede trabajar el circuito diseñado es de 105 MHz.

Capítulo 6

Conclusión y trabajos futuros

Se concluye finalmente que se ha logrado la construcción de un filtro digital reconfigurable capaz de realizar la interpolación y filtrado de la señal a transmitir satisfactoriamente.

Además, se ha logrado internalizar conceptos básicos de procesamiento digital de señales y comprender los fundamentos de los filtros CIC.

Mediante la metodología elegida, a lo largo del desarrollo del presente trabajo integrador se logra verificar satisfactoriamente mediante simulación que el modelo propuesto, tanto en punto flotante como punto fijo, coincide con los resultados esperados de acuerdo con su respectiva curva de respuesta de magnitud en frecuencia en concordancia a los parámetros configurados de R, M y N. Los modelos desarrollados también cuentan con una validación satisfactoria en FPGA para entradas de tren de impulsos, senoidal y de ruido de distribución uniforme, pudiendo medir correctamente los valores de atenuación pertinentes una vez implementados en la placa.

6.1. Conclusión

La economía de los filtros CIC se debe a:

1. No requiere multiplicadores
2. No se requiere almacenamiento para los coeficientes del filtro.
3. El almacenamiento intermedio es reducido al integrar a la tasa alta de muestreo y realizar el filtrado comb a la tasa baja de muestreo, comparado con la implementación de un filtro polifásico y a la implementación equivalente usando filtros uniformes FIR en cascada
4. La estructura de los filtros CIC es secilla, consistiendo en tan sólo 2 bloques básicos
5. Se requiere poco control externo

6. El mismo diseño del filtro puede ser usado para amplios rangos de factor de cambio de tasa R, con la adición de escalar el circuito y mínimos cambios en el timing del filtro

6.2. Debilidades

Sin embargo, se han encontrado las siguientes limitaciones:

1. El ancho de los registros puede tornarse grande para grandes factores de cambio de tasa R
2. La respuesta en frecuencia del filtro CIC es determinado sólo con 3 parámetros: N, R y M, resultando en un rango limitado de respuestas del filtro
3. El filtro CIC necesariamente debe ir acompañado de su respectivo filtro compesador debido a la caída en su respuesta en magnitud de frecuencia

6.3. Mejoras y trabajos futuros

Entre las mejoras y trabajos a realizar a futuro se encuentran:

1. El agregado de un control de errores y excepciones para evitar la incorrecta configuración de los parámetros del filtro interpolador CIC en el archivo *config*
2. La realización de un entorno GUI que facilite la configuración de los parámetros del filtro interpolador CIC
3. La realización del diseño e implementación del filtro compensador correspondiente de acuerdo a los parámetros seteados del filtro interpolador CIC

Bibliografía

[1] Inc. Pentek. Software radio. `http://www.pentek.com/sftradcentral/sftradcentral.cfm`, 2017. [Online; accessed 2017-04-17].

[2] C Richard Johnson Jr, William A Sethares, and Andrew G Klein. *Software receiver design: build your own digital communication system in five easy steps*. Cambridge University Press, 2011.

[3] Eugene Hogenauer. An economical class of digital filters for decimation and interpolation. *IEEE transactions on acoustics, speech, and signal processing*, 29(2):155–162, 1981.

[4] Terasic Technologies Inc. Terasic de0 nano user manual. 2013.

[5] Altera Corporation. De2 development and education board user manual. 2006.

[6] The MathWorks Inc. Matlab. `https://es.mathworks.com/products/matlab.html`, 2017. [Online; accessed 2017-03-30].

[7] The MathWorks Inc. Simulink. `https://www.mathworks.com/products/simulink.html`, 2017. [Online; accessed 2017-03-30].

[8] Intel Corporation. Dsp builder for intel fpgas. `https://www.altera.com/products/design-software/model---simulation/dsp-builder/overview.html`, 2017. [Online; accessed 2017-03-30].

[9] DSP Builder Handbook. Volume 1: Introduction to dsp builder. 2014.

[10] DSP Builder Handbook. Volume 2: Dsp builder standard blockset. 2016.

[11] Intel Corporation. Intel® quartus® prime design software overview. `https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html`, 2017. [Online; accessed 2017-03-30].

[12] Richard Lyons. Understanding cascaded integrator-comb filters. *Embed Syst Program*, 18(4):14–27, 2005.

[13] Matthew P Donadio. Cic filter introduction. In *IEEE International Symposium on Communications*, pages 1–6, 2000.

[14] Douglas L. Jones. Multirate signal procesing. 2012.

[15] Alan V. Oppenheim and Ronald W. Schafer. *Tratamiento de señales en tiempo discreto*. PEARSON EDUCACIÓN, S.A., 2011.

[16] Fredric J Harris. *Multirate signal processing for communication systems*. Prentice Hall PTR, 2004.

[17] Richard G. Lyons. *Understanding digital signal processing*. Prentice Hall, 2004.

[18] Pierre L'ecuyer. Maximally equidistributed combined tausworthe generators. *Mathematics of Computation of the American Mathematical Society*, 65(213):203–213, 1996.

[19] Signal Hound. Signal hound. `https://signalhound.com/spike/`, 2017. [Online; accessed 2017-06-10].

Anexos

Anexo A

Scripts de MATLAB

A.1. main

```
%% Path adding

close all;
clear all;
clc;

addpath(genpath('../interpolation_filter'))     %add subfolder path
addpath(genpath('../simulink_simulations'))     %add subfolder path

%% Config_Files

run('config');
% run('config_Impulse_R8_M2_N1_Floating_vs_Fixed');
% run('config_Sin_R8_M2_N1_Floating_vs_Fixed');
% run('config_Noise_R8_M2_N1_Floating_vs_Fixed');
% run('config_Noise_R8_M2_N1_Floatingsource_fixedcic');
% run('config_Impulse_R75_M1_N3_Floating_vs_Fixed');
% run('config_Sin_R75_M1_N3_Floating_vs_Fixed');
% run('config_Noise_R75_M1_N3_Floating_vs_Fixed');
% run('config_Noise_R75_M1_N3_Floatingsource_fixedcic');

%% Create Input Signal
[x]=input_config(Signal_Type,Total_samples,Impulse_Amplitude,A,f,phi,Offset,
Noise_var,Noise_mean,Noise_Add);

%% Floating Point Simulation
if strfind(Sim_Type,'floating')
    [c,interpol,y]=floating_point(x);
```

```
end

%% Fixed Point Simulation
if not(isempty(strfind(Sim_Type,'fixed')))
    % Calculate the bit width of all the stages
    W=Word_Width(R,M,stages,Input_Bits);
    % Simulink simulation
    DataOut=simulation(sim_time);
end

%% Plots
if strfind(Sim_Type,'floatingfixed')
    all_plots(x,c,interpol,y,Sim_Type,DataOut);
elseif strfind(Sim_Type,'floating')
    all_plots(x,c,interpol,y,Sim_Type);
elseif strfind(Sim_Type,'fixed')
    all_plots(Sim_Type,DataOut);
end
```

A.2. config

```
% Config file. Sets different parameters of simulation.

%% CIC Filter configuration files
R=8;                % Rate change factor
M=1;                % Differential Delay
D=M*R;              % Differential Delay at high sampling rate
stages=1;           % Filter Order
Fs=50e6;            % High sampling frequency
Ts=1/Fs;            % time interval between samples

save CIC_Parameters R M D stages Fs Ts

%% Input signal configuration
% 'Impulse' for impulse input
% 'Sin' for sin input
% 'Noise' for noise input
Signal_Type='Impulse';
Noise_Add=0;            % Boolean variable for adding noise to input signal
Noise_Source=1;         % Boolean variable for selecting the source
                        % Noise_Source=0 means the noise
                        % generator is from floating point
                        % Noise_Source=1 noise generator from
```

```
                        % fixed point.
Sin_Source=0;           % Boolean variable for selecting the source.
                                    % 0=from floating point
                        % 1=from fixed point by the LUT
Total_samples=2^14;     % Total number of simulated samples
save input_Parameters Signal_Type Noise_Add Total_samples
% Impulse Signal configuration
Impulse_Amplitude=511;            % Impulse amplitude

% Sin Signal configuration
A=15;                                     % Sin amplitude
f=(Fs/R)/16;                              % Sin frequency
phi=pi/4;                                 % Sin phase offset
Offset=0;                                 % Sin offset

% Noise Gaussian Signal configuration
Noise_var=1;                            % Noise Variance
Noise_mean=0;                           % Noise Mean

%% Simulation config
% 'floating' runs all the floating point simulations
% 'fixed'   runs all the fixed point simulations
% 'floatingfixed' runs floating and fixed point simulations
Sim_Type='floating';

% We set some parameters as Impulse Input for the simulation
Input_Bits=10;                            % Input number of bits
Fractional_Bits=0;                        % Fraccional Bits for
                                                                        %
LUT_address_width=Input_Bits;             % LUT Address Width
Integer_Bits=Input_Bits-Fractional_Bits; % Integer bits for numeric representation

Output_Bits=10;
Output_Frac_bits=0;
Output_Int_bits=Output_Bits-Output_Frac_bits;


run('Simulation_Booleans'); % Generate the simulation CIC booleans
run('Input_Selector');      % Generate the input key selector
                                                  % for input choice
run('calculo_seno');        % Generate the correct sin frequency
                                                    % for LUT integration

sim_time=Total_samples*R;      % Simulation total samples to run

```

```
%% Boolean variables for plotting

Plot_All_Signals=0;          % Plots the signals in time
Plot_Phase=0;                % Plots the phase response of the filter
Plot_tf=1;                   % Plots the zeros and poles of the tf
Plot_I_O=0;                  % Plot input vs output spectrum
Plot_Error=0;                % Plot the output spectrum comparation
                                                    % between absolute magnitude
                             % floating and fixed point simulation

save Plots_Variables Plot_All_Signals Plot_Phase Plot_tf Plot_I_O Plot_Error
```

A.3. Simulation Booleans

```
% Generate the simulation booleans for the different order of filter

if not(isempty(strfind(Sim_Type,'fixed'))) ||
   not(isempty(strfind(Signal_Type,'Noise')))
        Btwo_on=0;
        Bthree_on=0;
        Bfour_on=0;
        Bfive_on=0;
        Bsix_on=0;
    if stages>=2
        Btwo_on=1;
    end
    if stages>=3
        Bthree_on=1;
    end
    if stages>=4
        Bfour_on=1;
    end
    if stages>=5
        Bfive_on=1;
    end
    if stages>=6
        Bsix_on=1;
    end
end
```

A.4. Input Selector

```
% Create a constant for selecting the input in the simulation

if not(isempty(strfind(Sim_Type,'fixed'))) ||
   not(isempty(strfind(Signal_Type,'Noise')))
    if strcmp(Signal_Type,'Impulse')
        Input_S=0;
    elseif strcmp(Signal_Type,'Sin')
        Input_S=1;
    elseif strcmp(Signal_Type,'Noise')
        Input_S=2;
    end
end
```

A.5. Calculo Seno

```
% Generate a sin signal with the desired normalized point in simulation

if not(isempty(strfind(Sim_Type,'fixed'))) &&
   strcmp(Signal_Type,'Sin') && Sin_Source==1
    LUT_address_width=11;
    Integer_Bits=5;
    Fractional_Bits=5;
    fs_low=Fs/R;
    Ts_low=1/fs_low;
    desired_point=0.0625;
    f_sin=desired_point*fs_low;
    periods_sin=(2*pi*f_sin*((2^LUT_address_width)-1)*Ts_low)/
    (2*pi-(1/(2^LUT_address_width)));
    periods_sin_round=round(periods_sin);
    f_sin_real=periods_sin_round*(2*pi-(1/(2^LUT_address_width)))/
    (2*pi*((2^LUT_address_width)-1)*Ts_low);
    real_point=f_sin_real/fs_low;

    % We modify the config parameters
    f=f_sin_real;
    warndlg(sprintf('Se cambio la frecuencia de la onda senoidal a
    %2.3g\nque representa el punto normalizado %2.3g',f,real_point),'Aviso')
    Input_Bits=Integer_Bits+Fractional_Bits;
    Total_samples=2^LUT_address_width;
end
```

A.6. Funciones

A.6.1. Normalize frequency

```
% Normalize a frequency vector
% Inputs: f (vector), fs, R

function [freq_norm]=normalize_freq(f,fs,R)

freq_norm=1:length(f);
for i=1:length(f)
    freq_norm(i)=f(i)/(fs/R);
end
```

A.6.2. Input Config

```
% Create the input vector of the desired signal for the CIC Filter
% Then saves the input in a m file named 'input'
%
% Inputs:
%       Signal_Type=  'Impulse' for impulse input
%                     'Sin' for sin input
%                     'Noise' for noise input
%
%
%       P= The vector length of the Impulse input
%
%       A=Amplitude of the sin input
%       f=Frequency of the sin input
%       time=total time simulation for the sin input
%       phi=the phase shift of the sin input
%
%       N_A=Amplitude of the noise input
%       Noise_length=The vector length of the Noise input
%
function [x]=input_config(Signal_Type,Total_samples,Impulse_Amplitude,A,f,
                    phi,Offset,Noise_var,Noise_mean,Noise_Add)
load CIC_Parameters

if strcmp(Signal_Type,'Impulse')
    x=Impulse(Impulse_Amplitude,Total_samples,M);
elseif strcmp(Signal_Type,'Sin')
```

```
    x=Sin(Ts,A,f,Total_samples,phi,Offset);
elseif strcmp(Signal_Type,'Noise')
    x=Noise(Noise_var,Total_samples,Noise_mean);
end

if Noise_Add==true
    x=x+Noise(Noise_var,Total_samples,Noise_mean);
end

end
```

A.6.3. Impulse

```
% Create an impulse input
% Inputs:
%          Total_samples=length of input vector
%          M=the differential delay
%          Impulse_Amplitude=integer that defines the amplitude of the
%          impulse signal

function [x]=Impulse(Impulse_Amplitude,Total_samples,M)
    x=zeros(1,Total_samples);
    x(M+1)=1*Impulse_Amplitude;
end
```

A.6.4. Sin

```
% Create an sin input signal
% Inputs:
%          A=sin amplitude
%          f=sin frequency
%          time=simulation time
%          phi=sin phase
%          Offset=sin offset

function [x]=Sin(Ts,A,f,Total_samples,phi,Offset)
    load CIC_Parameters
    t=Ts*R*(0:Total_samples-1);                    % create a time vector
    x=A*sin(2*pi*f*t+phi)+Offset;
    t=0:1:(length(x)-1);
    t=t.*R;
```

```
    signal=[t; x];
    save sin signal
end
```

A.6.5. Noise

```
% Create a noise gaussian input signal or adds noise to the signal input
% Inputs:
%         Noise_var= Noise Variance
%         Noise_mean= Noise Mean

function x=Noise(Noise_var,Total_samples,Noise_mean)
   load CIC_Parameters
   x=sqrt(Noise_var)*(rand(1,Total_samples)-0.5)+Noise_mean;
   t=0:1:(length(x)-1);
   t=t.*R;
   signal=[t; x];
   save noise signal
end
```

A.6.6. Comb Filter

```
% Do a comb filter.
% Input: the input vector and the parameter M of differential delay
% Output: the vector filtered

function [output_vector]=comb_filter(M,x)
    output_vector=zeros(1,length(x));
    for i=1:M
        output_vector(i)=x(i);
    end
    for i=(M+1):length(x)
        output_vector(i)=x(i)-x(i-M);
    end
end
```

A.6.7. Integrator Filter

```
% Do a integrator filter.
% Input: the input vector
% Output: the vector filtered

function [output_vector]=integrator_filter(x)
    output_vector=1:length(x);
    output_vector(1)=x(1);
    for n=2:length(x)
        output_vector(n)=x(n)+output_vector(n-1);
    end
end
```

A.6.8. Rate Change Factor

```
% Do an interpolation.
% Input: the input vector
% Output: the vector interpolated

function [output_vector]=rate_change_factor(R,x)
    output_vector=zeros(1,length(x)*R);
    for i=1:length(x)
        output_vector(i*R)=x(i);
    end
end
```

A.6.9. Floating Point Simulation

```
%% Floating Point Simulation
% x=the input signal;
% c=the signal after the different comb filters
% interpol=the signal after the rate change factor
% y=the output signal after the different integrator filters

function [c,interpol,y]=floating_point(x)
load CIC_Parameters
%% Comb Filtering
    c=x;
    for i=1:stages
        c=comb_filter(M,c);
    end
```

```
14
15  %% Interpolation
16      interpol=rate_change_factor(R,c);
17
18  %% Integrator Filtering
19      y=interpol;
20      for i=1:stages
21          y=integrator_filter(y);
22      end
23
24  end
```

A.6.10. Word Width

```
1   % Calculate the bit word width of the CIC Interpolation Filter
2
3   function W=Word_Width(R,M,stages,Input_Bits)
4
5   G=length(2*stages);
6   W=length(2*stages);
7
8   for i=1:2*stages
9       if i<=stages            % Si es una etapa de comb
10          G(i)=2^i;
11      elseif i<=2*stages
12          G(i)=((2^(2*stages-i))*((R*M)^(i-stages)))/R;
13      end
14      W(i)=ceil(Input_Bits+log2(G(i)));
15  end
16  if M==1
17      W(stages)=Input_Bits+stages-1;
18  end
19  if length(W)==10
20      W=[Input_Bits W Input_Bits];
21  elseif length(W)==8
22      W=[Input_Bits Input_Bits W Input_Bits Input_Bits];
23  elseif length(W)==6
24      W=[Input_Bits Input_Bits Input_Bits W Input_Bits Input_Bits Input_Bits];
25  elseif length(W)==4
26      W=[Input_Bits Input_Bits Input_Bits Input_Bits W Input_Bits Input_Bits
27        Input_Bits Input_Bits];
28  elseif length(W)==2
29      W=[Input_Bits Input_Bits Input_Bits Input_Bits Input_Bits W Input_Bits
30        Input_Bits Input_Bits Input_Bits Input_Bits];
```

```
31 end
32
33 end
34
35 % save bit_width W
36
37 % Ref: An Economical Class of Digital Filters for Decimation and
38 % Interpolation,Eugene B. HOGENAUER,22-23.
```

Anexos del Proyecto Integrador

Solicitud de Aprobación de Tema

Facultad de Ciencias Exactas, Físicas y Naturales
ÁREA INGENIERÍA
ESCUELA DE ELECTRÓNICA
C.C. 755 - Correo Central - 5000 - CÓRDOBA
Tel. Directo (0351) 33-4147 int 110
Conmutador: 433-4141 y 33-4152 - Interno 10

Sr. Director de la Escuela de Ingeniería Electrónica

Ing.: Rodrigo Bruni

Me dirijo a Ud. a fin de solicitar la **aprobación del tema del Proyecto Integrador (PI)** que propongo a continuación:

Tema

Nombre del Proyecto: FILTRO INTERPOLADOR DIGITAL PARA PLATAFORMA SDR BASADA EN FPGA

Descripción: En ANEXO.

Desarrollo del prototipo: Diseño e implementación de IP core en FPGA y bancos de prueba.

Director de PI

Nombre: Graciela Corral Briones

Cargo: Profesor Asociado - Investigador Adjunto Conicet

Dirección Personal o Laboral: Campiña del Sur Nro 68

TE: (351) 153 720341

eMail: graciela.corral@unc.edu.ar

Firma del Director: .....

Co-Director de PI

Nombre: Juan Martin Ayarde

Cargo: Profesor asistente Electrónica Digital 3

Dirección Personal o Laboral: Velez Sarsfield 1611

TE: 351 5353800 (int29085)

eMail: martin.ayarde@unc.edu.ar

Firma del Co-Director: .....

Datos del Estudiante

Nombre y Apellido: Ariane Zain RAFAEL ALMEDRA

Matrícula: 34601285

Materias que faltan aprobar: 0

Dirección: Nicolás Avellaneda 1963

Localidad: Córdoba.

Provincia: Córdoba.

e-mail: ariane.rafael@alumnos.unc.edu.ar

Teléfono: 0351-152428234

Firma:.....

Objetivo: (*Indicar los motivos por los cuales se desarrolla este tema: Inexistencia en el medio, motivos económicos, requerimientos de terceros, etc*)

El presente trabajo integrador se enmarca dentro de un proyecto de diseño de sistemas de comunicaciones con múltiples antenas que cuenta con financiado de la SeCyT-UNC. El módulo en particular que se desarrollará es un filtro digital que realiza la interpolación y filtrado de la señal a transmitir. La arquitectura del filtro debe permitir la configuración del ancho de banda y cambio de tasas de muestreo en un amplio rango de valores.

Antecedentes de Proyectos similares: *(Indicar la existencia de otro PI similar desarrollado dentro del área de la Facultad y marcar las diferencias con el presente)*

El presente trabajo final es nuevo en el área de aplicación de comunicaciones digitales.

Duración y Fases de las tareas previstas: En ANEXO.

Metodología:

Lugar previsto de realización: Laboratorio de Comunicaciones Digitales – FCEFyN - UNC

Requerimiento de Instrumental y equipos: Osciloscopio, placa FPGA, generador de espectro, adaptador VGA/BNC.

Inversión estimativa prevista por el alumno: Ninguna.

Apoyo Económico externo a la Facultad: Ninguno.

Referencias Bibliográficas o de Software:

[1] MATLAB, © 1994-2017 The MathWorks, Inc., MATLAB –Mathworks, https://www.mathworks.com/products/matlab.html.

[2] DSP Builder, © Intel Corporation, DSP Builder, https://www.altera.com/support/suppor resources/intellectual-property/dsp/dsp-builder/ips-dsp-builder.html.

[3] Intel® Quartus® Prime Design Software Overview, © Intel Corporation, Quartus Prime – Overview, https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html.

[4] Hogenauer, E. B. (Abril 1981). An Economical Class of Digital Filters for Decimation and Interpolation. IEEE Transactions on acoustics, speech, and signal processing, ASSP-29(2), 155-162.

[5] Richard Johnson, C. Jr., Sethares, W. A. y Klein, A. G., (2011), Software Receiver Design. Build your own digital communication system in five easy steps, Nueva York, Estados Unidos: Cambridge University Press.

[6] Lyons, R. G., (2004), Understanding digital signal processing, Nueva Jersey, Estados Unidos: Prentice Hall Professional Technical Reference

Recibido Cátedra PI

.....

Firma

Córdoba, / / .

ANEXO

(Informe, con no mas de 4 hojas, que explique el proyecto, diagramas en bloques, funcionamiento esperado, alcances, etc.)

1. Descripción Detallada del Proyecto

El presente proyecto de investigación forma parte de un sistema de transmisor digital **SDR** (Software Defined Radio). Radio definida por software o SDR es un sistema de radiocomunicaciones donde varios de los componentes típicamente implementados en hardware (mezcladores, filtros, moduladores/demoduladores, detectores, etc) son implementados en software. Esto facilita la reconfiguración de protocolos, formas de onda y procesamiento de señales en general, permitiendo la reutilización del código en diferentes hardware.

El filtro físicamente será implementado como un IP core en una FPGA *(Field Programmable Gate Array)*. El dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser reconfigurada mediante software. Para la programación de la FPGA se utiliza el programa DSP Builder, que se encarga de generar el código VHDL que luego se escribe en la FPGA mediante el programa Quartus Prime.

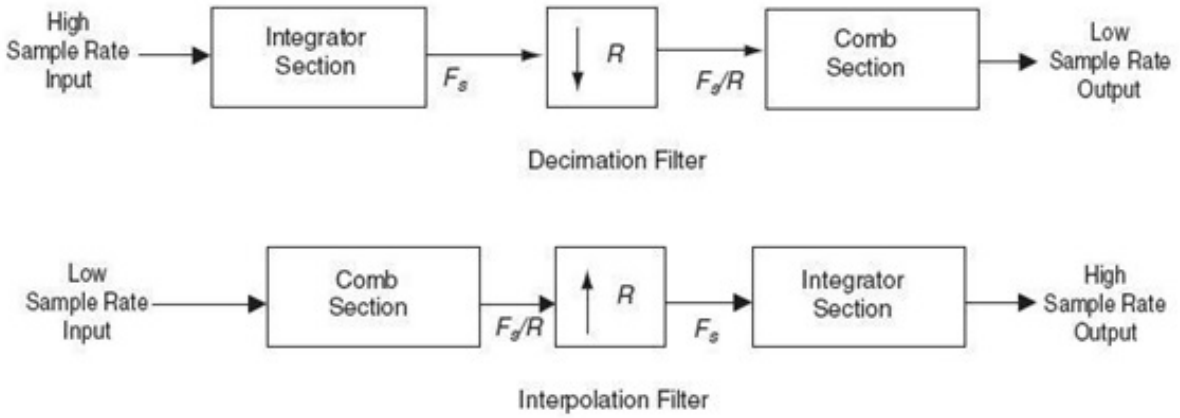


Figura A.1: Filtro CIC Decimador e Interpolador

Los filtros CIC (Cascaded integrator-comb) son implementaciones computacionalmente eficientes de filtros pasabajos y están frecuentemente embebidos en los sistemas de comunicaciones modernos. Los filtros CIC son muy adecuados para el filtrado anti-aliasing antes de la decimación y para el filtrado anti-imaging de señales interpoladas. Ambas aplicaciones están asociadas con el filtrado a muy altas tasas de datos.

La función esencial de un filtro decimador o interpolador es aumentar a disminuir la tasa de muestreo y atenuar el aliasing o las imágenes.

Los filtros CIC son implementado mediantes filtros integradores y peines (comb) en cascada (CIC) lo cual simplifica bastante el hardware requerido. La parte integradora opera en una tasa de muestreo alta y la parte comb a una tasa de muestreo baja.

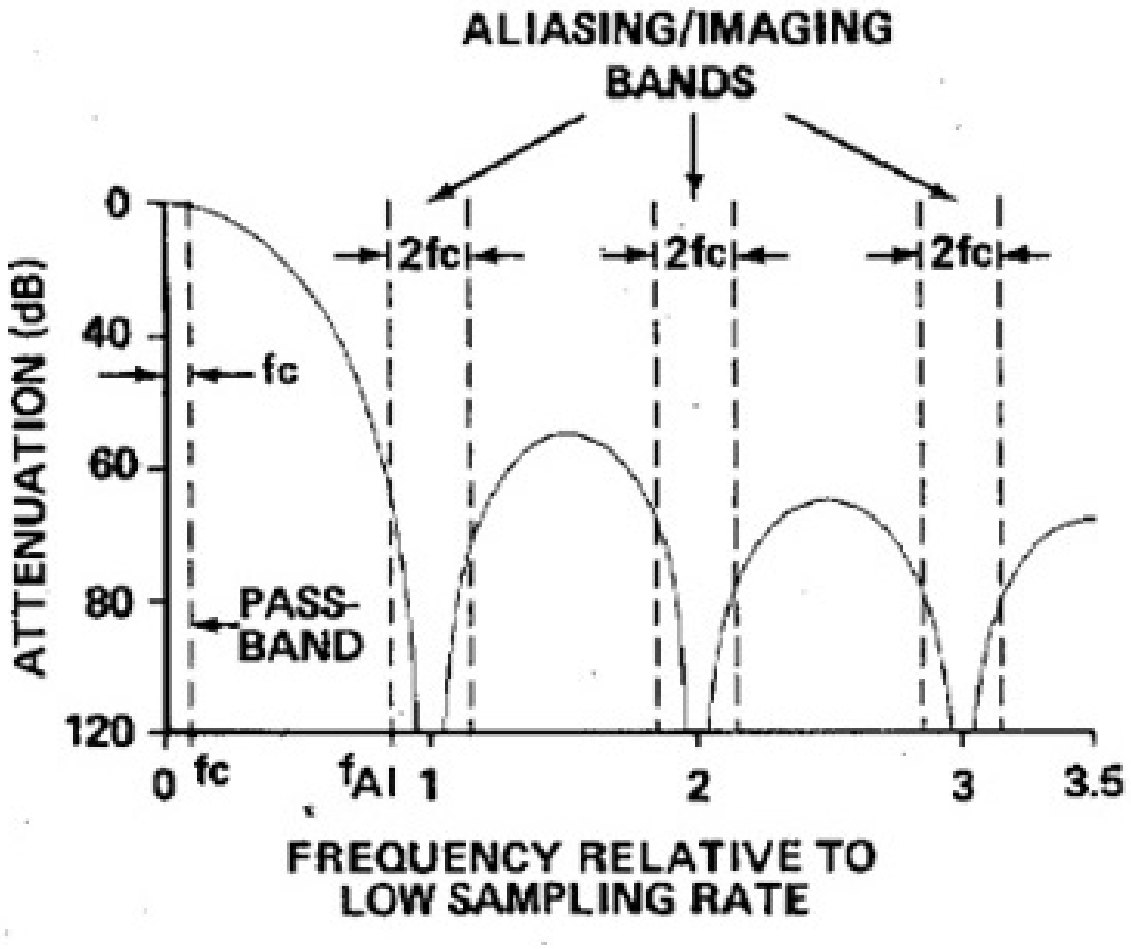


Figura A.2: Ejemplo de la respuesta en frecuencia de un filtro CIC con los parámetros N=4, M=1, R=7 y $f_c = \frac{1}{8}$

Se pretende llegar a un diseño modular y parametrizable tanto en punto flotante como en punto fijo que satisfaga los niveles de atenuación reportadas en el paper paper seminal [4]. Para ello el diseño en punto flotante debe presentar los valores de la tabla I y II de dicho paper.

Una vez verificado los resultados de la implementación con los del paper [4], se procederá al diseño en punto fijo, corroborando que el error no sea significativo con respecto al diseño en punto flotante.

Las pruebas a realizar serán:

- Test con entrada impulsiva
- Test con entrada senoidal
- Test con entrada de ruido uniforme
- Test multietapa de entrada impulsiva
- Test de BER en sistema de transmisión (en la medida de lo posible)

Se espera poder apreciar la atenuación deseada tanto en la banda de paso como en las bandas de imagen. Para ello se utilizará un generador de espectro y se comparará con los resultados de la simulación del diseño en punto fijo.

Una vez obtenidos los resultados esperados se procederá a modificar el diseño para incluir hasta 6 etapas del filtro y un cambio de tasa variable. De esta forma se repetirán las prueba corroborando los valores de atenuación para un filtro multietapa.

2. Duración y fases de las tareas previstas

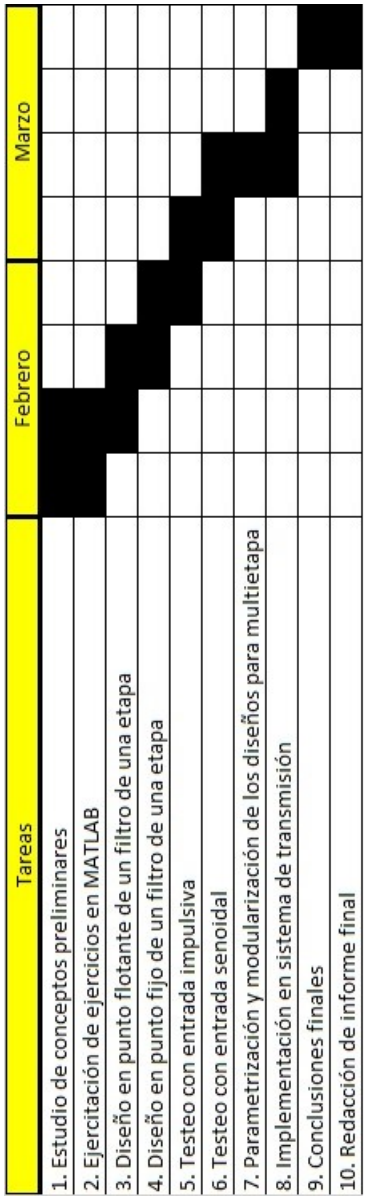


Figura A.3: Diagrama de Gannt

3. Referencias Bibliográficas y de Software

[1] MATLAB, © 1994-2017 The MathWorks, Inc., MATLAB –Mathworks, https://www.mathworks.com/products/matlab.html.

[2] DSP Builder, © Intel Corporation, DSP Builder, https://www.altera.com/support/suppor resources/intellectual-property/dsp/dsp-builder/ips-dsp-builder.html.

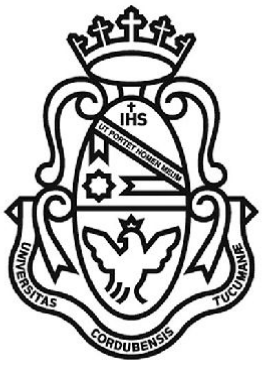
[3] Intel® Quartus® Prime Design Software Overview, © Intel Corporation, Quartus Prime – Overview, https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html.

[4] Hogenauer, E. B. (Abril 1981). An Economical Class of Digital Filters for Decimation and Interpolation. IEEE Transactions on acoustics, speech, and signal processing, ASSP-29(2), 155-162.

[5] Richard Johnson, C. Jr., Sethares, W. A. y Klein, A. G., (2011), Software Receiver Design. Build your own digital communication system in five easy steps, Nueva York, Estados Unidos: Cambridge University Press.

[6] Lyons, R. G., (2004), Understanding digital signal processing, Nueva Jersey, Estados Unidos: Prentice Hall Professional Technical Reference.

Nota de Aprobación Final del Director



UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales
Escuela de Ingeniería Electrónica

Quien suscribe la Profesora **Dra. Ing. Graciela CORRAL BRIONES** en su carácter de Directora del Proyecto Integrador de la Estudiante **Ariane Zain RAFAEL ALMENDRA** denominado: **FILTRO INTERPOLADOR DIGITAL PARA PLATAFORMA SDR BASADA EN FPGA** considera que el desarrollo del trabajo se ha completado según lo especificado en la Solicitud de Aprobación de Tema y se encuentra en condiciones de tramitar su defensa.

A los efectos de quién corresponda, en fecha 10/07/2017

Firma y aclaración del Director