**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
MAESTRÍA EN INGENIERÍA ELÉCTRICA – TELECOMUNICACIONES

# CODIFICACIÓN DE SEÑALES DE AUDIO EN TIEMPO REAL UTILIZANDO MICROPROCESADORES DE LA FAMILIA TMS320C6713

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
**ING. ADÁN BONILLA CHÁVEZ**

TUTOR
DR. BOHUMIL PSÉNICKA
FACULTAD DE INGENIERÍA

MÉXICO, D. F., OCTUBRE DE 2013

**Jurado asignado:**

| | |
|---|---|
| Presidente: | Dr. Francisco García Ugalde. |
| Secretario: | M.I. Larry Escobar Salguero. |
| $1^{er}$ Vocal: | Dr. Bohumil Psenicka. |
| $1^{er}$ Suplente: | Dr. Víctor Rangel Licea. |
| $2^{o}$ Suplente: | Dra. Lucía Medina Gómez. |

Lugar donde se realizó la tesis:

Laboratorio de Procesamiento Digital de Señales del
Depto. de Telecomunicaciones de la Facultad de Ingeniería
de la Universidad Nacional Autónoma de México (UNAM).

Tutor de tesis:

_____

**Dr. Bohumil Psenicka.**

Este trabajo se desarrolló con instrumental y equipo del Laboratorio de Procesamiento Digital de Señales del Departamento de Telecomunicaciones de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM) bajo la tutoria del doctor Bohumil Psenicka.

El desarrollo de los estudios de Posgrado que derivaron esta tesis se realizaron con el apoyo de una Beca de la Coordinación de Estudios de Posgrado de la UNAM, a la cual agradezco de manera especial.

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM, Proyecto IN114012: Desarrollo de banco de filtros para su aplicación en procesamiento digital de señáles orientado a telecomunicaciones. Agradezco a la DGAPA-UNAM la beca recibida.

Redacción y edición de tesis
con LaTeX 2ε
GNU/LINUX.

A Antonia, Adán, Arce, Adolfo y Mizra:

Mi familia, mi inspiración y mi fuente motora para seguir adelante... Sin ellos nada, con ellos todo

Al Dr Psenicka:

Mi tutor, por haber confiado en este loco para la realización de este proyecto

A Alicia:

Mi gran amiga: Por ser compañera indiscutible en este proceso de ser Maestro

A Amalia y José Antonio:

Confidentes, Compañeros, Amigos. Su inspiración me han ayudado a soñar que puedo caminar aún mas lejos.

♡

Confía en el tiempo, que suele dar dulces salidas a muchas amargas dificultades.

*Miguel de Cervantes Saavedra*

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: La Voluntad

*Albert Einstein*

No hay mayor señal de Ignorancia que creer imposible lo inexplicable

*Anónimo*

# Agradecimientos

Siempre he pensado que por mas planes que uno haga para su vida, la vida misma es un proceso tan estocástico que no es posible controlar todos los factores para que las metas sean exactamente las que uno se propuso. Sin embargo, no creo que tengamos un "destino" predeterminado, ni tampoco creo ciegamente en la suerte. Aun cuando no tenemos una "bolita mágica" que nos indique cual será nuestro designio, pienso que nuestras acciones son determinantes para conseguir un fin dado.

Decía Almodovar en su película "Todo sobre mi Madre" que "Uno es mas auténtico mientras mas se parece a lo que soñó sobre uno mismo". Es justamente este espacio en el que deseo expresar mi total gratitud y agradecimiento a todas aquellas personas e instancias que me han permitido y brindado la oportunidad de parecerme en gran medida a lo que soñé desde un principio.

En primer lugar, y por sobre todas las cosas, quiero agradecer a mis dos grandes profesores que me han enseñado mas que cualquier otro: Mi Padre Adán y mi Madre Antonia. Gracias por su apoyo, su amor, sus desvelos, sus regaños, sus motivaciones, sus esfuerzos pero sobre todo gracias por enseñarme a ser quien soy, siguiendo sus consejos pero al mismo tiempo aprendiendo por mi mismo de mis aciertos y mis errores. Con ellos todo, sin ellos nada. Los amo infinitamente y nunca dejaré de agradecerles su valiosa ayuda para realizar mis metas y mis sueños. Ellos son en definitiva mi mas grande impulso y mis dos enormes pilares en la vida. A ellos les debo todo y no hay forma ni manera en que pueda retribuirles, sin embargo sirva este trabajo como un pequeño testimonio de la enorme aportación que han hecho a este mundo loco. No se si sean la mejor famillia del mundo, pero de lo que si estoy seguro que son la mejor familia de MI mundo. Adan, Antonia: ni un millón de "Te amo" son suficientes para expresar todo lo que siento por ustedes.

No menos importantes son mis compañeros de toda la vida: Arcelia y Adolfo, dos grandes hermanos que han sido mi fuente de inspiración para continuar adelante. Gracias a su impulso, a su paciencia, a su fraternidad y su invaluable apoyo. En verdad me siento plenamente orgulloso y satisfecho de ser el compañero de tantas batallas que hemos pasado juntos. Me siento mas que afortunado de contar con dos grandes compañeros de vida. Con ustedes he ganado mil batallas y estoy seguro que cualquier guerra, por muy fuerte que esta sea, la ganaremos siempre juntos, jamás divididos. Los amo profundamente.

Debo agradecer a Conchita, que sin ser Ingeniera, es sin duda una de mis colegas mas admiradas. Y digo "colega" porque ella me inspiró a dedicar aunque sea un poquito de mi tiempo a la valiosa tarea de ser docente, profesión que ambos compartimos de manera orgullosa y satisfactoria. Tía, te agradezco por estar siempre al pendiente de estos tus sobrinos. Aunque no eres nuestra madre, estoy plenamente convencido que si algún día ella hubiera faltado e incluso, tocando madera, ella llegara a faltarnos aún en estos momentos, tu serías la primera en brindarnos tu apoyo incondicional. Por ser mi primera maestra, por ser mas que mi tía una gran amiga, confidente y compañera, te agradezco infinitamente.

Muchas han sido las instituciones que han participado para que yo pueda estar en lugar que actualmente ocupo, pero ninguna me ha dado tantas satisfacciones ni tanto orgullo como mi alma mater: Mi amada Universidad. Quiero agradecer profunda y enormemente a la Universidad Nacional Autónoma de México por brindarme muchas de las herramientas y armas con las que actualmente cuento para poder defenderme ante el futuro. A todos aquellos que a diario trabajan y se esmeran

por hacer de esta entidad la máxima casa de estudios de nuestro país mi mas grande admiración y respeto.

Sin menospreciar a nadie, debo justamente hacer una mención honorífica al Dr. Bohumil Psenicka por su valiosa tutoría y ayuda para la culminación de este trabajo. Su gran apoyo fue decisivo para poder seguir adelante en esta tarea. Agradezco su paciencia, sus enseñanzas, sus valiosas aportaciones pero sobre todo su enorme impulso para poder culminar con éxito mi maestría. Mas que mi profesor, es un gran amigo y como tal confió en este loco para realizar este trabajo. Muchos han sido los profesores que merecen mi gratitud y mi respeto, pero sin duda el Dr. Bohumil merece el máximo reconocimiento: no se que me depara el destino, pero sin duda uno de mis sueños es algún día tener cuando menos la mitad de la inteligencia, el tesón, la sagacidad, la paciencia y la habilidad para transmitir conocimiento que tiene este gran profesor.

No puedo dejar de agradecer a la familia Estrada García por ser grandes amigos en los momentos difíciles de mi vida. Aunque son pocos los años que llevo de conocerlos, han sido un gran ángel que se han atravesado en el camino. En especial quiero agradecer a mi niña Alicia Estrada, quién se ha ganado con creces el título de compañera, confidente y amiga. A veces, cuando piensas que todo está perdido, cuando incluso sientes que la vida no tiene sentido, siempre hay algo que te indica todo lo contrario. Alicia en especial me enseñó que "Aún cuando La Luna no se asome, no quiere decir que no esté ahí". Gracias en verdad por todo.

No olvido tampoco a mis compañeros y profesores de la Facultad de Ingeniería por hacer la estadía en este monstruo de 1000 cabezas algo mas habitable. Agradezco en especial a todos y cada uno de los profesores del Departamento de Telecomunicaciones por sus invaluables enseñanzas. También agradezco a mis compañeros por su fraternidad y por los momentos agridulces que permitieron hacerme cada día mas fuerte ante la adversidad. Mi mención especial a dos angelitos que nunca me retiraron su apoyo incondicional en esta entidad: Amalia y Jose Antonio. Gracias a todos ustedes.

Debo también agradecer a la Dirección General de Asuntos del Personal Académico de la UNAM, en especial por su valiosa ayuda para la conclusión de esta tesis mediante el proyecto DGAPA-PAPIIT IN114012. Sin su valiosa ayuda en el terreno económico, académico e incluso moral no hubiera podido concluir satisfactoriamente este trabajo.

Finalmente, y sin menor importancia, quiero agradecer a esta hermosa tierra llamada "México". Muchas veces nos quejamos de las injusticias que suceden aquí e incluso nos lamentamos de no haber nacido en otro rincón, pero definitivamente no es mi caso. Aunque con muchos sinsabores, muchos corajes, muchos desalientos y muchas decepciones, me siento orgulloso de haber nacido aquí, de ser proveniente de esa casta de guerreros que no nos cansamos nunca de luchar por que esta tierra algún día despierte de su enorme letargo. Estoy consciente que han sido muchos los que han dado su sangre y su sudor para que yo pueda estar en la posición que actualmente ocupo. A ellos me debo y a ellos les dedico este trabajo.

No se si existas, pero en cada flor, en cada destello de luz, en cada olor, en cada sabor, en cada satisfacción, en cada risa, en cada latir de mi corazón y, por qué no, en cada lágrima y en cada decepción, cada vez me das testimonio de que hay una enorme fuerza mas allá de nuestra comprensión que permite que el órden de este mundo siga tal cual lo conocemos. No se que seas ni a que te debas, pero sea lo que seas te agradezco a ti Dios por brindarme la oportunidad de nacer y disfrutar de este hermoso milagro llamado vida (y de todas sus manifestaciones, materiales y no materiales, corporeas y no corporeas, que la palabra VIDA implica). Gracias por hacerme tan afortunado.

¡Muchas Gracias a todos! ¡México, Pumas, Universidad!

# Índice general

**Agradecimientos** ix

**Abstract** xvii

**Abstract** xix

**1. Introducción** 1
- 1.1. Objetivos Generales . . . 2
- 1.2. Marco Teórico . . . 2
  - 1.2.1. Justificación . . . 3
  - 1.2.2. Metodología . . . 3
  - 1.2.3. Objetivo de la Tesis . . . 4
  - 1.2.4. Contribución del Presente Trabajo . . . 5

**2. Antecedentes y Estado del Arte** 7
- 2.1. La Voz como Fenómeno Físico . . . 7
- 2.2. La Transformada Z y su utilidad en el Procesamiento Digital de Señales . . . 8
- 2.3. Estado del Arte . . . 9
  - 2.3.1. Técnicas empleadas para la Codificación de Señales de Voz . . . 9
    - 2.3.1.1. Transformada de Fourier . . . 9
  - 2.3.2. Importancia de la Matríz de Pascal en el Procesamiento Digital de Señales . . . 10
  - 2.3.3. Procesadores Digitales de Señales . . . 11
    - 2.3.3.1. ¿Qué es un DSP? . . . 11

**3. Transformación Bilineal como Método Base para la Obtención de funciones de transferencia en el Dominio de Z** 13
- 3.1. La Transformaciòn Bilineal: Marco Teórico . . . 13
- 3.2. Implementación de la Transformada Bilineal en MATLAB para la construcción de un filtro IIR . . . 15

**4. Introducción a los Sistemas Multitasa** 17
- 4.1. Reducción de la Tasa de Muestreo . . . 17
  - 4.1.1. Submuestreo . . . 17
- 4.2. Aumento de la Tasa de Muestreo . . . 18
  - 4.2.1. Sobremuestreo . . . 18
- 4.3. Equivalencias . . . 21
  - 4.3.1. Cascada de Sobremuestreador y Submuestreador . . . 21
  - 4.3.2. Identidades Nobles . . . 21
- 4.4. Requerimientos Computacionales . . . 21
- 4.5. Descomposición Polifase . . . 22
  - 4.5.1. Proceso de descomposición . . . 22
  - 4.5.2. Estructuras del decimador e interpolador eficientes computacionalmente . . . 23

**5. Transformada Z mediante el Algoritmo de la Matriz de Pascal y sus aplicaciones en el diseño de Filtros IIR** **25**
5.1. La Matríz de Pascal y su Aplicación en la obtención de la función de transferencia en el dominio de Z . . . 25
5.2. Transformación Pasobajas-Pasobajas . . . 26
5.2.1. Cálculo Sistemático de los valores de la matríz $P_{LP}^{n}$ . . . 27
5.3. Transformación Pasobajas-Pasoaltas . . . 28
5.3.1. Cálculo Sistemático de los valores de la matríz $P_{HP}^{n}$ . . . 28
5.4. Transformación Pasobajas-Pasobanda . . . 29

**6. Uso de la Matríz de Pascal para la Codificación de Señales de Voz** **33**
6.1. Algoritmo de la Matriz de Pascal para su aplicación en Condificación de Señales de Voz . . . 33
6.2. Uso de Simulink de MATLAB como herramienta de simulación para la Codificación de Señales de Voz . . . 34

**7. El DSP TMS320C6713** **39**
7.1. Características del Microprocesador TMS320C6713 . . . 39
7.2. Arquitectura del Microprocesador . . . 40
7.2.1. Unidad de procesamiento Central (CPU) . . . 40
7.2.2. Caminos de datos del CPU . . . 41
7.2.3. Archivos de registros de propósito general (register files) . . . 41
7.2.4. Unidades funcionales . . . 43
7.2.5. Archivos de registros de control del TMS320C6713 . . . 43
7.2.6. Caminos entre archivos de registros (Register Filters) . . . 43
7.2.7. Caminos de Memoria, Cargas y Almacenamiento . . . 44
7.2.8. Caminos de direccionamiento de datos . . . 44
7.2.9. Mapeo entre instrucciones y Unidades Funcionales . . . 44
7.3. Modos de direccionamiento . . . 46
7.4. Interrupciones . . . 46

**8. Implementación de la Codificación de Voz en el TMS320C6713** **47**
8.1. Implementación del banco de filtros de cinco canales . . . 47

**9. Conclusiones** **51**

## Apéndices **53**

**A. Códigos** **55**
A.1. Código de Banco de Filtros de 5 Canales . . . 55

**Bibliografía** **59**

# Índice de figuras


2.1. Analogía Gráfica de la Transformada Z. . . . . 8
2.2. Efecto Aliasing . . . . . 10

3.1. Mapeo de la Frecuencia analógica con respecto a la frecuencia digital . . . . . 14
3.2. Diagrama de bloques del proceso de diseño de un filtro IIR con el uso de la transformada bilineal . . . . . 15

4.1. Proceso de Submuestreo . . . . . 17
4.2. Señal Original . . . . . 18
4.3. a)Señal submuestreada a partir de la señal en Figura 4.2 y b) Señal sobremuestreada a partir de la señal submuestreada. . . . . 19
4.4. Proceso de Sobremuestreo . . . . . 20
4.5. Sobremuestreo de una señal . . . . . 20
4.6. Equivalencias en Cascada . . . . . 21
4.7. Descomposición Polifase . . . . . 22
4.8. Decimador o Submuestrador . . . . . 23
4.9. Implementación de un decimador y descomposición Polifase . . . . . 24

5.1. Triángulo de Pascal . . . . . 27
5.2. Descripción gráfica para calcular los elementos de la matríz $P_{LP}^{n}$ . . . . . 27
5.3. Descripción gráfica para calcular los elementos de la matríz $P_{HP}^{n}$ . . . . . 29
5.4. Descripción gráfica para calcular los elementos de la matríz $P_{BP}^{n}$ . . . . . 31

6.1. Banco de Filtros empleando la Matríz de Pascal . . . . . 34
6.2. Banco de Filtros implementado en Simulink de MATLAB . . . . . 35
6.3. Subsistema de Filtrado para el caso de $H_0$ . . . . . 35
6.4. Configuración del Decimador . . . . . 35
6.5. Configuración del Interpolador . . . . . 36
6.6. Señales de entrada y salida en el banco de filtros de 5 canales técnica de Matríz de Pascal . . . . . 36
6.7. Espectrograma del banco de filtros de 5 canales en Simulink. . . . . 37

7.1. Diagrama de Bloques de TMS320C6713 . . . . . 41
7.2. Camino de los datos del TMS320C6713 . . . . . 42

8.1. Implementación del banco de filtros de 5 canales en Simulink . . . . . 48
8.2. Ejecución del banco de filtros en Code Composer . . . . . 49

# Índice de tablas

5.1. Número de columnas de la matríz $R^n_{BP}$ . . . . . 30

7.1. Unidades funcionales y las operaciones realizadas . . . . . 43
7.2. Registros de Control . . . . . 44
7.3. Registros de control extendidos para el TMS320C6713 . . . . . 44
7.4. Mapeo de instrucciones de punto fijo y unidades funcionales . . . . . 45
7.5. Mapeo de instrucciones de punto flotante y unidades funcionales . . . . . 45
7.6. Generación de direccionamiento indirecto para Load/Store . . . . . 45

## Codificación de Señales de Audio en Tiempo Real utilizando Microprocesadores de la Familia TMS320C6713

Tesis de Maestría

### Resumen

Ing. Adán Bonilla Chávez
**Facultad de Ingeniería**
**Universidad Nacional Autónoma de México**

El presente trabajo pretende realizar un estudio de la codificación de señales de Audio mediante el algoritmo de la Matríz de Pascal. Se pretende dar una muestra de lo poderoso que puede ser dicho método mediante procesos muy sencillos. Dicho algoritmo ha sido discutido y presentado en los últimos años como un método innovador debido a que se requieren de operaciones matriciales muy simples que pueden ser programadas de una manera muy rápida y eficiente mediante las herramientas adecuadas, sin embargo, dicho método no ha sido llevado a la práctica en entornos reales (existen evidencias a manera de simulaciones de que es muy probable que pueda funcionar, pero no ha sido probado en aplicaciones reales).

La aplicación directa que se pretende en este trabajo es en la codificación de voz. Se eligió este fenómeno debido a que a pesar de que es muy común en nuestro entorno, aún sigue siendo de gran utilidad su estudio en la industria de las telecomunicaciones. Al codificar de manera eficiente la voz humana, podemos evitar retrasos en su transmisión y podemos hacer más eficiente dicho proceso. Al estudiar el fenómeno de la voz y su codificación mediante el algoritmo propuesto, se pretende sentar las bases para estudios subsecuentes en otras áreas de particular interés para las telecomunicaciones, e incluso para otros campos en el que pueda ser aplicable el método aquí expuesto.

Para el estudio de dicho algoritmo, se presenta la teoría que respalda dicho proceso y se detalla en los primeros capítulos las bases desde donde está sentado. Al analizar la teoría correspondiente, es posible justificarlo.

Posteriormente, se realiza una descripción exhaustiva del método para poder comprenderlo y se hace un análisis de cómo puede ser utilizado en procesos de codificación de voz. La explicación exhaustiva de este algoritmo también se realiza con la finalidad de que otros actores interesados en codificación y procesamiento digital de señales puedan consultar esta obra y tener un panorama completo del algoritmo como tal.

Una vez analizado dicho algoritmo, se explica la aplicación específica en un DSP Kit Develop, en el cual se puede apreciar el Procesamiento y Codificación de Señales de voz en tiempo real. Así mismo se evalúa su calidad y su utilidad en el entorno de las telecomunicaciones.

Finalmente, se dan impresiones de lo realizado a manera de conclusiones y sugerencias específicas que pueden tomarse posteriormente para el estudio en otras áreas afines dentro de las telecomunicaciones.

## Coding of Digital Signals of Audio in Real Time using microprocessors of TMS320C6713's Family

Master Degree Thesis

### Abstract

Ing. Adán Bonilla Chávez
**Faculty of Engineering**
**National Autonomous University of Mexico**

This work aims to conduct a study of the Speech signals coding using the Pascal Matrix Algorithm. It aims to give an example of how powerful it can be said very simple method by this process. This algorithm has been discussed and presented in recent years as an innovative method because they require very simple matrix operations that can be programmed in a very quick and efficient way with the right tools, however this method has not been carried practice in real settings (there is evidence by way of simulations is likely to work, but has not been tested in real applications).

Direct application is intended in this work is in speech coding. This phenomenon was chosen because despite being very common in our environment, it is still very useful in the study of the telecommunications industry. To efficiently encode the human voice, we can avoid delays in transmission and can streamline the process. In studying the phenomenon of voice and consolidated in the proposed algorithm is intended to lay the groundwork for subsequent studies in other areas of particular interest to telecommunications, and even to other fields in which the method can be applied here exposed.

For the study of this algorithm, we present the theory behind the process and detailed in the opening chapters of the bases from where he sits. Analyzing the corresponding theory, it is possible to justify it.

Subsequently performed an exhaustive description of the method to understand and provides an analysis of how it can be used in speech coding processes. The full explanation of this algorithm is also done in order that other stakeholders in coding and digital signal processing to consult this work and have a complete overview of the algorithm as such.

After analyzing the algorithm is explained in a specific application "Kit Develop DSP", in which you can see the Signal Processing and Coding of speech in real time. It also evaluates the quality and usefulness in the telecommunications environment. Finally, there are views of achievements as conclusions and specific suggestions that can be taken later to study in other related areas within telecommunications.

# Capítulo 1

# Introducción

Sin duda, uno de los grandes avances en la era moderna en cuanto a tecnología se refiere, se ha presentado en el área del Procesamiento Digital de Señales. Gracias a este enorme avance, numerosos fenómenos han podido ser caracterizados de una forma mas eficiente y se ha podido tener un estudio mas detallado de los mismos.

Gracias al Procesamiento Digital de Señales, se ha logrado estudiar señales antes muy poco entendidas, extremedamente complejas o simplemente inentendibles para el ser humano. Recordemos que la teoría de Análisis de Sistemas y Señales ha dado pasos gigantes en los últimos dos siglos, sin embargo, las técnicas de análisis y sintésis de los mismos requieren de algoritmos y métodos sumamente complejos y muy difíciles de procesar de manera manual. [(6)]

El Procesamiento Digital de Señales (PDS) nos ha permitido muestrear, cuantificar y codificar numerosas señales analógicas para su posterior análisis mediante sistemas computacionales. Esto ha impulsado el estudio de diversas señales y nos ha dado luz de muchos fenómenos que antes considerábamos incluso inentendibles o imposibles de estudiar. Mediante el PDS hemos logrado comunicarnos desde distancias remotas; escuchado de manera clara los latidos de nuestro corazón (e incluso estudiar enfermedades y plantear la cura de algunas de ellas en base a su estudio); obtener y almacenar sonidos, imágenes e incluso videos (de hecho, el video no podría concebirse sin el PDS), entre otro tipo de aplicaciones que hoy nos parecen tan comunes y tan importantes en nuestra vida cotidiana.

Aunque se han dado avances significativos dentro del área del Procesamiento Digital de Señales, uno de los retos que aún sigue vigente en nuestros tiempos es la cantidad de algoritmos que se requiere para todo el proceso de conversión analógico-digital. Prácticamente toda la teoría de Análisis de Sistemas y Señales se basa en algoritmos y métodos matemáticos que, aunque son muy precisos y confiables, requieren de cálculos sumamente complejos y muy tardados.

Se puede llegar a pensar que con la tecnología disponible actualmente, lo escrito anteriormente no es impedimento, sin embargo hay aplicaciones en las cuales el tiempo es un factor decisivo. Hay aplicaciones que requieren del PDS prácticamente en tiempo real, [1] y la demora de incluso una fracción mínima de tiempo implicaría un funcionamiento no adecuado o esperado de nuestro sistema en cuestión.

Actualmente, son numerosos los ejemplos que podemos mencionar dentro de las Aplicaciones de Tiempo Real. Tan solo pensemos en las Videoconferencias (el retraso de información entre el transmisor y el receptor puede ser un factor que impida una comunicación efectiva), la transmisión de voz y datos, la Telemedicina (las señales deben tener perfecta sincronía entre los robots y las manos del cirujano para garantizar que ciertas tareas médicas surtirán efectos óptimos e incluso salvarán vidas), los sistemas computarizados en los motores de un automóvil (los cuales deben de tener una

[1] Entiéndase por "Aplicaciones en Tiempo Real" como aquellas aplicaciones enmarcadas dentro de un sistema digital que interactúa activamente con un entorno con dinámica conocida en relación con sus entradas, salidas y restricciones temporales, para darle un correcto funcionamiento de acuerdo con los conceptos de predictibilidad, estabilidad, controlabilidad y alcanzabilidad [ (8)]

perfecta sincronía con el sistema mecánico), entre otras. Pensemos además que entre mas complicadas y de mayor complejidad sean las tareas a efectuar por algún sistema en cuestión, mayores serán los requerimientos físicos como financieros.
La elaboración de Métodos y Algoritmos en Tiempo Real mas eficientes y de mejor calidad permitirán crear Sistemas Tecnológicos cada vez más rápidos sin sacrificar la calidad, optimizando así la cantidad de recursos que se utilicen para el diseño de los mismos y minimizando costos tanto financieros como de procesamiento. Si los algoritmos empleados son sencillos y menos complejos, ello repercute en sistemas cada vez mas veloces y pequeños.
Si bien la miniaturización es ya factible en gran medida en nuestro tiempo, aún se tiene el reto de la complejidad en los procesos llevados por los Microprocesadores. Al tener una mayor complejidad en las tareas realizadas por un microprocesador, mayor cantidad de circuitos y recursos serán necesarios para su funcionamiento, lo cual inside directamente en el costo de los mismos.
La idea esencial del presente trabajo es diseñar, analizar, implementar y probar un Sistema de Codificación de Voz mediante Procesadores Digitales de Señales de la Familia Texas Instruments, utilizando algoritmos y métodos eficientes e incluso proponiendo algunos que sean de mejor calidad. Se decidió utilizar la Voz al ser un ejemplo claro de una señal que requiere en muchos casos de un Procesamiento en Tiempo Real para aplicaciones en el área de la Telecomunicaciones.
Es cierto que actualmente la tendencia es la transmisión no solo de voz, sino también de imágenes e incluso de la combinación de ambas señales y en movimiento. Sin embargo, el estudio de un algoritmo mas eficiente requiere de un mayor tiempo de estudio y análisis que está fuera del alcance del presente trabajo, de manera que se pretende sentar las bases teóricas y prácticas para que en un futuro estas mismas sirvan como referencia para su aplicación en señales en tiempo real mas complejas.

## 1.1. Objetivos Generales

- Estudiar y Analizar la Transformada Z y su aplicación en el filtrado de Señales para su implementación en Procesadores Digitales de Señales de Voz.
- Estudiar y Analizar el Método de la Matríz de Pascal para su implementación en Procesadores Digitales de Señales de Voz.
- Diseñar e Implementar un Sistema de Codificación de Voz, empleando métodos y algoritmos basados en la Transformada Z y la Transformada de Fourier.
- Diseñar e Implementar un Sistema de Codificación de Voz empleando métodos y algoritmos mas eficientes, tales como la Matríz de Pascal.
- Comparar los sistemas antes propuestos y evaluar su utilidad en Aplicaciones en tiempo Real.
- Proponer mejoras en los sistemas implementados y establecer objetivos de estudio para la implementación futura en aplicaciones que impliquen imágenes y video

## 1.2. Marco Teórico

Mediante esta tesis se pretende desarrollar un algoritmo eficiente para el Procesamiento Digital de Señales de Voz con la finalidad de implementarlo de manera física en Microprocesadores Texas Instruments de la Familia TMS320CXX.
Este proyecto pretende impactar de manera positiva en el desarrollo de dispositivos de Procesamiento Digital de Señales para aplicaciones en Tiempo Real. Se busca implementar de manera física un sistema de Procesamiento Digital de Señales de Voz para tener un marco teórico que permita en un futuro desarrollar aplicaciones de Procesamiento Digital de Señales de Imágenes e incluso Video. Con ello se pretende también impulsar la investigación en un área que es de suma importancia pero que sin embargo no ha tenido el impulso ni la calidad suficiente.

Se pretende crear investigación para el mejoramiento del Procesamiento de Señales Discretas con la inclusión y desarrollo de nuevas técnicas mas innovadoras, algoritmos mas eficaces, sistemas de gran calidad y la generación de conocimiento que permita una sólida formación de Recursos de diversas índoles.

### 1.2.1. Justificación

En la actualidad, se cuenta con algoritmos que son eficaces pero no son eficientes dentro del área de Procesamiento Digital de Señales. De esa manera, tenemos que hay aplicaciones que cuentan con una gran calidad pero que cuentan con algoritmos y métodos de un grado mayor de complejidad que comprometen de manera negativa el desarrollo de Aplicaciones en tiempo real.
El hecho de contar con algoritmos eficaces pero no eficientes implica tener que recurrir a Microprocesadores sumamente complejos, lo cual hace que los Sistemas Orientados a Aplicaciones en Tiempo Real (SOATR) sean sumamente costosos en terminos financieros y tecnológicos.
Por una parte, contamos con SOATR sumamente poderosos, pero que por la complejidad y la cantidad de las operaciones que realizan, requieren de un espacio mayor para su operación, o bien existen SOATR de tamaños sumamente pequeños pero su costo suele ser demasiado elevado.
Es por ello que el presente trabajo tiene la finalidad de explorar, diseñar, implementar y analizar métodos y algoritmos mas eficientes para el Procesamiento Digital de Señales de Voz para con ello sentar las bases que permitan el diseño de SOATR mas eficientes sin tener que sacrificar costos financieros ni tecnológicos.

### 1.2.2. Metodología

Para elaborar el presente trabajo, se pretende realizar las siguientes acciones:

- Establecer el marco Teórico de la Transformada Z para el análisis de Sistemas y Señales Orientado a Procesamiento Digital.
- Establecer el marco Teórico de los elementos básicos que requiere el Procesamiento Digital de Señales (adquisición, muestreo, filtrado, cuantización y codificación)
- Establecer el marco Teórico del Método de la Matríz de Pascal [2]
- Diseñar e Implementar un Sistema de Codificación de Voz empleando métodos y algoritmos mas eficientes, tales como la Matríz de Pascal.
- Comparar los sistemas antes propuestos y evaluar su utilidad en Aplicaciones en tiempo Real.
- Proponer mejoras en los sistemas implementados y establecer objetivos de estudio para la implementación futura en aplicaciones que impliquen imágenes y video

Las principales áreas tecnológicas de aplicación son las telecomunicaciones en el área de Procesamiento Digital de Señales (conversión de señales analógicas a digitales y su posterior transmisión) y puede tener cierto impacto en áreas como la medicina, la microelectrónica, la industria espacial y las telecomunicaciones.
Ahora bien, sabiendo que se tiene como problema primordial el hecho de tener algoritmos eficaces pero poco eficientes en el rubro de los SOATR, se pretende implementar mediante Microprocesadores un nuevo Algoritmo de Procesamiento Digital de Señales conocido como la Matriz de Pascal. Este método, aunque ha sido ampliamente descrito, no ha sido explorado ni mucho menos implementado, de manera que con esta nueva técnica se pretende llegar a algoritmos eficaces y eficientes.
La meta es conseguir un sistema de Procesamiento de Voz en Tiempo Real mediante el uso de la técnica antes descrita, además de compararlo con algunas de las técnicas conocidas actualmente (Transformada Z, Banco de Filtros, Transformada Bilineal, entre otras) y emitir un juicio objetivo

[2] El Método de la Matríz de Pascal es ampliamente explicado en este trabajo, consulte para mayores referencias el Capítulo 5

de si es posible mejorar lo ya existente. Lo anterior también nos llevará a proponer un camino y una metodología que podría emplearse en un futuro para el Procesamiento Digital de Imágenes e incluso de Video (si bien estas señales están completamente fuera del alcance de este trabajo, se pretende emitir una metodología orientada a desarrollar en un futuro algoritmos de Procesamiento Digital para este tipo de señales.

En primer lugar, se pretende investigar lo concerniente y actualmente disponible acerca del Procesamiento Digital de Voz. Se busca comprender y analizar los métodos actualmente existentes y posteriormente implementar la Técnica de Matríz de Pascal. Con lo conseguido hasta ese momento, se realizarán pruebas de desempeño y comparación entre algunos métodos existentes y el método propuesto para concluir de manera fehaciente si es posible o no mejorar la eficiencia de las técnicas actualmente disponibles.

Para llevar a cabo la tarea anteriormente descrita, se pretende implementar las aplicaciones de Procesamiento Digital de Voz en Microprocesadores Texas Instruments de la Serie TMS320CXX. Se escogieron estos microprocesadores pues son los que actualmente están disponibles en el Departamente de Telecomuniciones de la Facultad de Ingeniería de la UNAM. Estos microprocesadores nos permitirán evaluar de una manera mas certera y objetiva el desempeño de los algoritmos propuestos, puesto que son de funciomiento básico, no son costosos y no requieren de un grado de complejidad elevado en cuanto a desempeño se refiere. Si es posible implementar la Matríz de Pascal en este tipo de microprocesadores, será un hecho que podrá ser implementado y/o extrapolado en cualquier microprocesador mas actual de Texas Instruments.

Cabe señalar que aunque existen procesadores mas potentes y que permiten realizar Procesamiento Digital de Señales de manera exitosa con las técnicas actualmente disponibles, estos requieren de mayores necesidades informáticamente hablando, lo cual implica un aumento en el costo del desarrollo de SOATR. Ello hace a los SOATR el impedimento principal para desarrollar aplicaciones baratas y eficientes.

### 1.2.3. Objetivo de la Tesis

Los Sistemas Orientados a Aplicaciones en Tiempo Real (SOATR) no solo requieren de algoritmos eficaces [3] sino que tambièn requieren que estos últimos sean eficientes. Al conseguir que se cumplan ambas metas, se puede hablar de un SOATR de alta calidad.

El objetivo Primordial del siguiente Trabajo puede resumirse en los siguientes enunciados.

- Comprender y analizar los algoritmos existentes en la actualidad para el Procesmiento Digital de Voz
- Describir el marco Teórico del Algoritmo de Procesamiento Digital de Señales empleando la Matrìz de Pascal.
- Implementar un SOATR empleando el Algoritmo de Procesamiento Digital de Señales empleando la Matrìz de Pascal con la ayuda de Microprocesadores Texas Instruments de la Familia TMS320CXX
- Realizar pruebas exhaustivas del Sistema conseguido, evaluarlo y compararlo con algunas de las tècnicas actuales existentes en cuanto a Procesamiento Digital de Voz en SOATR, evaluando su utilidad en Aplicaciones en tiempo Real.
- Proponer mejoras en los sistemas implementados y establecer objetivos de estudio para la implementación futura en aplicaciones que impliquen imágenes y video

[3] Entiendase con "eficacia" en el presente Trabajo como a la capacidad de un SOATR de tener un desempeño que sea igual o cercanamente equivalente a la entrada y a la salida. Nótese que la Eficacia no es sinónimo de Eficiencia. En cuanto a Eficiencia, se entiende que este término se refiere a la capacidad que tienen un SOATR de realizar su tarea dentro de parámetros de tiempo aceptables

### 1.2.4. Contribución del Presente Trabajo

La contribución del presente trabajo es dotar de un algoritmo eficiente que permita a Investigadores en el Área de Procesamiento Digital de Señales contar con una herramienta útil y eficaz que facilite el Procesamiento y Codificación de Audio, en especial de señales complejas como lo es la voz, utilizando como principal herramiento bancos de filtros digitales y Microprocesadores de la familia TMSC6713

# Capítulo 2

# Antecedentes y Estado del Arte

El presente capítulo se dedica a los antecedentes sobre el trabajo presentado. Se divide en tres partes: La Voz como Fenómeno Físico; La Transformada Z y su utilidad en el Procesamiento Digital de Señales; y finalmente Estado del Arte. Se dan los antecedentes y conceptos básicos sobre la Voz Humana y posteriormente se da un breve repaso sobre la Transformada Z, para finalmente hacer una revisión del uso de Microprocesadores para el Procesamiento Digital de Voz, especificamente para la codificación de la misma.

## 2.1. La Voz como Fenómeno Físico

Una manera de estudiar los fenómenos que acontecen a nuestro alrededor es mediante un enfoque de sistemas. Partimos del hecho de que todo lo que nos rodea está conformado por 1 o más sistemas que permiten su realización o aparición. Una manera de estudiar dichos sistemas es mediante el análisis y síntesis de los mismos.
Una forma para iniciarse en el adentramiento del estudio de la caracterización de dichos sistemas es comprendiendo fenómenos de manera ideal. Un ejemplo de un fenómeno "ideal" puede ser una función matemática bien caracterizada (como una función seno o una función coseno). Sin embargo, la realidad es que la mayoría de los fenómenos que suceden a nuestro alrededor no se presentan de manera ideal.
Casi todos los fenómenos a nuestro alrededor ocurren de manera aleatoria, es decir, de manera no previsible. No podemos saber con exactitud la fecha y hora exacta en el que temblará y con qué intensidad, o bien tampoco sabemos con qué frecuencia latirá nuestro corazón en los próximos 5 minutos, ni mucho menos podemos saber con exactitud si el día de mañana será soleado o nublado. De lo anterior, se tiene la necesidad de estudiar todos los fenómenos que nos ocurren con un enfoque de análisis y síntesis que involucre la aleatoriedad como un factor preponderante.
Gracias a la probabilidad y a la estadística es posible caracterizar un fenómeno aleatorio. Un ejemplo de ello es que no podemos saber con certeza y exactitud quien será el próximo ganador en una votación presidencial de un país dado, sin embargo, mediante técnicas estadísticas y probabilísticas es posible saber la tendencia que llevará el curso de dicho proceso y estimar quien es el probable vencedor en dicha contienda.
La voz humana es un excelente ejemplo de un proceso aleatorio, no es lo mismo el tono de voz cuando apenas hemos despertado que cuando ya llevamos avanzada nuestra jornada, o incluso no es el mismo tono de voz cuando estamos contentos que cuando estamos tristes. A pesar de la aleatoriedad de dicho fenómeno, existen ciertos parámetros que nos permiten caracterizarlo e incluso comprenderlo un poco mejor, de manera que podamos aplicar dicho estudio para el beneficio de nuestros intereses. La investigación acerca del procesamiento de voz ha llevado a la creación de importantes aportaciones técnicas a la sociedad. El soporte fundamental de la voz es el sonido. El sonido se caracteriza por fluctuaciones de presión en un medio compresible. Dos cosas deben existir a fin de que se produzca una onda sonora: una fuente mecánica de vibración y un medio elástico a través del cual pueda propagar la perturbación. [(9)]

La voz se produce por la vibración de las cuerdas vocales. El aire exhalado de los pulmones es modulado y dado forma por la vibración en las cuerdas vocales y el tracto vocal. Ese sonido producido por la vibración de las cuerdas vocales es llevado al exterior por el propio aire espirado que causó la vibración.
Las señales de voz, que no son más que una secuencia de símbolos acústicos elementales (conocidos como fonemas), además de servir para la comunicación de la información lingüística, llevan consigo multitud de información de otros tipos. La voz no es igual para todas las personas.
Las investigaciones que han existido en este campo estudian temas relacionados con el análisis y síntesis de voz, la creación de códigos de voz, el reconocimiento de voz, el realce de voz ó la comunicación multimodal.
Diversos sistemas basados en bancos de filtros han sido la base para demostrar que una señal de voz se puede representar en términos de un paquete de parámetros. Estos parámetros pueden utilizarse posteriormente no sólo para re-sintetizar la señal de voz, sino para reconocer la identidad de la persona que está hablando. Además abre la posibilidad de comprimir el ancho de banda de una señal de voz, principio que actualmente se utiliza en la telefonía digital moderna. [(10)]

## 2.2. La Transformada Z y su utilidad en el Procesamiento Digital de Señales

La Transformada Zeta (TZ) es una transformación matemática que se emplea, entre otras aplicaciones, en el estudio del Procesamiento de Señales Digitales, como son el análisis y síntesis de Circuitos Digitales, los Sistemas de Radar o Telecomunicaciones y especialmente los Sistemas de Control de Procesos por computadoras. [(1)]
La TZ es un ejemplo más de Transformada, como lo son la Transformada de Fourier para el caso de tiempo discreto y las Transformada de Fourier y Laplace para el caso del tiempo continuo.
La importancia de la Transformada Z radica en que permite reducir Ecuaciones en Diferencias o ecuaciones recursivas con coeficientes constantes a Ecuaciones Algebraicas lineales.
La Transformada Zeta es una aplicación entre un espacio de Sucesiones (funciones discretas) y un espacio de Funciones Analíticas (desarrollables en serie de Laurent).
La función que los liga es la Serie de Laurent cuyos coeficientes son los elementos de la Sucesión de origen. La importancia del modelo de la Transformada Zeta radica en que permite reducir Ecuaciones en Diferencias o ecuaciones recursivas con coeficiente constantes a Ecuaciones Algebraicas lineales (Figura 2.1).

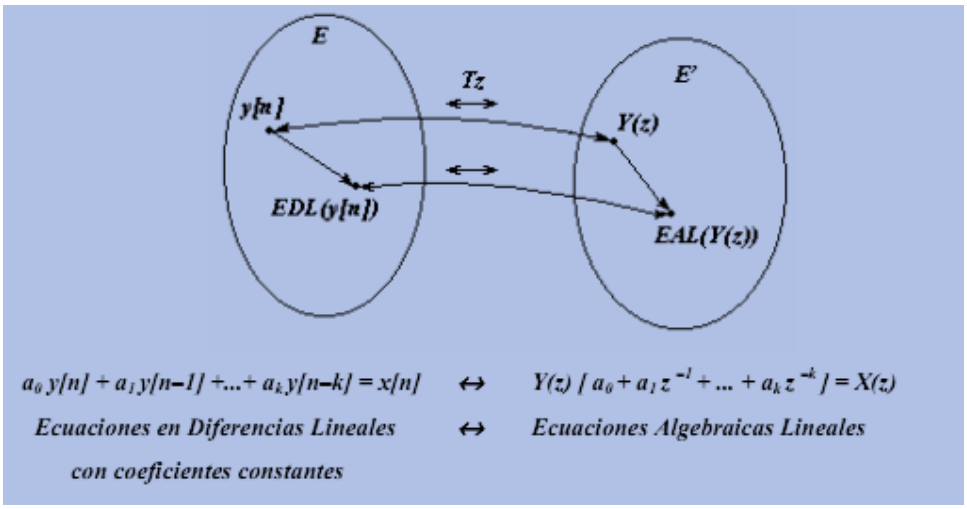


Figura 2.1: Analogía Gráfica de la Transformada Z.

La utilidad de la Transformada Z radica en que es posible estudiar el muestreo de señales analógicas y realizar su conversión a señales digitales de una manera mas eficiente y sin involucrar operaciones tan complejas, tal y como se haría en algunas otras transformadas (Tal como es la Transformada de Fourier).

## 2.3. Estado del Arte

### 2.3.1. Técnicas empleadas para la Codificación de Señales de Voz

Con el avance de la tecnología y la invención de equipos cada vez más sofisticados surge la necesidad de asegurar su integridad, autenticidad de los usuarios y confidencialidad para lograr una comunicación segura por un canal de voz.
Diseñar un sistema de comunicación motiva a tomar en consecuencia varios criterios teóricos para llevarlos a la práctica a una determinada aplicación. En tal sentido las tecnologías empleadas para la codificación de voz hacen uso de los siguientes conceptos:

1. Transformada de Fourier y Transformada Inversa de Fourier.
2. Scrambler.
3. Elección de la clave.
4. Técnicas de codificación empleadas por Scrambler.

El teorema de Nyquist es más conocido como el Teorema de Muestreo por ser un criterio muy inherente en la teoría de la comunicación y se define de la siguiente manera. [(2)]
El teorema establece que una señal de banda limitada a B Hz definida en un intervalo mostrado en la expresión ( 2.1) se puede reconstruirse a partir de sus muestras tomadas uniformemente a una razón no menor de 2B muestras por segundo.

$$f_0 - \frac{B}{2} \prec\mid f \mid\prec f_0 + \frac{B}{2} \tag{2.1}$$

Cuando existe la necesidad de reconstrucción de una señal automáticamente se le asocia a la idea del valor de Nyquist. Este concepto permite encontrar el valor de muestreo que nos dará una reconstrucción perfecta de la señal. Si se muestrea con un valor por debajo del valor de Nyquist surgirán problemas para hacer la reconstrucción. A este problema se le conoce como *Aliasing* (algunos autores lo llaman solapamiento). El efecto Aliasing ocurre cuando hay un traslape en el desplazamiento, o bien copias periódicas de la señal en frecuencia.
En el dominio de la frecuencia, se observa que parte de la señal se trasladará con la señal siguiente a él. En este solapamiento los valores de la frecuencia serán sumados juntos y la forma del espectro de la señal será indeseablemente alterada como se puede apreciar en la figura 2.2
La figura 2.2 se divide en tres gráficas una debajo de la otra. La primera gráfica muestra una señal limitada en banda a (W Hz). En la segunda gráfica se ilustra un muestreo en la cual ocurre una superposición de la señal llamada Aliasing. En la tercera gráfica se muestrea la señal cumpliendo con el teorema de Nyquist evitando el Aliasing. Nótese que si la señal no fuera limitada en banda, el componente del espectro siempre sería traslapado

#### 2.3.1.1. Transformada de Fourier

La transformada discreta de Fourier (DFT) es uno de dos grandes procedimientos encontrados en el campo del tratamiento digital de señales. El otro procedimiento es el filtrado digital. La DFT nos permite analizar, manipular y sintetizar señales no posibles con procesamiento de señales continuas ó analógicas.

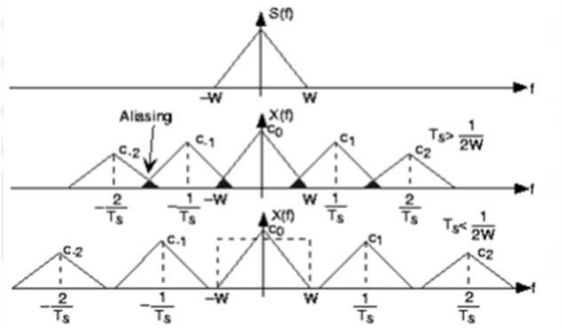


Figura 2.2: Efecto Aliasing

La DFT es un procedimiento matemático para determinar las armónicas ó frecuencias contenidas en una secuencia de señal discreta. Para propósitos de la presente tesis, una secuencia de señal discreta es un conjunto de valores obtenidos mediante el muestreo periódico de una señal continua en el dominio del tiempo. [(3)]
La DFT tiene su origen de la transformada continua de Fourier y se define:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \tag{2.2}$$

Donde x(t) es alguna señal continua en el dominio del tiempo. En el campo del procesamiento de señales continuas la ecuación ( 2.2) se usa para transformar una expresión de una función continua en el dominio del tiempo a X(f) una función continua en el dominio de la frecuencia. La función X(f) permite conocer el contenido de frecuencias de cualquier señal de interés [(4)]
Con la llegada de las computadoras y los esfuerzos anticipados de los primeros procesamientos digitales se motivó a una mayor difusión de la DFT definida como la secuencia discreta en el dominio de la frecuencia donde X(m) es la ecuación DFT de forma exponencial, como se puede mostrar en la ecuación (2.3)

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi n\frac{m}{N}} \tag{2.3}$$

### 2.3.2. Importancia de la Matríz de Pascal en el Procesamiento Digital de Señales

Como hemos analizado hasta este momento, contamos con algoritmos sumamente eficaces en cuanto a Procesamiento Digital de Señales, pero no lo suficientemente eficientes. Esto retrasa la calidad de Filtros Digitales y el tiempo de procesamiento, haciendo costosos los sistemas de procesamiento en términos de tiempo de procesado e incluso en términos monetarios. [(7)]
Es por ello que surge la necesidad de implementar sistemas que muestren eficiencia y eficacia en el rubro de Procesamiento de Señales Orientadas a Telecomunicaciones. Surgen métodos de análisis que requieren solamente de sumas y restas y de operaciones matriciales que en términos computacionales suelen ser muy rápidos y de gran versatilidad.
El objetivo de este trabajo en su totalidad es analizar el método de la Matríz de Pascal para el Procesamiento Digital de Señales, al ser una herramienta de gran utilidad para el desarrollo de aplicaciones en Tiempo Real. Con ello, estaremos en posibilidades de codificar señales complejas como lo son las señales de voz de manera muy eficiente, sin sacrificar la calidad de la misma. [(5)]

### 2.3.3. Procesadores Digitales de Señales

Un sistema de procesado digital de señal puede definirse como cualquier sistema electrónico que realice procesado digital de señal, entendiéndose por él la aplicación de operaciones matemáticas a señales representadas de forma digital. Las señales son representadas de forma digital mediante secuencias de muestras. A menudo, estas muestras se obtienen de señales físicas (por ejemplo, señales de audio) utilizando transductores (un micrófono en este caso) y convertidores analógico-digitales. Después del procesado matemático, las señales digitales pueden volver a convertirse en señales físicas mediante convertidores digital-analógicos. Si bien, en principio, el corazón de un sistema de procesado digital puede ser un microcontrolador, un procesador de propósito general o un procesador digital de señal (DSP), en sistemas en los cuales la carga computacional es extremadamente intensa la solución óptima pasa por escoger a un DSP.
En la actualidad, los cuatro grandes fabricantes de DSP son Texas Instruments, con la serie TMS320; Motorola, con las series DSP56000, DSP56100, DSP56300, DSP56600 y DSP96000; Lucent Technologies (anteriormente AT&T), con las series DSP1600 y DSP3200; y Analog Devices, con las series ADSP2100 y ADSP21000.

#### 2.3.3.1. ¿Qué es un DSP?

Estrictamente hablando, el término DSP se aplica a cualquier chip que trabaje con señales representadas de forma digital. En la práctica, el término se refiere a microprocesadores específicamente diseñados para realizar procesado digital de señal. Los DSP utilizan arquitecturas especiales para acelerar los cálculos matemáticos intensos implicados en la mayoría de sistemas de procesado de señal en tiempo real. Por ejemplo, las arquitecturas de los DSP incluyen circuitería para ejecutar de forma rápida operaciones de multiplicar y acumular, conocidas como MAC. A menudo poseen arquitecturas de memoria que permiten un acceso múltiple para permitir de forma simultánea cargar varios operandos, por ejemplo, una muestra de la señal de entrada y el coeficiente de un filtro simultáneamente en paralelo con la carga de la instrucción. También incluyen una variedad de modos especiales de direccionamiento y características de control de flujo de programa diseñadas para acelerar la ejecución de operaciones repetitivas. Además, la mayoría de los DSP incluyen en el propio chip periféricos especiales e interfaces de entrada salida que permiten que el procesador se comunique eficientemente con el resto de componentes del sistema, tales como convertidores analógico-digitales o memoria. [(11)]
La diferencia esencial entre un DSP y cualquier otro microprocesador es que el DSP tiene características diseñadas para soportar tareas de altas prestaciones, repetitivas y numéricamente intensas. Por contra, los microprocesadores de propósito general o microcontroladores de cualquier otro tipo pueden no estar especializados para ninguna aplicación en especial; en el caso de los microprocesadores de propósito general, por ejemplo, no están orientados a aplicaciones de control ni de procesamiento de señales.
Aunque el ejemplo del filtro de respuesta impulsional finita (FIR) ha sido ampliamente utilizado en el entorno DSP, es quizás el más simple que permite ilustrar la necesidad de estas prestaciones en los DSP, las cuales permiten concebir muchas de las funciones de procesado en tiempo real.

# Capítulo 3

# Transformación Bilineal como Método Base para la Obtención de funciones de transferencia en el Dominio de Z

## 3.1. La Transformaciòn Bilineal: Marco Teórico

La transformación bilineal es un mapeo de un plano s complejo analógico a un plano z complejo digital. La conversión mapea los polos y ceros analógicos en polos y ceros digitales, donde un punto del plano s es mapeado a un único punto del plano z. En términos matemáticos, mediante la transformación bilineal se pretende pasar de un dominio desde $-\infty \leq \Omega \leq \infty$ a un dominio sobre $-\pi \leq \omega \leq \pi$ , (12) Esta transformación está basada en la técnica de integración numérica, usada para simular un integrador de un filtro analógico, la cual está definida por:

$$s = (\frac{2}{T})(\frac{1 - z^{-}1}{1 + z^{-}1}). \tag{3.1}$$

Donde T es el periodo de muestreo, la relación inversa esta dada por:

$$z = \frac{2 + sT}{2 - sT}. \tag{3.2}$$

El mapeo de la frecuencia analógica con respecto a la frecuencia digital (Figura 3.1) se relaciona por la siguiente ecuación:

$$\Omega = (\frac{2}{T})(\frac{\tan w_d}{2}) \tag{3.3}$$

$$w_d = 2 \arctan \frac{\Omega T}{2} \tag{3.4}$$

La transformada bilineal es uno de los métodos más simples para el diseño de los filtros IIR, el método consiste en:

- Especificar los requerimientos en términos de la frecuencia analógica en Hz, valores de atenuación y la frecuencia de muestreo.

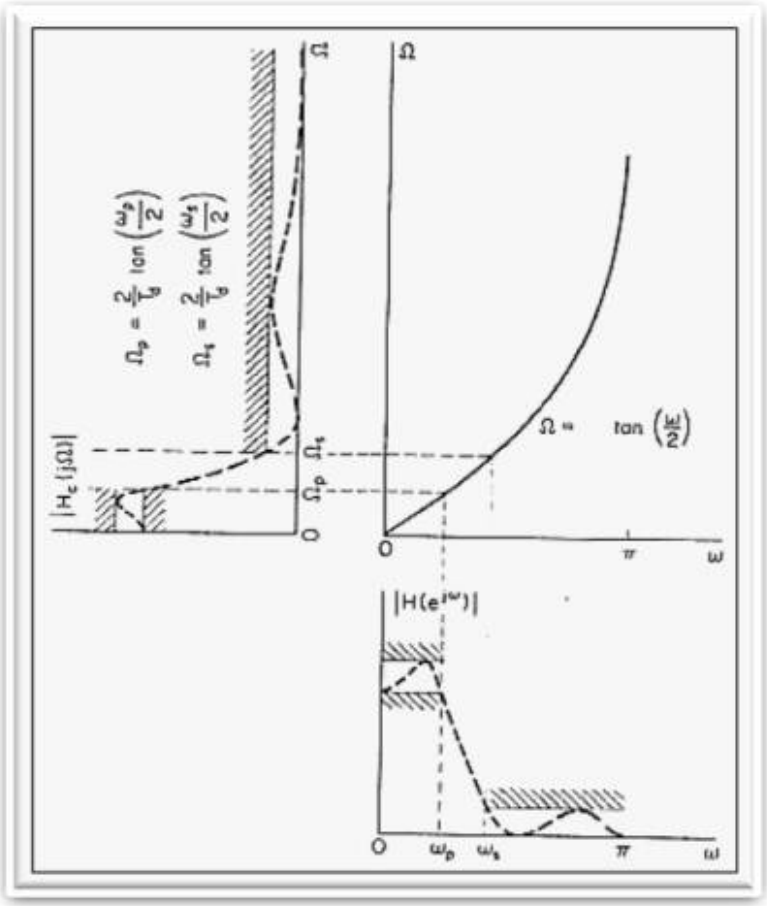


Figura 3.1: Mapeo de la Frecuencia analógica con respecto a la frecuencia digital

- Transformar las especificaciones a un filtro analógico equivalente.
- Diseñar el filtro analógico $H_a(s)$.
- Utilizar la transformación bilineal para obtener el filtro digital $H(z)$.

La figura 3.2 muestra las etapas en el diseño de filtros IIR usando la transformada bilineal

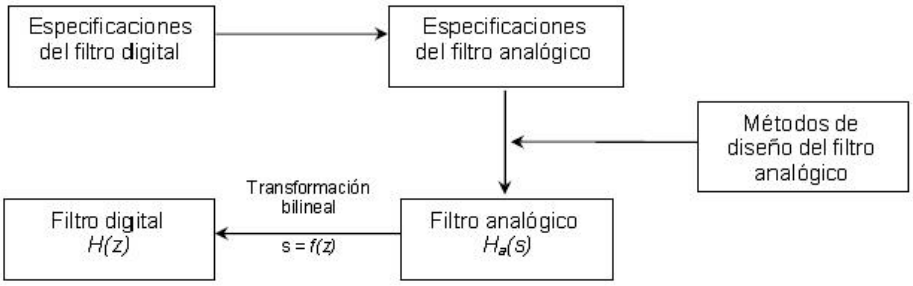


Figura 3.2: Diagrama de bloques del proceso de diseño de un filtro IIR con el uso de la transformada bilineal

## 3.2. Implementación de la Transformada Bilineal en MATLAB para la construcción de un filtro IIR

Como es posible apreciar, realizar el análisis de un filtro digital de manera manual es muy complejo. Se requieren de operaciones que pueden incluso conducir a errores durante el proceso. Para evitar este tipo de errores, es posible utilizar la plataforma MATLAB diseñar un filtro como tal y utilizar las funciones propias de este software.

Mediante el uso de paqueterías como MATLAB, es posible ejemplificar el uso y la aplicaciòn de la Transformada Bilineal en la construcción de filtros digitales.

Sea un filtro digital paso altas con aproximación Chebyshev de orden 4, de 2 dB en su rizo de paso, 1000 Hz en la frecuencia de corte, 8 Khz en la frecuencia de muestreo, es posible realizar el cálculo de los coeficientes de dicho filtro mediante un script de MATLAB. Para ello puede utilizarse la función cheb1ap.

De acuerdo a MATLAB, [1] la función cheb1ap sirve para obtener los coeficientes en el dominio s de un filtro Chebyshev tipo analógico.

La funciòn [z,p,k] = cheb1ap(n,Rp) devuelve los polos y la ganancia de un filtro analógico Chebyshev tipo I, de orden n, paso bajas con 'Rp' dB de onda en la banda de paso. La función devuelve los polos en el vector 'p' de longitud 'n', y la ganancia 'k' como escalar. 'z' es una matriz vacía, porque no hay ceros. La función de transferencia es:

$$H(s) = \frac{z(s)}{p(s)} = \frac{k}{(s-p(1))(s-p(2))...(s-p(n))}. \tag{3.5}$$

Pero nosotros requerimos un filtro paso altas. Por lo tanto, solo se requiere invertir el orden de los vectores obtenidos para obtener el filtro solicitado. Además, se requiere desnormalizar dichos

[1] http://www.mathworks.com/help/signal/ref/cheb1ap.html

coeficientes para obtenerlos en la frecuencia en Hertz deseada. El proceso completo, así como el código puede ser consultado a detalle en la sección de anexos.

$$H(s) = \frac{z(s)}{p(s)} = \frac{k}{(s-p(1))(s-p(2))...(s-p(n))}. \tag{3.6}$$

En la sección de Anexo, podrá encontrar el código específico para diseñar el filtro propuesto. Con dicho còdigo, fue posible obtener los siguientes coeficientes:

```
Z =
     []

P =
  -0.1049 + 0.9580i
  -0.2532 + 0.3968i
  -0.2532 - 0.3968i
  -0.1049 - 0.9580i

K =
    0.1634

B2 =
  1.0e-012 *
    0.1634         0         0         0         0

A2 =
    0.0000    0.0000    0.0000    0.0007    1.0000
```

Con lo cual, la función de transferencia en el dominio de s puede espresarse de la siguiente forma:

$$H(s) = \frac{1{,}64 \times 10^{-13} s^4}{2{,}0580 \times 10^{-13} + 5{,}1680 \times 10^{-10} s + 1{,}2565 \times 10^{-6} s^2 + 7{,}1620 \times 10^{-4} s^3 + s^4} \tag{3.7}$$

# Capítulo 4

# Introducción a los Sistemas Multitasa

La característica común más importante en los filtros y los bancos de filtros usados en este escrito, es que usan procesamiento digital multitasa de señales. Para poder entender los sistemas multitasa, es esencial, primero, entender como el muestreo puede ser cambiado.

## 4.1. Reducción de la Tasa de Muestreo

La reducción de la tasa de muestreo de una señal se denomina *Decimación de la Tasa de Muestreo* o simplemente *Decimación*. Esta consiste en dos etapas: La primera se conforma por un filtro anti-aliasing y la segunda realiza la decimación o submuestreo. [(13)]

### 4.1.1. Submuestreo

El submuestreo es el proceso en el cual la tasa de muestreo de una señal discreta *x[n]* es reducida por un factor M. Lo anterior se logra tomando en cuenta solamente cada *M-ésimo* valor de la señal. La relación entre la señal resultante *y[m]* y la señal original esta dada por:

$$y[m] = x(m \cdot M). \tag{4.1}$$

La figura 4.1 muestra la representación del flujo de una señal durante el submuestreo. El símbolo cuadrado con una flecha apuntando hacia abajo, se denomina Submuestreador. La señal *y[m]* es la señal submuestreada con respecto a la señal *x[n]*.
En el dominio de $z$, se puede utilizar la transformada $z$ de la señal original

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n} \tag{4.2}$$


Figura 4.1: Proceso de Submuestreo

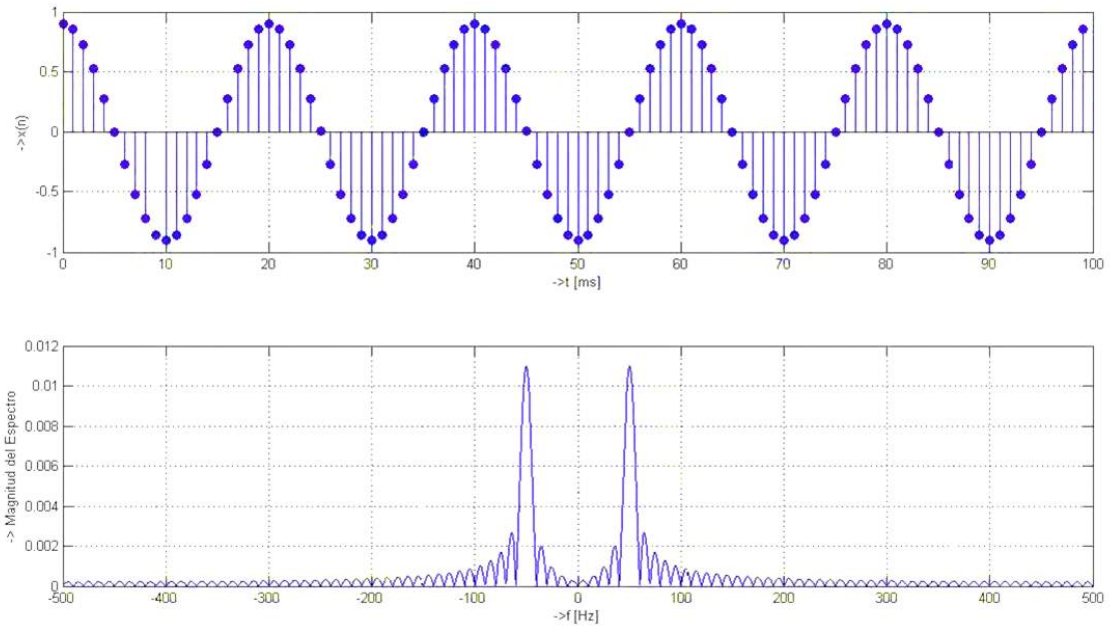


Figura 4.2: Señal Original

para obtener la tranformada $z$ de la señal y[m], considerando también que la *fase offset*, $\lambda$ , es igual a cero, es decir, durante el submuestreo, la primera muestra que se tomó como válida fue x[0]:

$$X_0^{(p)} = \sum_{n=-\infty}^{\infty} x(m \cdot M) \cdot z^{-mM} X_0^{(p)} = \sum_{n=-\infty}^{\infty} y(m) \cdot (z^M)^{-m} X_0^{(p)} = Y(z^M) = Y(z') \tag{4.3}$$

La transformada $z$ de la señal muestreada $Y(z^M)$, puede ser expresada usando componentes de modulación, para lo que es necesario definir dicho término en el contexto de la tranformada z.
La modulación de una transformada $z$ es realizada por la multiplicación de la variable independiente $z$ con el número $W_M^k$. Para el caso de $Y(z^M)$ se tiene que:

$$Y(z^M) = \frac{1}{M} \sum_{k=0}^{M-1} X(zW_M^k) \tag{4.4}$$

## 4.2. Aumento de la Tasa de Muestreo

Si algunas señales de banda angosta son combinadas para formar una señal de banda ancha, su tasa de muestreo debe primero ser aumentada. La tasa de muestreo también necesita ser incrementada cuando una señal de banda angosta será observada con una resolución muy fina en dominio del tiempo. El aumento en la tasa de muestreo es denominado *Interpolación*, y consiste de un sobremuestreo seguido de un *filtro anti-imagen* o *interpolador*. [(14)]

### 4.2.1. Sobremuestreo

La tasa de muestreo de una señal discreta $y(m)$ es aumentada por un factor $L$, colocando $L-1$ ceros igualmente espaciados entre cada par de muestras. La señal resultante $u(n)$ esta dada por:

$$u(n) = \begin{cases} y(n/L) & \text{para } n = mL, \quad \text{m entero,} \\ 0 & \text{otro caso.} \end{cases} \tag{4.5}$$

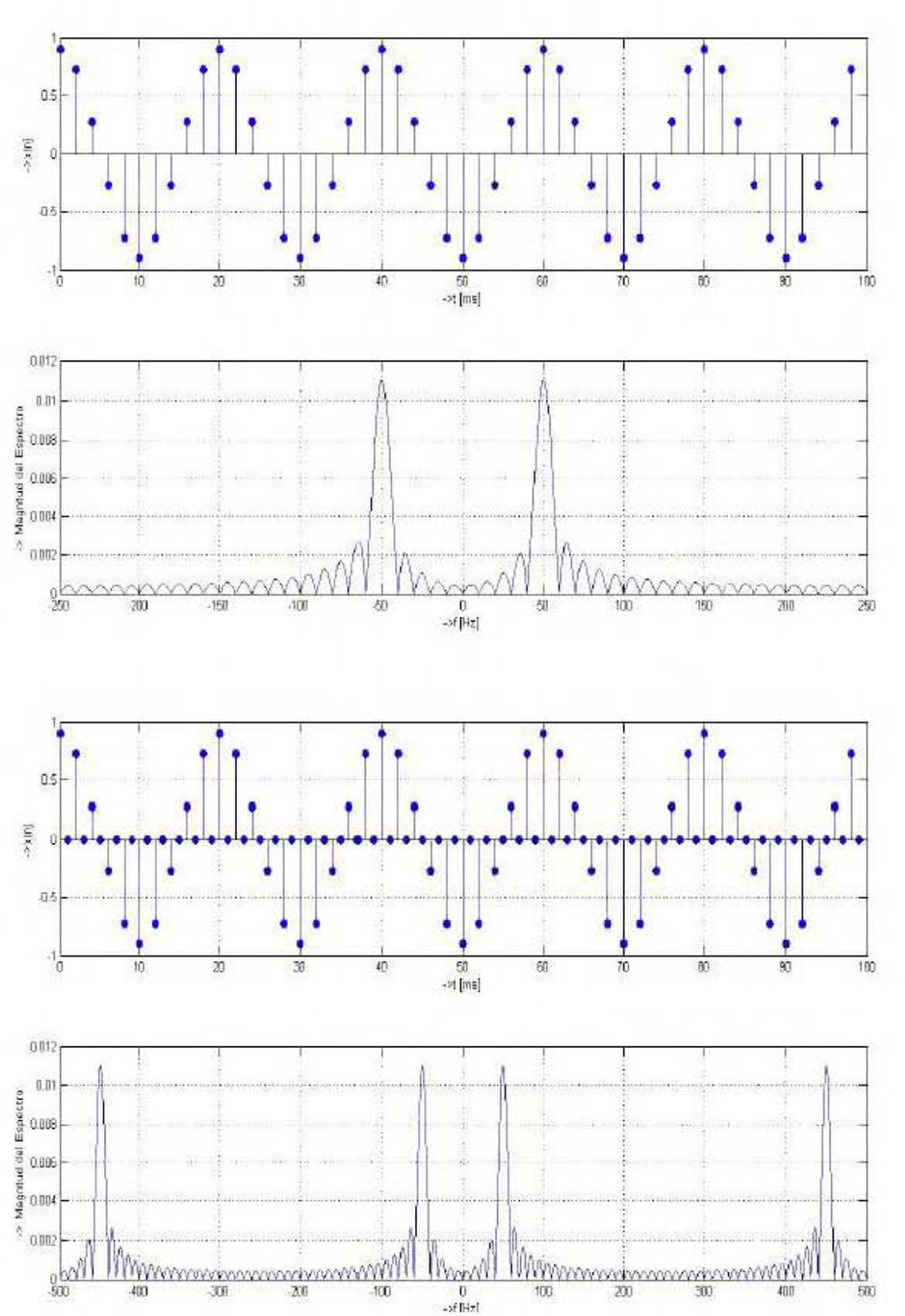


Figura 4.3: a)Señal submuestreada a partir de la señal en Figura 4.2 y b) Señal sobremuestreada a partir de la señal submuestreada.


Figura 4.4: Proceso de Sobremuestreo

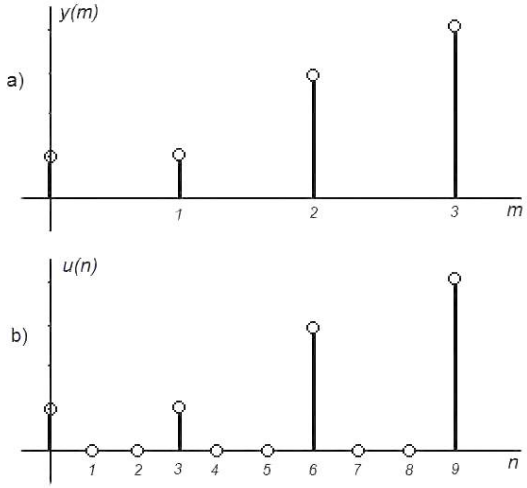


Figura 4.5: Sobremuestreo de una señal

El símbolo correspondiente al proceso de sobremuestreo se muestre en la figura 4.4
La figura 4.5a muestra una señal $y(m)$ y la figura 4.5b la señal obtenida después del sobremuestreo por un factor de 3.
El sobremuestreo es el proceso inverso al submuestreo, por lo tanto, si la transformada $z$:

$$Y(z) = \sum_{m=-\infty}^{\infty} y(m)z^{-m}$$

de la señal $y(m)$ antes del sobremuestreo es identificada con $Y(z)$ de la ecuación ( 4.3), y la transformada $z$ de la señal sobremuestreada $u(n)$ con $X_0^{(p)}(z^M$ de la ecuación ( 4.3), se puede utilizar esta misma ecuación para deducir lo siguiente:

$$U(z) = Y(z^L) \tag{4.6}$$

Un efecto muy importante que se presenta en la frecuencia durante el proceso de sobremuestreo, es el denominado formación de imágenes. Si un sobremuestreo es realizado con un factor $L$, en el dominio de la frecuencia se observará $L$ veces el espectro de la señal original. Este efecto recibe su nombre debido a que se obtiene, en el espectro de salida, una imagen adicional del espectro de entrada. Para el caso general, un aumento en la tasa de muestreo con un factor $L$ provocará $L-1$ imágenes adicionales del espectro de entrada. Un filtro pasobajas remueve las $L-1$ imágenes y, en efecto, rellena las muestras valuadas en cero con muestras con valores interpolados. [(15)] Como ejemplo de decimación e interpolación en conjunto, considere la señal que se muestra en la figura 4.2. Después de un submuestreo la señal resultante es la que se muestra en la figura 4.3a, note que en comparación con la señal la única diferencia visible es la frecuencia de muestreo. Si a la señal submuestreada, (figura 4.3), se le aplica un sobremuestreo, se obtiene la señal mostrada en la figura 4.3b. En el espectro de la señal sobremuestrada se observa la formación de frecuencias imágenes de las cuales ya se hizo mención en el párrafo anterior.

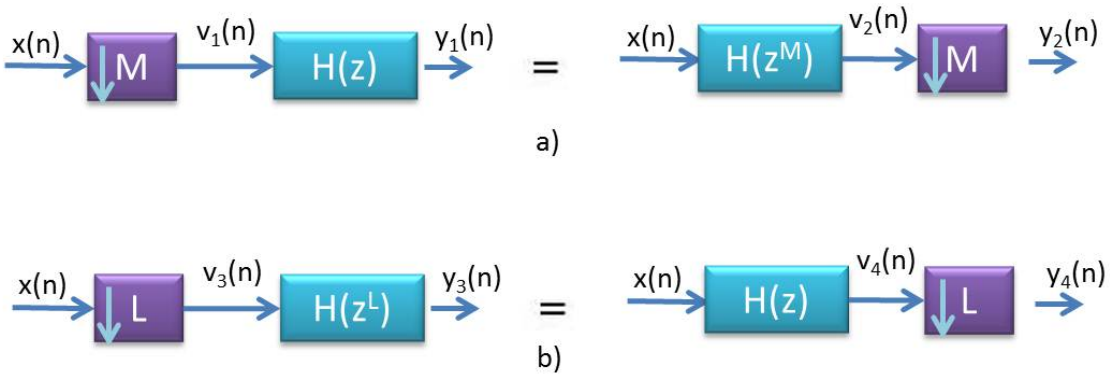


Figura 4.6: Equivalencias en Cascada

## 4.3. Equivalencias

Como se verá más adelante, un sistema multitasa esta conformado por interconexiones de dispositivos que alteran la tasa de muestreo, en este caso submuestreadores y sobremuestreadores. En muchas aplicaciones estos dispositivos aparecen en un estructura de cascada. A menudo se intercambian las posiciones de los dispositivos para obtener una mejor eficiencia computacional. [(16)]

### 4.3.1. Cascada de Sobremuestreador y Submuestreador

El sobremuestreador o el submuestreador pueden ser usados para cambiar la tasa de muestreo de una señal solamente por un factor entero. Si se desea implementar un cambio de tasa de muestreo con un factor fraccionario, es común usar un submuestreador y un sobremuestreador en cascada. Dichos dispositivos, de factor $M$ y $L$, respectivamente, pueden ser intercambiados sin ocasionar ningún cambio en la relación entrada-salida, si y sólo si $M$ y $L$ son primos relativos, es decir, $M$ y $L$ no deben tener un factor común que sea entero mayor a 1.

### 4.3.2. Identidades Nobles

Otras dos equivalencias muy simples pero muy utilizadas en el procesamiento multitasa son las mostradas en la figura 4.6. Estas identidades nobles nos permiten mover los dispositivos de alteración de tasa de muestreo para así obtener una posición mas ventajosa en cuestión de eficiencia.

## 4.4. Requerimientos Computacionales

Los filtros decimadores e interpoladores pueden ser diseñados con filtros digitales IIR o FIR. En el caso de procesamiento digital de señales con tasas simples, los filtros IIR en el aspecto computacional, son en general, más eficientes que los filtros FIR. La situación no es exactamente la misma en el caso del procesamiento multitasa, un ejemplo sencillo es el de suponer un filtro decimador de longitud $N$. Si la implementación se realiza con un filtro FIR en la forma directa, entonces

$$v(n) = \sum_{m=0}^{N-1} h(m)x(n-m)$$

Como el submuestreador conserva solo cada M-ésima muestra de $v(n)$ en su salida, es suficiente calcular $v(n)$ usando la ecuación anterior sólo para valores de $n$ que sean múltiplo de $M$ y saltar las muestra restantes. Esto deja un ahorro computacional con un factor de $M$. Por otro lado, si se

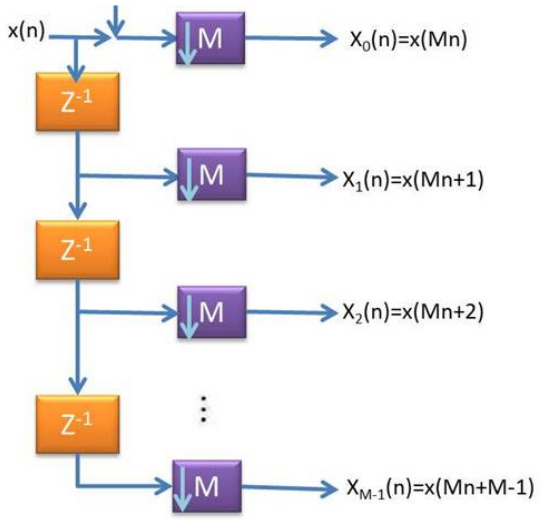


Figura 4.7: Una interpretación estructural de la descomposición polifase de M bandas de una secuencia $x(n)$

usa un filtro IIR, la complicación se nota inmediatamente al observar la forma de su función de transferencia, lo que ocasiona trabajar con una señal intermedia que para ser calculada necesita todos los valores de $n$. Para el caso de un filtro interpolador, los argumentos explicados anteriormente se mantienen. Si la función de tranferencia $H(z)$ del filtro se implementa con filtros FIR, entonces el ahorro en cálculos es por un factor de $L$ ya que, en este caso, $v(n)$ tiene $L-1$ ceros entre dos valores consecutivos que son diferentes de cero.

## 4.5. Descomposición Polifase

Como se vió en la sección anterior, un decimador o interpolador empleando filtros FIR pasobajas puede ser eficiente computacionalmente, adicionalmente a esto, si los filtros FIR se realizan usando la descomposición polifase, se puede obtener una reducción mayor de complejidad. A continuación explicaremos la descomposición polifase de forma general, incluyendo su aplicación en la eficiente realización del interpolador y decimador.

### 4.5.1. Proceso de descomposición

Si se considera una secuencia arbitraria $x(z)$ con transformada $z$:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

se puede escribir $X(z)$ como

$$X(z) = \sum_{k=0}^{M-1} z^{-1} X_k(z^M) \tag{4.7}$$

donde:

$$X_k(z) = \sum_{n=-\infty}^{\infty} x_k(n) z^{-n} = \sum_{n=-\infty}^{\infty} x(Mn+k) z^{-n}$$

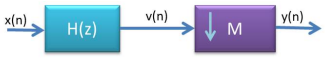


Figura 4.8: Decimador o Submuestrador

$$0 \leq k \leq M-1$$

Las subsecuencias $x_k(n)$ se llaman componentes polifase de la secuencia madre $x(n)$, y las funciones $X_k(z)$, dadas por la transformada $z$ de $x_k(n)$, se denominan componentes polifase de $X(z)$. La relación entre las subsecuencias $x_k(n)$ y la secuencia original $x(n)$ està dada por:

$$x_k(n) = x(Mn+k), 0 \leq k \leq M-1$$

La ecuación ( 4.7) se puede reescribir en la forma matricial como:

$$X(z) = \begin{bmatrix} 1 & z^{-1} & \dots & z^{-(M-1)} \end{bmatrix} \begin{bmatrix} X_0(z^M) \\ X_1(z^M) \\ \vdots \\ X_{M-1}(z^M) \end{bmatrix}$$

Una interpretación multitasa estructural de la descomposición polifase se muestra en la figura 4.7

### 4.5.2. Estructuras del decimador e interpolador eficientes computacionalmente

Es posible obtener estructuras de interpolador y decimador eficientes computacionalmente aplicando una descomposición polifase a su función de transferencia correspondiente.
Considerando la aplicación de la descomposición polifase en el filtro de decimación de la figura 4.8, la estructura completa del decimador es la forma que se muestra en la figura 4.9a. Una realización equivalente es la mostrada en la figura 4.9b, obtenida usando la equivalencia de cascada de la figura 4.6a. Esta última realización es más eficiente computacionalmente que la estructura de la figura 4.9a; para ilustrar este hecho es necesario asumir que el filtro de decimación $H(z)$ es un filtro FIR de longitud N con un periodo de muestreo de entrada $T = 1$. Ya que la salida $y(n)$ del decimador es obtenida submuestreando la salida del filtro $v(n)$ por un factor de $M$, es necesario sólo calcular $v(n)$ en $n = \dots, -2M, -Mt, 0, M, 2M, \dots$. Los requerimientos de cálculo son entonces $N$ multiplicaciones y $(N-1)$ sumas por muestra de salida calculada. Sin embargo, si $n$ crece, las señales almacenadas en los elementos de retardos cambian, como resultado de esto, todos los cálculos necesitan ser completados en un periodo de muestreo, mientras que para los siguientes $(M-1)$ periodos, las unidades aritméticas se mantienen sin actividad. (17) Ahora, si se considera la estructura de la figura 4.9b y si la longitud de los subfiltros $E_k(z)$ es $N_k$, entonces $N = \sum_{k=0}^{M-1}$. Los requerimientos de cálculo para el k-ésimo subfiltro son $N_k$multiplicaciones y $N_k - 1$ sumas por muestra de salida, por lo tanto, la estructura completa tiene entonces $\sum_{k=0}^{-1} N_k = N$ multiplicaciones y $\sum_{k=0}^{-1}(N_k - 1) + (M-1) = N-1$ sumas por muestra de salida en el decimador. La ventaja en esta estructura es que la unidades aritméticas funcionan en cada periodo de muestreo de salida, lo que significa $M$ veces que el periodo de muestreo de entrada.

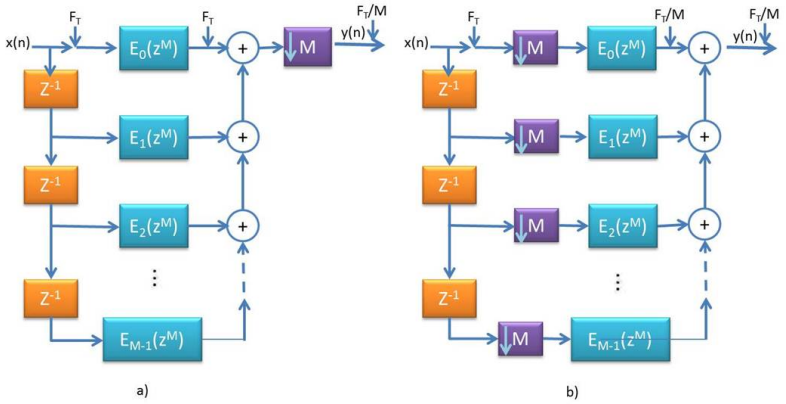


Figura 4.9: a) Implementación de un decimador basado en una descomposición polifase tipo I y b)estructura del decimador eficiente computacionalmente. En las figuras se muestran las tasas de muestreo.

# Capítulo 5

# Transformada Z mediante el Algoritmo de la Matriz de Pascal y sus aplicaciones en el diseño de Filtros IIR

Obtener la conversión de funciones de transferencia es un proceso que en ocasiones se torna incómodo y un tanto largo. Existen aplicaciones (como MATLAB o MAPLE) que tienen funciones predeterminadas para hacer este tipo de operaciones, sin embargo suelen ser caras o bien en términos computacionales suelen ser muy costosas para los microprocesadores. Una alternativa a este tipo de problemas es emplear un método conocido como "Tranformación a Dominio Z por medio del Uso de la Matriz de Pascal", el cual será ampliamente explicado a continuación

## 5.1. La Matríz de Pascal y su Aplicación en la obtención de la función de transferencia en el dominio de Z

Un gran número de procedimientos están disponibles para el diseño de filtros digitales, muchas de esas transformaciones ayudan a convertir filtros analógicos a filtros digitales equivalentes. La función de transferencia de un filtro analógico está representada por la siguiente ecuación:

$$H(s) = \frac{B_0 + B_1 s + B_2 s^2 + \ldots + B_m s^m}{A_0 + A_1 s + A_2 s^2 + \ldots + A_m s^m} \tag{5.1}$$

De la ecuaciòn ( 5.1) se pueden representar los vectores:

$$B = [B_0 + B_1 + B_2 + \ldots + B_m] \tag{5.2}$$

$$A = [A_0 + A_1 + A_2 + \ldots + A_m] \tag{5.3}$$

Donde $B_i$ y $A_i$ son coeficientes reales.
Por otro lado, la función de transferencia que caracteriza a un filtro digital en el dominio de z es:

$$H(z) = \frac{b_0 + b_1 z + b_2 z^2 + \ldots + b_n z^n}{a_0 + a_1 z + a_2 z^2 + \ldots + a_n s^n} \tag{5.4}$$

De igual forma se puede representar los vectores:

$$b = [b_0 + b_1 + b_2 + \ldots + b_n] \tag{5.5}$$

$$a = [a_0 + a_1 + a_2 + \ldots + a_n] \tag{5.6}$$

Con coeficientes reales $a_i$ y $b_i$

## 5.2. Transformación Pasobajas-Pasobajas

Para filtros paso bajas la función de transferencia digital $H(z)$ puede ser obtenida del prototipo en tiempo real (ver sección 5.1) usando la transformada bilineal:

$$s = C\frac{1 - Z^{-1}}{1 + Z^{-1}} \tag{5.7}$$

Donde:

$$C = cot(\frac{\pi f_c}{f_m}) \tag{5.8}$$

Y las constantes $f_c$ y $f_m$ representan la frecuencia de corte y la frecuencia de muestreo respectivamente.
Si se considera $n = 2$, se puede obtener una relación matricial entre los vectores ( 5.2), ( 5.3) y ( 5.5),( 5.6) aplicando la transformación ( 5.7) en ( 5.1) para generar ( 5.4). La relación encontrada es la siguiente:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1C \\ A_2C^2 \end{bmatrix} \tag{5.9}$$

De la misma manera, se puede obtener una matríz similar para los coeficientes de $b_i$:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1C \\ B_2C^2 \end{bmatrix} \tag{5.10}$$

En este caso, de las ecuaciones matriciales ( 5.9) y ( 5.10), se obtiene lo siguiente:

$$a = P_{LP}^n A' \tag{5.11}$$

Donde $P_{LP}$ es la matríz paso-bajas y los vectores A' y B' están representados por:

$$B' = [B_0, B_1C, B_2C^2, \ldots, B_mC^m] \tag{5.12}$$

$$A' = [A_0, A_1C, A_2C^2, \ldots, A_mC^m] \tag{5.13}$$

Como queda evidenciado, el càlculo de la matríz $P_{LP}^n$ se puede realizar de una forma sistemática. Para ello se considera un triángulo de Pascal (Figura 5.1).
Observe que los coeficientes del renglón n = 2 crean la última columna de la matriz de Pascal paso-bajas ( 5.9) y ( 5.10) con excepción de que el renglón par es negativo. Así de esta forma se puede generar un procedimiento para encontrar los valores de la matriz de Pascal. [(18)]

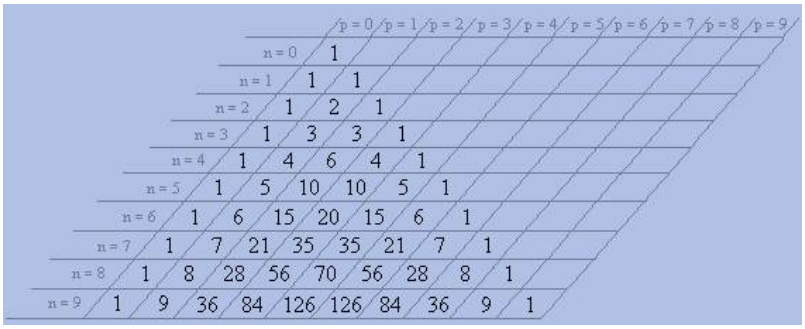


Figura 5.1: Triángulo de Pascal

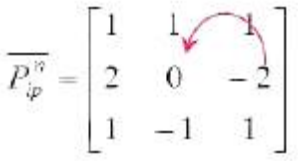

Figura 5.2: Descripción gráfica para calcular los elementos de la matríz $P_{LP}^{n}$

### 5.2.1. Cálculo Sistemático de los valores de la matríz $P_{LP}^{n}$

PASO 1: En el primer renglón de la matriz de Pascal, todos los elementos deben ser igual a uno. Si n=2

$$P_{LP}^{n} = \begin{bmatrix} 1 & 1 & 1 \\ & & \end{bmatrix} \tag{5.14}$$

PASO 2:
Los elementos de la última columna pueden ser calculados de la siguiente forma:

$$p_{i,m+1} = (-1)^{i-1} \frac{n!}{(n-i+1)!(i-1)!} \tag{5.15}$$

donde $i = 1, 2, \ldots, n+1$
para nuestro ejemplo, este es el resultado:

$$P_{LP}^{n} = \begin{bmatrix} 1 & 1 & 1 \\ & & -2 \\ & & 1 \end{bmatrix} \tag{5.16}$$

PASO 3
Los elementos restantes, $p_{i,j}$, se pueden obtener usando la siguiente ecuación:

$$p_{i,j} = p_{i-1,j} + p_{i-1,j+1} + p_{i,j+1} \tag{5.17}$$

donde $i = 2, \ldots, n+1; j = n, n-1, n-2, \ldots, 2, 1$

## 5.3. Transformación Pasobajas-Pasoaltas

En este segundo caso, para transformar la función de transferencia paso bajas a una función de transferencia digital paso altas $H(z)$, se sustituye la variable $s$ por $1/s$, de esta forma se obtiene la transformación bilineal de la siguiente forma:

$$s = K\frac{1+Z^{-1}}{1-Z^{-1}} \tag{5.18}$$

con:

$$K = tan(\frac{\pi f_c}{f_m}) \tag{5.19}$$

Donde $f_c$ representa la frecuencia de corte paso altas y $f_m$ representa la frecuencia de muestreo. Siguiendo el mismo procedimiento para n=2 se obtiene:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -2 & 0 & 2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1K \\ A_2K^2 \end{bmatrix} \tag{5.20}$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -2 & 0 & 2 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1K \\ B_2K^2 \end{bmatrix} \tag{5.21}$$

Por lo que se pueden escribir las ecuaciones ( 5.20) y ( 5.21) como:

$$a = P^n_{HP} x A'' \qquad\qquad b = P^n_{HP} x B'' \tag{5.22}$$

Donde $P^n_{HP}$ es la variante de la matrìz de Pascal que corresponde al filtro paso-altas.

### 5.3.1. Cálculo Sistemático de los valores de la matríz $P^n_{HP}$

PASO 1:
En el primer renglón de la matriz de Pascal, todos los elementos deben ser igual a uno.
Si n=2

$$P^n_{LP} = \begin{bmatrix} 1 & 1 & 1 \\ & & \end{bmatrix} \tag{5.23}$$

PASO 2:
Los elementos de la primera columna pueden ser calculados como lo indica la ecuación ( 5.15):
donde $i = 1, 2, \ldots, n+1$
para nuestro ejemplo, este es el resultado:

$$P^n_{HP} = \begin{bmatrix} 1 & 1 & 1 \\ -2 & & \\ 1 & & \end{bmatrix} \tag{5.24}$$

PASO 3
Los elementos restantes, $p_{i,j}$, se pueden obtener usando la siguiente ecuación:

$$p_{i,j} = p_{i,j-1} + p_{i-1,j-1} + p_{i-1,j} \tag{5.25}$$

donde $i = 2, \ldots, n+1; j = 2, \ldots, n+1$

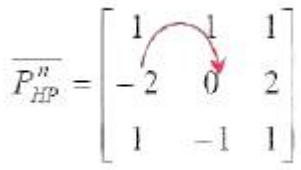

Figura 5.3: Descripción gráfica para calcular los elementos de la matríz $P^n_{HP}$

## 5.4. Transformación Pasobajas-Pasobanda

El filtro paso Banda se puede obtener con la superposición de un filtro paso-bajas y un filtro paso-altas, así la transformación $s$ a $z$ queda de la siguiente forma:

$$s = \left[C\frac{1-Z^{-1}}{1+Z^{-1}} + K\frac{1+Z^{-1}}{1-Z^{-1}}\right] m \tag{5.26}$$

donde:

$$C = cot(\frac{\pi f_1}{f_m}), \qquad K = tan(\frac{\pi f_{-1}}{f_m}), \qquad m = \frac{cot(\frac{\pi(f_1 - f_{-1})}{f_m})}{k + c} \tag{5.27}$$

Donde $f_1$ y $f_{-1}$ representa la frecuencia de corte superior y la frecuencia de corte inferior respectivamente paso altas y $f_m$ representa la frecuencia de muestreo. De la misma forma que en los procedimientos pasados se pueden encontrar los vectores $a$ y $b$ a partir de los vectores de coeficientes de la función de transferencia analógica A y B. Por ejemplo, para n = 2 (analógico) se obtendrá un n = 4(digital) de la siguiente forma:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -2 & 0 & 2 & 4 & 0 \\ 6 & 0 & -2 & 0 & 6 & -2 \\ -4 & 2 & 0 & -2 & 4 & 0 \\ 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} A_2C^2m^2 \\ A_1Cm \\ A_0 \\ A_1Km \\ A_2K^2m^2 \\ 2A_2CKm^2 \end{bmatrix} \tag{5.28}$$

La matríz $P^n_{PB}$ no es cuadrada, entonces no es posible calcular matriz inversa y hacer la transformación $z$ —— $s$. Por lo que la ecuación ( 5.28) se puede escribir como:

$$a = P^n_{LP} \cdot A'' \tag{5.29}$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -4 & -2 & 0 & 2 & 4 \\ 6 & 0 & -2 & 0 & 6 \\ -4 & 2 & 0 & -2 & 4 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} A_2C^2m^2 \\ A_1Cm \\ A_0 + 2A_2CKm^2 \\ A_1Km \\ A_2K^2m^2 \end{bmatrix} \tag{5.30}$$

La ventaja de esta representación es que la matriz es cuadrada y se puede calcular la matriz inversa:

$$(P^n_{BP})^{-1} = \frac{1}{n} P^n_{BP} \tag{5.31}$$

De igual manera $P^n_{BP}$ se puede obtener con la concatenación de dos matrices como se muestra en la ecuación [( 5.32)]:

$$\left[P^n_{BP}\right] = \left[S^n_{BP}\right]\left[R^n_{BP}\right] \tag{5.32}$$

Donde $S^n_{BP}$ es una matriz cuadrada y es calculada exactamente de la misma forma que en la transformación de paso-bajas a paso-altas. Por otro lado la matríz $R^n_{BP}$ es una matriz rectangular con n+1 renglones. A priori, el número de columnas han sido calculados por medio de contar el número de elementos diferentes de 1, incluído en el triángulo superior de Pascal. Por ejemplo, la siguiente tabla muestra el número de columnas de la matriz $R^n_{BP}$ para diferentes valores de $m$ y $n$. Por ejemplo, para m = 2 y n = 4 se tiene que $R^n_{BP}$ tiene una columna, que es precisamente la columna central de $S^n_{BP}$. Otro ejemplo es para m = 3 y n = 6, los números diferentes de uno en el triángulo de Pascal para n = 6 son tres, por lo que el número de columnas en $R^n_{BP}$ es 3, que son la columna central (pivote) de $S^n_{BP}$, la siguiente es la columna contigua que se encuentra a la derecha del pivote y finalmente la última columna es la que se encuentra contigua al pivote pero del lado izquierdo. Por ejemplo para obtener los coeficientes $a$ del filtro digital paso banda con m = 3 y n = 6, se obtiene la ecuación matricial ( 5.33).

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -6 & -4 & -2 & 6 & 2 & 4 & 6 \\ 15 & 5 & -1 & -3 & -1 & 5 & 15 \\ -20 & 0 & 4 & 0 & -4 & 0 & 20 \\ 15 & -5 & -1 & 3 & -1 & -5 & 15 \\ -6 & 4 & -2 & 0 & 2 & -4 & 6 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} A_3C^3m^3 \\ A_2C^2m^2 \\ A_1Cm + 3A_3C^2Km^3 \\ A_0 + 2A_2CKm^2 \\ A_1Km + 3A_3CK^2m^3 \\ A_2K^2m^2 \\ A_3K^3m^3 \end{bmatrix} \tag{5.33}$$

Tabla 5.1: Número de columnas de la matriz $R^n_{BP}$

| M | N | Col |
|---|---|---|
| 2 | 4 | 1 |
| 3 | 6 | 3 |
| 4 | 8 | 6 |

Para el caso del vector b, el procedimiento se realiza de igual manera.
Donde:

$$a = P^n_{LP} \cdot A''A'' = (P^n_{LP})^{-1} = \frac{1}{n} P^n_{LP} A'' \tag{5.34}$$

De esta forma se calcula la transformación inversa $z$ ——- $s$. Si se conoce $H(z)$, mediante la ecuación ( 5.34) se calcula $H(s)$

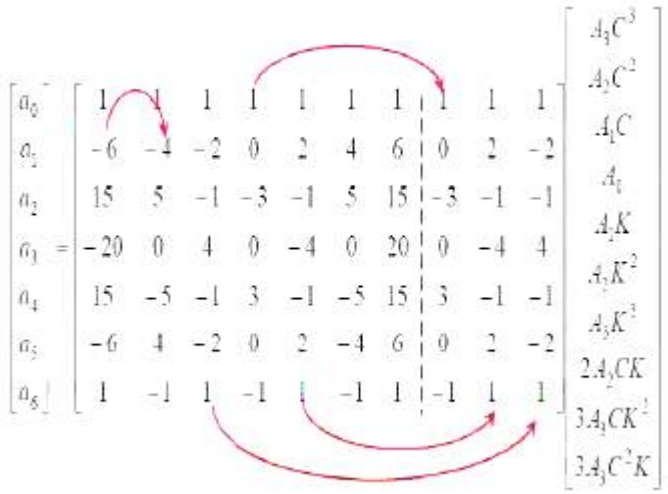

Figura 5.4: Descripción gráfica para calcular los elementos de la matríz $P_{BP}^{n}$

# Capítulo 6

# Uso de la Matríz de Pascal para la Codificación de Señales de Voz

Teniendo ya conocimiento de los Capítulos anteriores, es posible diseñar un Codificador de Señales de Audio eficiente, aplicando un Banco de Filtros aplicando la técnica de Matriz de Pascal antes expuesta y codificando mediante técnicas de Multitasa

## 6.1. Algoritmo de la Matriz de Pascal para su aplicación en Condificación de Señales de Voz

Para este caso en específico, utilizaremos un Banco de filtros de 5 canales, recordando que entre mas canales tenga un filtro, mayor será su eficacia pero también aumentará su tiempo de procesamiento. Aunque para el propósito de esta tesis se diseñó de la forma descrita, es necesario mencionar que este método puede realizarse de igual forma para $n$ número de canales.
Sea la Matriz de Pascal para n=5 calculada mediante la técnica explicada en el Capítulo 5:

$$H_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 4 & 2 & 0 & -2 & -4 \\ 6 & 0 & -2 & 0 & 6 \\ 4 & -2 & 0 & 2 & -4 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} \tag{6.1}$$

Podemos obtener un Banco de Filtros conformado de acuerdo a la Figura 6.1, que se basa en la técnica Polifase estudiada en la Sección 4.5, en donde se emplea un decimador y un interpolador para cada canal de orden 5, y una Matriz de Admitancias G que estará formada por la Matriz transpuesta $H(z)$
Entonces, de acuerdo con lo anterior, la matriz de admitancias está dada por:

$$G_5 = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ 1 & 0 & -2 & 0 & 1 \\ -1 & 2 & 0 & -2 & 1 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} \tag{6.2}$$

De las matrices 6.1 y 6.2, podemos obtener los siguientes arreglos:

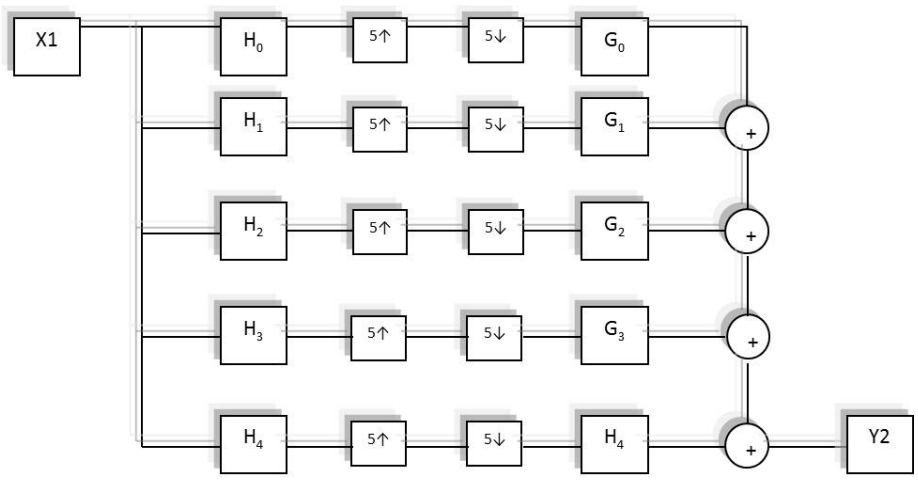


Figura 6.1: Banco de Filtros empleando la Matríz de Pascal

Para los coeficientes de H en cada banco de filtros de la figura 6.1:

$$\begin{aligned} H_0 &= 1 + z^{-1} + z^{-2} + z^{-3} + z^{-4} \\ H_1 &= 4 + 2z^{-1} - 2z^{-3} - 4z^{-4} \\ H_2 &= 6 - 2z^{-2} + 6z^{-4} \\ H_3 &= 4 - 2z^{-1} + 2z^{-3} - 4z^{-4} \\ H_4 &= 1 - z^{-1} + z^{-2} - z^{-3} + z^{-4} \end{aligned} \tag{6.3}$$

Para los coeficientes de G en cada banco de filtros de la figura 6.1, se debe dividir entre $2^n$, en nuestro caso como n=5, el valor es 16, quedando el siguiente arreglo:

$$\begin{aligned} G_0 &= 0{,}0625 + 0{,}25z^{-1} + 0{,}375z^{-2} + 0{,}25z^{-3} + 0{,}0625z^{-4} \\ G_1 &= -0{,}0625 - 0{,}125z^{-1} + 0{,}125z^{-3} + 0{,}06254z^{-4} \\ G_2 &= 0{,}0625 - 0{,}125z^{-2} + 0{,}0625z^{-4} \\ G_3 &= -0{,}0625 + 0{,}125z^{-1} - 0{,}125z^{-3} + 0{,}06254z^{-4} \\ G_4 &= 0{,}0625 - 0{,}25z^{-1} + 0{,}375z^{-2} - 0{,}25z^{-3} + 0{,}0625z^{-4} \end{aligned} \tag{6.4}$$

Y teniendo estos valores, se puede programar de manera sencilla mediante la plataforma MATLAB para analizar los resultados correspondientes.

## 6.2. Uso de Simulink de MATLAB como herramienta de simulación para la Codificación de Señales de Voz

Con base a lo desarrollado en la sección 6.1, podemos emplear la Plataforma de Simulink del Programa de MATLAB para realizar una simulación de lo expuesto anteriormente.
Considérese el siguiente filtro conformado en Simulink tal como se aprecia en la figura 6.2
En cada subsistema, se deberán colocar los valores correspondientes de los arreglos de $H(z)$ y de $G(z)$ calculados anteriormente. Para ejemplo de ello, ponemos el caso correspondiente a $H(0)$, el cual puede apreciarse en la Figura 6.3
Para los casos específicos de los decimadores y los interpoladores, se configuran tal como se puede pareciar en las figuras 6.4 y 6.5 respectivamente.

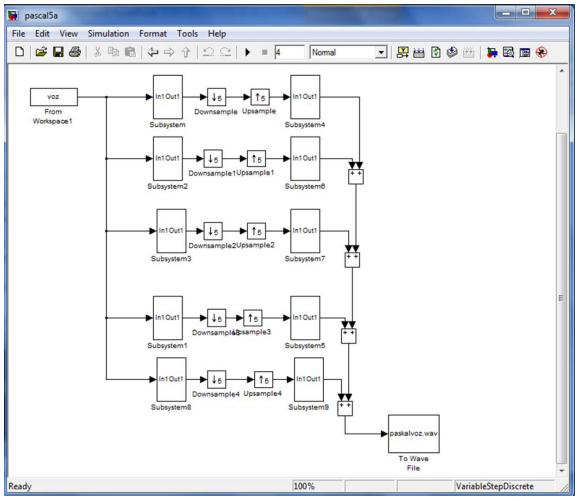


Figura 6.2: Banco de Filtros implementado en Simulink de MATLAB

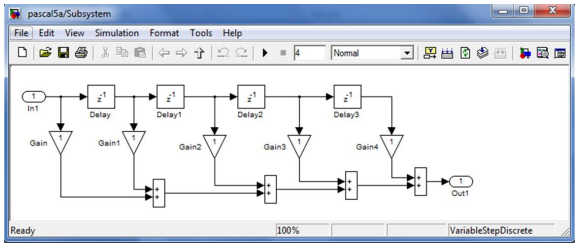


Figura 6.3: Subsistema de Filtrado para el caso de $H_0$

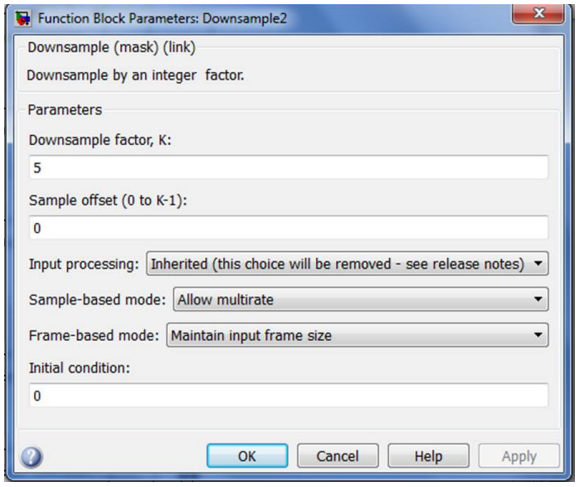


Figura 6.4: Configuración del Decimador

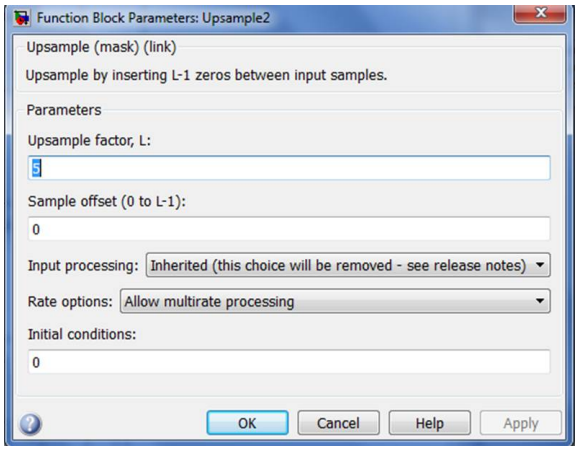


Figura 6.5: Configuración del Interpolador

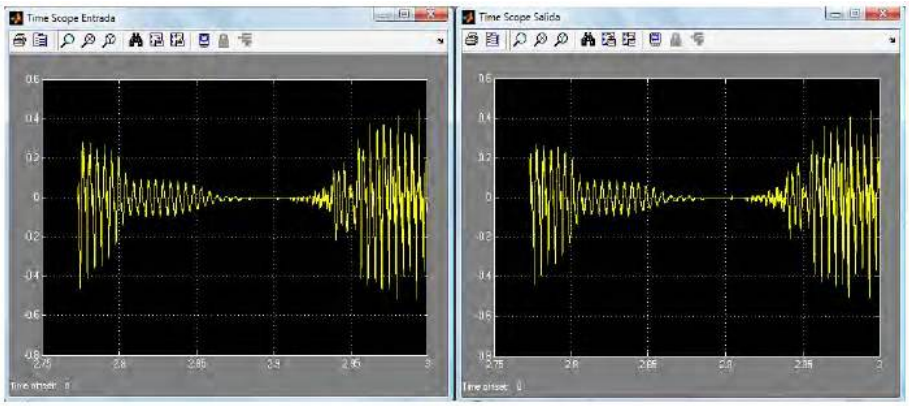


Figura 6.6: Señales de entrada y salida en el banco de filtros de 5 canales empleando la técnica de Matríz de Pascal en Simulink.

Como la señal que se pretende procesar es voz, es necesario cargar un archivo de audio en el ambiente de Simulink, lo cual se hace con el siguiente código descrito en el apéndice para Matlab. Este código permite leer un archivo de audio y convertir sus datos en una secuencia de datos que pueda ser entendida por Simulink.
Una vez cargado el archivo de audio necesario para la simulación, sólo falta iniciar la simulación propiamente. Es posible constatar si la simulación es un éxito de diferentes formas: con el bloque *To Wave Device* es posible escuchar la señal a la salida o la entrada del banco y así, de forma un tanto empírica, verificar el funcionamiento del banco; o bien, con el bloque *Time Scope* se puede verificar si las señales de entrada y de salida son parecidas en el tiempo; finalmente con un arreglo de bloques es posible comparar las señales entrada/salida en el dominio de la frecuencia, que finalmente es la mejor forma de observar si el banco recupera la señal de forma adecuada. A continuación, en la figura 6.6 se muestran los resultados obtenidos de la simulación del banco de filtros de 5 canales propuesto utilizando las dos ultimas herramientas descritas.
En ambas figuras, 6.6 y 6.7, se observa que las señales de entrada y de salida, en el tiempo y en la frecuencia, son practicamente iguales, lo que implica que el banco de filtros simulado presenta una reconstrucción perfecta.

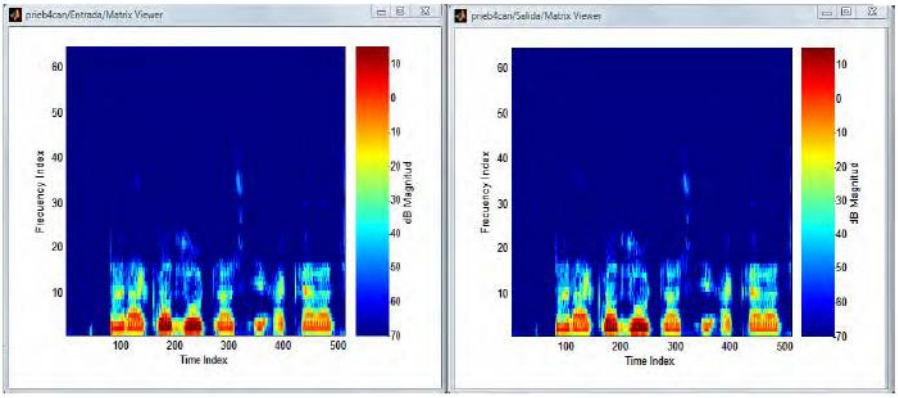


Figura 6.7: Espectrograma del banco de filtros de 5 canales en Simulink.

# Capítulo 7

# El DSP TMS320C6713

Existe una gran cantidad de aplicaciones dentro del procesamiento digital de señales, en áreas, tales como: radar, sonar, voz, comunicaciones, telefonía, medicina, control, sismología, imágenes, etc. Las herramientas que han permitido obtener soluciones reales y eficientes son el desarrollo de las tecnologías de programación y los microprocesadores de procesamiento digital de señales. En la actualidad se ha presentado un amplio crecimiento en la industria digital esto se debe en gran medida al desarrollo de algoritmos de procesamiento digital de señales. Dentro de muchas áreas se requiere tener aplicaciones, tales como filtrado, compresión, análisis en frecuencia, entre otras. El uso de sistemas digitales permite realizar estas tareas con una computadora digital o un microprocesador. En este capítulo se explicarán las características del microprocesador que se utilizará en el diseño del banco de filtros.

## 7.1. Características del Microprocesador TMS320C6713

El dispositivo TMS320C6713, ejecuta hasta 8 instrucciones, de 32 bits por ciclo de reloj. El CPU contiene 32 registros de propósito general, de 32 bits y 8 unidades funcionales. Este dispositivo tienen un conjunto completo de herramientas de desarrollo y optimización, que incluyen un compilador C eficiente, un optimizador de ensamble para simplificar la planificación y programación del lenguaje ensamblador, y un depurador con interfase gráfica, basada en Windows, para visualizar las características de ejecución en el código fuente. Además, contiene una tarjeta de emulación de hardware compatible con la interfase del emulador TI XDS510. Las características del dispositivo TMS320C6713, incluyen:

1. Un CPU avanzado VLWI (very long word instruction) con 8 unidades funcionales, que incluyen 2 multiplicadores y 6 ALU's (Unidades Lógico Aritméticas).
   *a*) Ejecuta un máximo de 8 instrucciones por ciclo, 8 veces más que los DSP's típicos.
   *b*) Permite rápido tiempo de desarrollo en diseños con código RISC altamente efectivos.
2. Empaquetado de instrucción
   *a*) Obtiene el tamaño del código equivalente a las 8 instrucciones ejecutadas serialmente o en paralelo.
   *b*) Reduce el tamaño del código y el consumo de energía
3. Ejecución condicional de todas las instrucciones.
   *a*) Reduce los Saltos costosos.
   *b*) Incrementa el paralelismo para mantener el alto desempeño
4. Ejecuta el código programado, en unidades funcionales independientes.

5. Proporciona soporte eficiente de memoria para una variedad de aplicaciones de 8, 16 y 32 bits de datos.
6. Maneja operaciones artiméticas de 40 bits, adicionando presición extra a vocoders y otras aplicaciones computacionalmente intensivas.
7. Proporciona soporte de normalización y saturación, en operaciones aritméticas claves.
8. Soporta operaciones comunes, halladas en aplicaciones de control y manipulación de datos, como manipulación de campos, extracción de instrucción, activación, desactivación y conteo de bits.

Además, el TMS320C6713 tiene las siguientes características:

- Un máximo de 1336 MIPS (millones de instrucciones por segundo), a 167 MHz.
- Un máximo de 1 G FLOPS (operaciones de punto flotante por segundo), a 167 MHz, para operaciones de precisión simple.
- Un máximo de 250 M FLOPS a 167 MHz, para operaciones de doble precisión.
- Un máximo de 688 M FLOPS a 167 MHz, para operaciones de multiplicación y acumulación.
- Soporte de Hardware para operaciones de punto flotante, de simple y doble presición.
- Multiplicación entera de 32x32 bits, con resultado de 32 o 64 bits.

El dispositivo TMS320C6713 tiene una gran variedad de opciones de memoria y periféricos tal como: Amplia memoria RAM para ejecución rápida de algoritmos, acceso a la memoria por medio del puerto host, controlador multicanal DMA, puertos serie multicanal y un timer de 32 bits.

## 7.2. Arquitectura del Microprocesador

El procesador TMS320C6713, consiste en tres partes: el CPU, los periféricos y la memoria. Además ocho unidades funcionales operan en paralelo (seis ALU's y dos multiplicadores), con dos conjuntos similares de cuatro unidades funcionales básicas. Las unidades se comunican usando un camino cruzado entre dos clasificaciones de registros, cada uno de los cuales contiene 16 registros de 32 bits. La figura 7.1 muestra el diagrama de bloques del dispositivos TMS320C6713.

### 7.2.1. Unidad de procesamiento Central (CPU)

El CPU contiene:

- Unidad fetch de programa
- Unidad de despacho de instrucción
- Unidad de decodificación de instrucción
- 32 registros de 32 bits
- Dos caminos de datos, cada uno con cuatro unidades funcionales
- Registros de control
- Lógica de control
- Lógica de interrupción

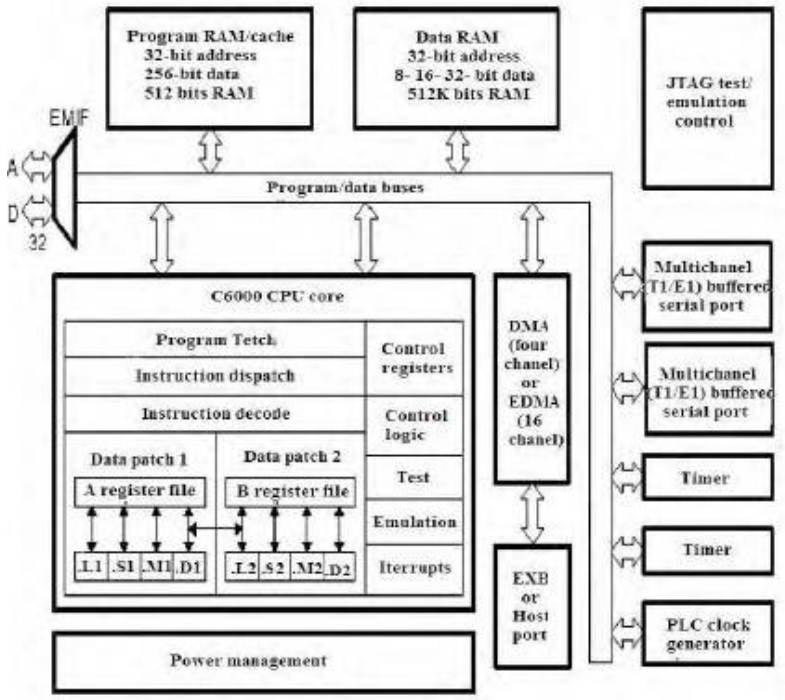


Figura 7.1: Diagrama de Bloques de TMS320C6713

El CPU tiene dos caminos de datos (A y B), cada camino tiene cuatro unidades funcionales (.L, .S, .M y .D) y un archivo de registros de 32 bits (register file). Las unidades funcionales ejecutan operaciones de lógica, corrimiento, multiplicación y direccionamiento de datos. Todas las instrucciones aceptan operaciones de carga y almacenamiento sobre los registros. Las dos unidades de direccionamiento de datos (.D1 y .D2) son exclusivamente responsables de toda la transferencia de datos entre los archivos de registros y la memoria.

### 7.2.2. Caminos de datos del CPU

Los caminos de datos del CPU consisten de: dos archivos de registros de propósito general (A y B), ocho unidades funcionales (.L1, .L2, .S1, .S2, .M1, .M2, .D1 y .D2), dos caminos de lectura de memoria (LD1 y LD2), dos caminos de almacenamiento en memoria (ST1 y ST2), dos caminos cruzados entre los archivos de registros(1X y 2X) y dos caminos de direccionamiento de datos (DA1 y DA2). La figura 7.2 muestra el camino de los datos del CPU.

### 7.2.3. Archivos de registros de propósito general (register files)

Hay dos archivos de registros de propósito general (A y B) en los caminos de datos del TMS320C6713. Cada uno de esos archivos contiene 16 registros de 32 bits (A0-A15) para el archivo A y (B0-B15) para el archivo B. Los registros de propósito general pueden ser usados para manejar datos o punteros de direccionamiento de estos. Los registros A1, A2, B0, B1 y B2 pueden ser utilizados como registros de condición. Los registros A4-A7 y B4-B7 pueden ser usados para el direccionamiento circular. Los archivos de registros de propósito general soportan datos de 32 y 40 bits de punto fijo. Los datos de 32 bits, pueden estar contenidos en cualquier registro de propósito general. Los datos de 40 bits están contenidos en dos registros; los 32 bits menos significativos del dato (LSB) son colocados en un registro par y los restantes ocho bits mas significativos (MSB) son colocados en los ocho bits menos significativos del registro próximo superior (que es siempre un

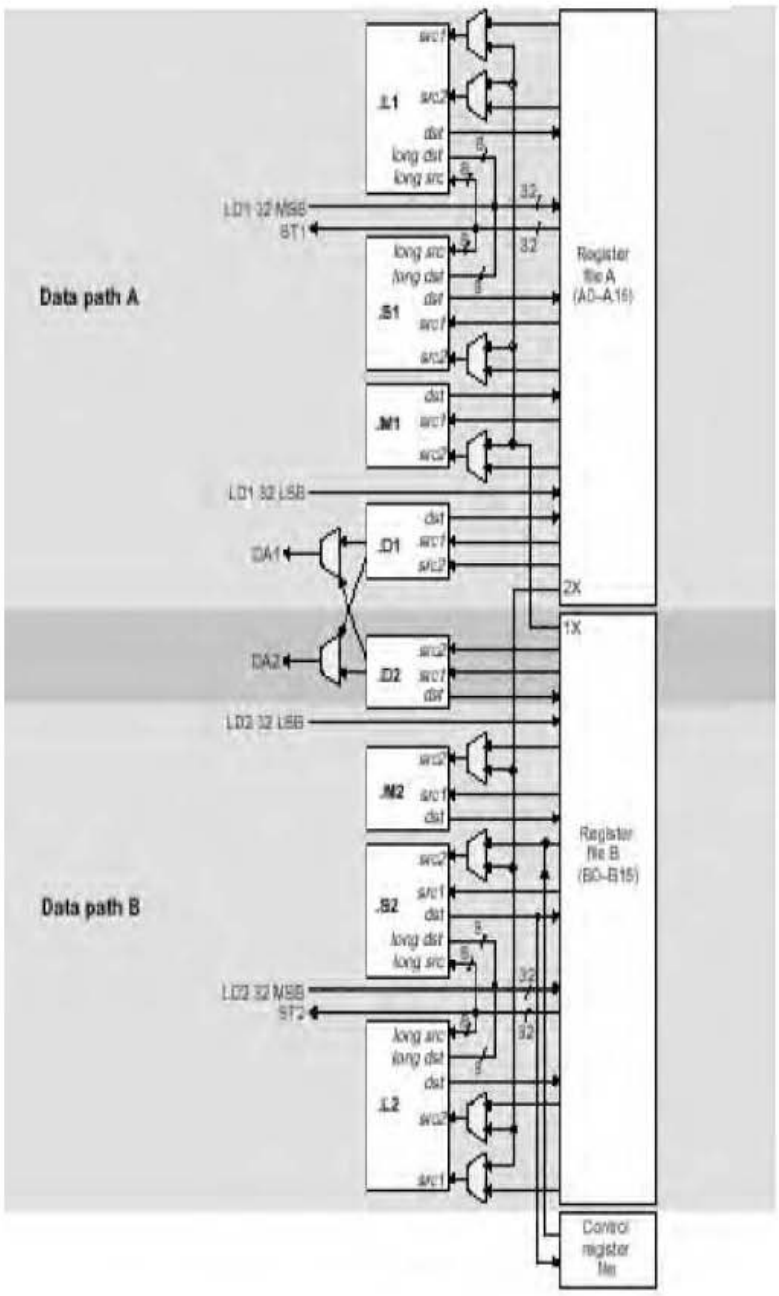


Figura 7.2: Camino de los datos del TMS320C6713

registro impar). El TMS320C6713 también usa ese par de registros para colocar valores de punto flotante de doble precisión de 64 bits.

Tabla 7.1: Unidades funcionales y las operaciones realizadas

| Unidad Funcional | Operaciones en punto fijo | Operaciones en punto flotante |
|---|---|---|
| Unidad .L (.L1, .L2) | Operaciones aritméticas y comparación de 32 y 40 bits. Cuenta de 0s a 1s mas a la izquierda para 32 bits. Normalización de 32 y 40 bits. | Operaciones lógicas; Operaciones de Conversión; DP $\rightarrow$ $SP, INT$ $\rightarrow$ $DP, INT \rightarrow SP$ |
| Unidad .S (.S1, .S2) | Operaciones aritméticas de 32 bits. Corrimientos de 32/40 bits y operaciones campos de bits en 32 bits. Operaciones lógicas de 32 bits. Saltos. Generación de Constantes. Transferencia de registros de/hacia registros (solamente S2) | Comparación recíproca; Operaciones de raíz cuadrada; Operaciones de Valor Absoluto; Operaciones de conversión SP a DP |
| Unidad .M (.M1, .M2) | Operaciones de multiplación, de 16x16 bits. | Operaciones de multiplicación de 32x32 bits. Operaciones de multiplicación de punto flotante |
| Unidad .D (.D1, .D2) | Sumas, restas y cálculos de direccionamiento circular de 32 bits. Carga y almacenamiento con offset constante de 5 bits. Carga y almacenamiento con offset constante de 15 bits (solo .D2) | Lectura de palabras dobles con offset constante de 5 bits |

### 7.2.4. Unidades funcionales

Las ocho unidades funcionales en los caminos de datos del TMS320C6713 pueden ser divididas en dos grupos de cuatro; cada unidad funcional, en un camino de datos, es casi idéntica a la unidad correspondiente, en el otro camino de datos por ejemplo, .L es muy similar a .L2. Las unidades funcionales son descritas en la tabla 7.1 La mayoría de los caminos en el CPU, soportan operaciones de 32 bits, y algunas soportan operaciones largas (40 bits). Cada unidad funcional tiene su propio puerto de escritura de 32 bits, en un archivo de registros de propósito general. Todas las unidades terminan en 1 (por ejemplo, .L1)cuando se refiere al archivo de registros A y en 2 cuando se refiere al archivo de registros B. Cada unidad funcional tiene dos puertos de lectura de 32 bits, para cada operando src1 y src2. Cuatro unidades (.L1, .L2, .S1 y .S2) tienen un puerto extra de 8 bits para ejecutar operaciones de 40 bits. Debido a que cada unidad tiene su propio puerto de escritura de 32 bits, las ocho unidades pueden ser usadas en paralelo en cada ciclo.

### 7.2.5. Archivos de registros de control del TMS320C6713

Una unidad (.S2) puede leer y escribir hacia los registros de control. Mostrados en la figura 7.2. La tabla 7.2, menciona y describe, los registros de control contenidos en el archivo de registros de control. Cada registro es accesado con la instrucción MVC.
El TMS320C6713 posee tres registros de configuración adicionales, para soportar operaciones de punto flotante. Los registros especifican los modos de redondeo de punto flotante para las unidades .M y .L. También contienen campos de bit para advertir si src1 y src2 son NaN (no es un número) o números desnormalizados. Además si resulta overflow o underflow, es inexacto, infinito o invalido. Hay campos que advierten si una división por cero fue ejecutada o si una comparación fue ejecutada con un NaN.

### 7.2.6. Caminos entre archivos de registros (Register Filters)

Cada unidad funcional lee directamente de y escribe directamente hacia el archivo de registros, dentro de su propio camino de datos. Esto es, las unidades .L1, .S1, .D1 y .M1 escriben en el archivo de registros A y las unidades .L2, .S2, .D2 y .M2 escriben en el archivo de registros B. Los archivos de registros son conectados a las unidades funcionales del archivo de registros opuesto, a través de los caminos cruzados 1X y 2X. Esos caminos cruzados permiten a las unidades funcionales, de un camino de datos, accesar a operandos de 32 bits, del lado opuesto. El camino cruzado 1X permite a las unidades funcionales del camino de datos A, leer su operando fuente del archivo de

Tabla 7.2: Registros de Control

| Abreviatura | Nombre | Descripción |
|---|---|---|
| AMR | Registro de Modo de Direccionamiento | Especifica si utiliza direccionamiento lineal o circular para cada uno de los ocho registros; también contiene el tamaño para el direccionamiento circular |
| CSR | Registro de Control de Estado | Contiene el bit de interrupción global, los bits de control del cache y otros bits de control de estado misceláneos |
| IFR | Registro de control de estado | Despliega el estado de las interrupciones |
| ISR | Registros para activar interrupción | Permite activar interupciones manualmente |
| ICR | Registro para interrupción | Permite limpiar interupciones pendientes manualmente |
| IER | Registro para retorno de interrupción | Permite habilitar/deshabilitar interrupciones individualmente |
| NRP | Puntero de retorno de interrupción no mascarable | Contiene la dirección de retorno de una interrupción no mascarable |
| PCE1 | Contador de Programa, Fase E1 | Contiene la dirección del paquete fetch (contiene el paquete de ejecución del pipeline) en la etapa E1. |

Tabla 7.3: Registros de control extendidos para el TMS320C6713

| Abreviatura | Nombre | Descripción |
|---|---|---|
| FADCR | Registro de configuración del sumador de punto flotante | Especifica el modo underflow, modo de redondeo, NaN y otras excepciones para la unidad .L |
| FAUCR | Registro de configuración auxiliar de punto flotante | Especifica modos de underflow, modos de redondeo, NaN y otras excepciones para la unidad .S |
| FMCR | Registro de configuración del multiplicador de punto flotante | Especifica modos de underflow, NaN y otras excepciones para la unidad .M |

registros B. El camino cruzado 2X permite a las unidades funcionales del camino de datos B, leer su operando fuente del archivo de registros A.

### 7.2.7. Caminos de Memoria, Cargas y Almacenamiento

Hay dos caminos de 32 bits, para leer los datos de memoria en los registros de almacenamiento: LD1 para el archivo de registros A y LD2 para el archivo de registros B. El TMS320C6713 también tiene un segundo camino de carga de 32 bits para ambos archivos de registros A y B. Este segundo camino permite a la instrucción LDDW leer simultáneamente dos registros de 32 bits en los lados A y B. Existen además dos caminos de 32 bits, ST1 y ST2, para almacenar valores de los registros a la memoria, para cada archivo de registros. Los caminos de lectura largos .L y .S son compartidos con los caminos de almacenamiento.

### 7.2.8. Caminos de direccionamiento de datos

Los caminos de direccionamiento de datos (DA1 y DA2) mostrados en la figura 7.2 colocados fuera de las unidades .D, permiten generar direcciones de datos de un archivo de registros. Con eso se sostienen cargas y almacenamientos en memoria, desde el otro archivo de registros. Sin embargo, las cargas y almacenamientos ejecutados en paralelo, debe cargar a y de el mismo archivo de registro. Aunque también existe la alternativa de que ambos usen un camino cruzado al registro opuesto.

### 7.2.9. Mapeo entre instrucciones y Unidades Funcionales

La tabla 7.4 muestra el mapeo entre las instrucciones y las unidades funcionales para las instrucciones de punto fijo del TMS320C6713. La tabla 7.5 muestra el mapeo entre las instrucciones y las unidades funcionales para las instrucciones de punto flotante del TMS320C6713.

Tabla 7.4: Mapeo de instrucciones de punto fijo y unidades funcionales

| Unidad .L | Unidad .M | Unidad .S | | Unidad .D | |
|---|---|---|---|---|---|
| ABS | MPY | ADD | SET | | ADD STB(15-bit offset) Solo .S2 |
| ADD | MPYU | ADDK | SHL | ADDAB | STH(15-bit offset) Solo .S2 |
| ADDU | MPYUS | ADD2 | SHR | ADDAH | STW(15-bit offset) Solo .S2 |
| AND | MPYSU | AND | SHRU | ADDAW | SUB |
| CMPEQ | MPYH | B disp | SHRL | LDB | SUBAB |
| CMPGT | MPYHU | B IRP | SUB | LDBU | SUBAH |
| CMPGTU | MPYHUS | B NRP | SUBU | LDH | SUBAW |
| CMPLT | MPYHSU | B reg | SUB2 | LDHU | ZERO |
| CMPLTU | MPYHL | CLR | XOR | LDW | |
| LMBD | MPYHLU | EXT | ZERO | LDB | |
| MV | MPYHLUS | EXTU | | LDBU | |
| NEG | MPYHSLU | MV | | LDH | |
| NORM | MPYLH | MVC | | LDHU | |
| NOT | MPYLHU | MVK | | LDW | |
| OR | MPYLUHS | MVKH | | MV | |
| SADD | MPYLSHU | MVLKH | | STB | |
| SAT | SMPY | NEG | | STH | |
| SSUB | SMPYHL | NOT | | STW | |
| SUB | SMPYLH | OR | | | |
| SUBU | SMPYH | | | | |
| SUBC | | | | | |
| XOR | | | | | |
| ZERO | | | | | |

Tabla 7.5: Mapeo de instrucciones de punto flotante y unidades funcionales

| Unidad .L | Unidad .M | Unidad .S | Unidad .D |
|---|---|---|---|
| ADDP | MPYDP | ABSDP | ADDAD |
| ADDSP | MPYI | ABSSP | LDDW |
| DPINT | MPYID | CMPEQDP | |
| DPSP | MPYSP | CMPEQSP | |
| INTDP | | CMPGTDP | |
| INTDPU | CMPGTSP | | |
| INTSP | | CMLTDP | |
| INTSPU | CMPLTSP | | |
| SPINT | | RCPDP | |
| SPTRUNC | | RCPSP | |
| SUBDP | | RSQRDP | |
| SUBSP | | RSQRSP | |
| SPDP | | | |

Tabla 7.6: Generación de direccionamiento indirecto para Load/Store

| Tipo de Direccionamiento | Ninguna modificación del registro de dirección | Preincremento o Predecremento del registro de dirección | Postincremento o Postdecremento del registro de dirección |
|---|---|---|---|
| Registro indirecto | *R | *++R | *R++ |
| | | *–R | *R–CMPEQDP |
| Registro relativo | *+R[$ucst5$]· | *++R[$ucst5$] | *++R[$ucst5$] |
| | *-R[$ucst5$] | *–R[$ucst5$] | *–R[$ucst5$] |
| Base + index | *+R[$offsetR$] | *++R[$offsetR$] | *++R[$offsetR$] |
| | *-R[$offsetR$] | *–R[$offsetR$] | *–R[$offsetR$] |

## 7.3. Modos de direccionamiento

Los modos de direccionamiento son lineales por default para el TMS320C6713 aunque también existe el modo de direccionamiento circular. El modo de direccionamiento se especifica con el registro modo de direccionamiento (AMR). Con todos los registros se puede ejecutar el direccionamiento lineal. Solo en ocho de ellos se puede ejecutar el direccionamiento circular: del A4 a A7 (que son usados por la unidad .D1) y del B4 a B7 (que son usados por la unidad D2). Ninguna otra unidades puede ejecutar direccionamiento circular. Las instrucciones LDB/LDH/LDW, STB/STH/STW, ADDAB/ADDAH/ADDAW, y SUBAB/SUBAH/SUBAW se apoyan en el registro AMR, para determinar que tipo de calculo del direccionamiento es ejecutado por esos registros. El CPU del dispositivo TMS320C6713 tienen arquitectura de carga/almacenamiento, lo que significa que la única manera de accesar datos en memoria es con la instrucción de carga o almacenamiento La tabla 7.6 muestra la sintaxis de un direccionamiento indirecto, para una localización en memoria

## 7.4. Interrupciones

El CPU del dispositivo TMS320C6713 tienen 14 interrupciones. Estas son reset, la interrupción no mascarable (NMI) e interrupciones de la 4 a la 15. Estas interrupciones corresponden a las señales RESET, NMI e INT4-INT15 respectivamente, sobre los límites del CPU. En el mismo dispositivo TMS320C6713, estas señales pueden estar ligadas directamente a lo pines del dispositivo, conectando periféricos al chip, o pueden ser desactivadas permanentemente, cuando están ligadas e inactivas en el chip. Generalmente, RESET y NMI son conectadas directamente a los pines del dispositivo. Las características del servicio de interrupción incluyen:

- El pin IACK del CPU es usado para confirmar la recepción de una petición de interrupción
- Los pines INUM0 INUM3 indican el vector de interrupción que está siendo utilizado
- Los vectores de interrupción son reubicables
- Los vectores de interrupción consisten de un paquete fetch. Con los paquetes se proporciona un rápido servicio.

# Capítulo 8

# Implementación de la Codificación de Voz en el TMS320C6713

El siguiente paso después de la simulación es la implementación, la cual es posible realizar con la ayuda de la herramienta *Real Time Workshop*, perteneciente también a la familia de *Mathwoks*.

## 8.1. Implementación del banco de filtros de cinco canales

Debido a que el microprocesador a usar es el TMS320C6713, es necesario insertar en el diseño de *Simulink* realizado anteriormente, elementos y configuraciones de acuerdo al microprocesador mencionado.
Primero es necesario insertar el bloque C6713DSK que representa las configuraciones para poner en marcha el microprocesador, dichas configuraciones son realizadas por *Simulink* automáticamente al insertar el bloque mencionado. Después, se agrega en la entrada el bloque ADC, que tiene como función convertir la señal analógica del micrófono a una señal digital, para que pueda ser procesada por el micropocesador. Es importante mencionar que es necesario especificar, en este último bloque, la tasa de muestreo a la cual se va a trabajar.
Del mismo modo se requiere insertar un bloque DAC en la salida, que tiene la tarea de convertir la señal digital ya procesada en una señal analógica disponible para ser escuchada.
El diseño del banco de filtros de cinco canales antes de ser implementado y con los elementos antes descritos se muestra a continuación en la figura 8.2
En el menú tools de Simulink es posible seleccionar la opción *Real Time Workshop/Build model*. Por medio de esta opción es que el diseño del banco de filtros inicia su implementación. Primero se crean códigos en el lenguaje de programación C del banco de filtros y posteriormente se establece un vínculo entre Matlab y el programa Code Composer el cual pone en funcionamiento la tarjeta del microprocesador y genera los códigos en ensamblador a partir de los códigos en C generados por Simulink.
Se puede observar en la figura 8.2 la pantalla del programa cuando la tarjeta está en uso y el banco de filtros se está ejecutando, sí en ese momento, en la entrada de la tarjeta se conecta un micrófono y en la salida una bocina o audifonos se puede corroborar que el banco de filtros está funcionando correctamente. Los códigos generados por Real Time Workshop se presentan en el apéndice y con ayuda de los comentarios ahí escritos y algunos conocimientos del lenguaje C, es posible entenderlos de forma correcta.

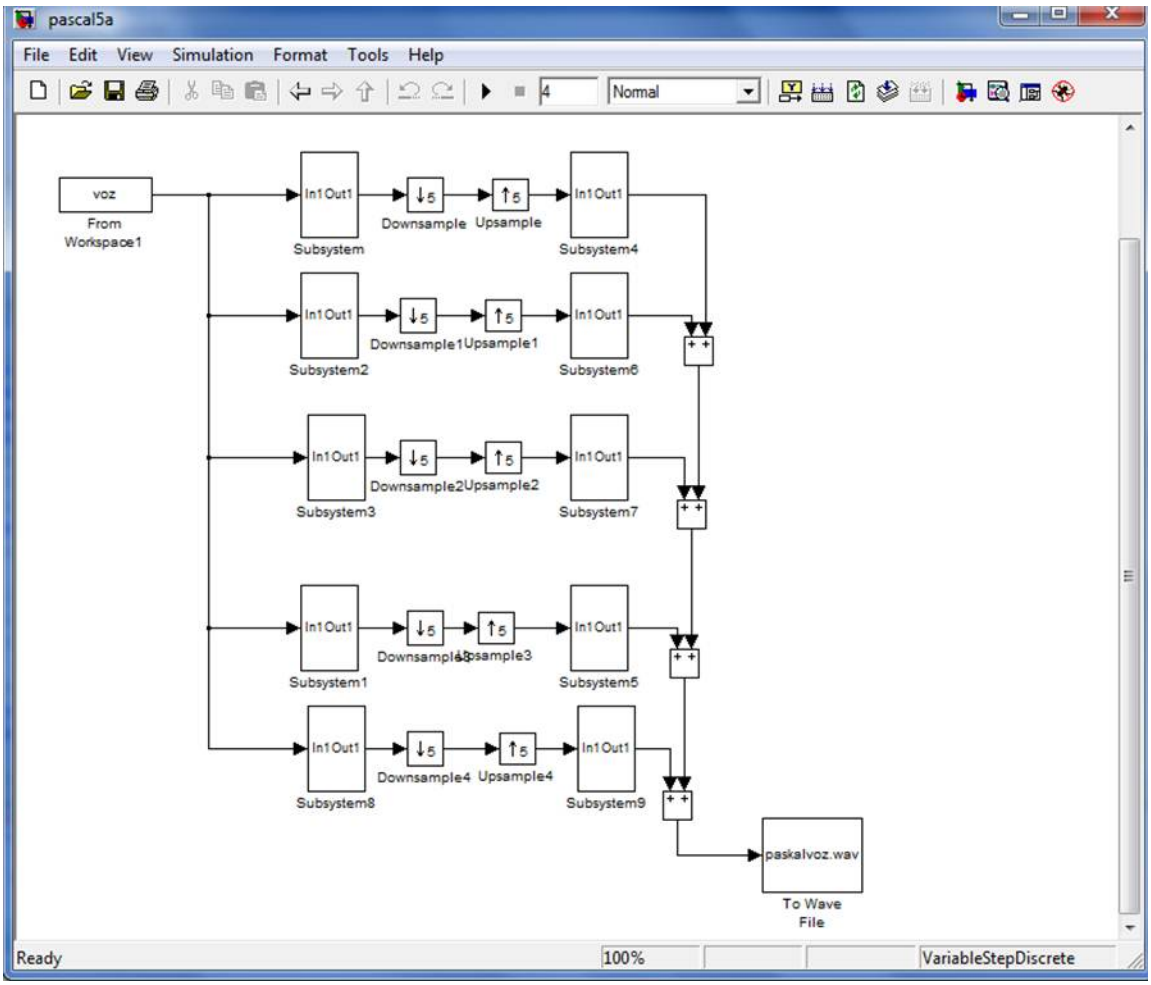


Figura 8.1: Implementación del banco de filtros de 5 canales en Simulink

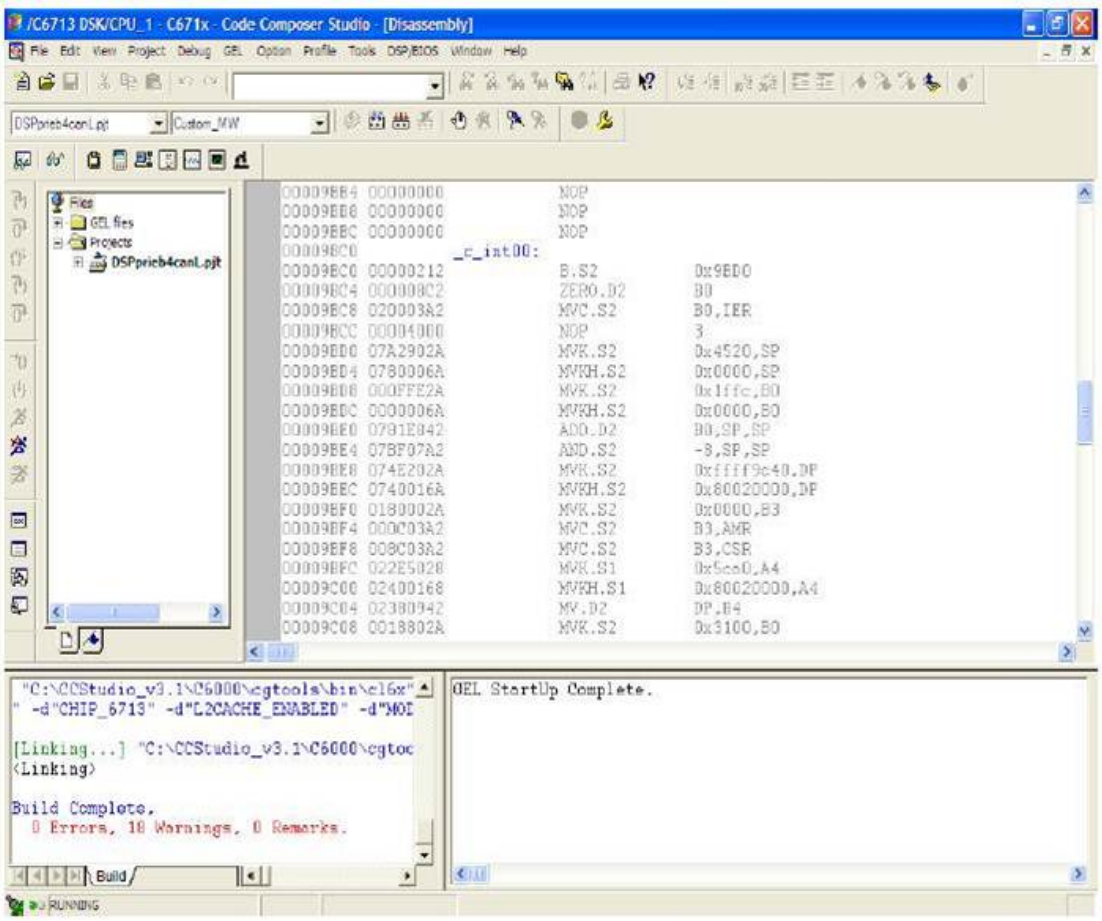


Figura 8.2: Ejecución del banco de filtros en Code Composer

# Capítulo 9

# Conclusiones

Como pudimos apreciar a lo largo de este trabajo, el procesamiento digital de señales aplicado al fenómeno de la voz ha sido muy subestimado al pensar que dicho fenómeno ha sido estudiado mas que a profundidad y exhaustivamente. Sin duda, la voz sigue siendo un fenómeno sumamente interesante y objeto de un estudio todavía muy amplio.

Uno de los problemas que se presenta con el fenómeno de la voz es que su procesamiento sigue siendo demasiado complejo en terminos de codificación. Contando con algoritmos eficientes que permitan su optimización en procesos de digitalización de señales, es posible aplicarlo para otro tipo de procedimientos que son inherentes a las telecomunicaciones.

Hablando específicamente de lo expuesto en esta tesis, una vez diseñados e implementados los bancos de filtros de 5 canales con resconstrucción perfecta para codificador de audio de los cuales tratamos, es momento de reportar los puntos relevantes de este trabajo y las conclusiones a las que se ha llegado, asi como verificar si todo el trabajo realizado tuvo los resultados esparados.

Tal como la literatura acerca de los bancos de filtros y el procesamiento multitasa menciona, es posible diseñar y obtener bancos de filtros de n canales que recuperen la señal procesada de forma adecuada, es decir, que presenten un reconstrucción perfecta independientemente del orden de los filtros que para ese objetivo se utilicen. Sin embargo, el hecho de usar filtros con un orden alto mejora la eficiencia del propio filtro, cuestión relevante cuando se trata de filtros FIR. Lo anterior fue corrobarado durante la simulación e implementación de los bancos de filtros diseñados.

A lo largo de este trabajo se presentaron bancos de filtros aplicados a Procesamiento de señales de voz empleandos aplicando un algoritmo eficiente. Si bien existen ya estructuras de bancos que son muy eficientes computacionalmente, como la estructura polifase, los bancos implementados aquí descritos se realizaron debido a su facilidad para comprenderlos pero sobre todo por la facilidad opeacional que implica su implementación. Al utilizar operaciones en su mayoría matriciales, fue posible hacer un proceso ligero, eficiente pero sobre todo bastante funcional. Ello nos puede llevar en un futuro a aplicarlo a otro tipo de problemas en el sector telecomunicaciones, tales como la transmisión y filtrado de señales satelitales, señales que impliquen imágenes e incluso señales que tengan ver con movimiento (video, por ejemplo)

El papel jugado por los lenguajes de alto nivel en esta tesis, y en el avance y desarrollo de una investigación en general, es de suma relevancia. Estos programas de computadora permitieron agilizar el trabajo ya que apoyaron durante todas las etapas de la investigación: Diseño, Simulación e Implementación, además de resultar ser muy amigables, es decir, fáciles de operar. La obtención de las funciones de transferencias de los filtros de cada banco de filtros no habría sido tan rápida sin la ayuda de Matlab, el cual posee herramientas e instrucciones que permiten el diseno eficaz de filtros FIR y IIR. La simulaci´on no habr´ia sido posible sin Simulink y la implementacion habría tomado largo tiempo sin la herramienta Real Time Workshop que permite crear códigos en lenguaje C a partir del modelo creado en simulink, que a su vez son utilizados por Code Composer Studio para generar archivos en lenguaje ensamblador que permiten poner en funcionamiento al Microprocesador utilizado (TMS320C6713).

Respecto a los resultados obtenidos, podemos resaltar que fueron de suma importancia, pues se pudo demostrar que un algoritmo que existía solamente a nivel descriptivo y en la teoría, pudo ser implementado en una aplicación real, consistente y tangible. Este algoritmo podría implementarse de manera específica en otro tipo de dispositivos (tales como circuitos satelitales, aparatos de comunicación móvil y dispositivos de transmisión/recepción de señales en redes de telecomunicaciones, entre otros) para lograr eficiencia y calidad en un fenómeno sumamente común pero hasta la fecha dificil de abordad: la voz humana. Mediante el algoritmo utilizado y las técnicas empleadas y ampliamente descritas en este trabajo, es posible incluso derivar otro tipo de estudios, tales como es la Fonética, el reconocimiento y caracterización de señales de audio y el procesado en términos analíticos de situaciones que impliquen la voz humana. Es posible disminuir los cálculos y las operaciones en un grado de suma importancia y con ello se pueden llegar a mejoras en los sistemas que actualmente ya se emplean.

Por otra parte, la metodología con la que fue realizada la implementación es tan flexible que es posible modificar los filtros del banco rápidamente y realizar un implementación diferente en cuestión de minutos. Lo anterior permite que la implementación y metodología antes mencionadas puedan ser utilizadas en el laboratorio correspondiente y así, de forma muy dinámica, permitirle a un alumno interactuar con un microprocesador, así como verificar y poner en práctica sus conocimientos sobre, filtros, procesamiento multitasa, banco de filtros, interpolación y/o decimación. Evidentemente, es necesario a partir del material que el presente trabajo ofrece, diseñar una serie de actividades determinadas para el correcto uso del material y un aprovechamiento eficiente de parte del alumno.

A manera de prospectiva, consideramos que este estudio está completo pero no por ello esta terminado. Se sugiere que sea la partida para otro tipo de estudios que por estar fuera del alcance de este trabajo no fueron abordados, pero que sin duda son de extrema importancia para las telecomunicaciones. Este mismo método puede emplearse para el estudio, codificación, filtrado y caracterización digital de señales correspondientes a otro tipo de fenómenos y de particular interés en el área de las telecomunicaciones, tales como es la transmisión/recepción de imágenes, aplicaciones satelitales, transferencia de información, redes telemáticas e incluso procesamiento de señales cinemáticas (video y voz en tiempo real).

A manera de sugerencia, se puede mencionar también que este estudio puede sentar las bases para otro tipo de aplicaciones que puedan ser de manufactura particular. Si bien fue utilizada la infraestructura de Texas Instuments, la cual fue de particular ayuda para la realización de este trabajo, y se emplearon técnicas de análisis y de simulación que redujeron en gran medida el trabajo realizado, lo cierto es que podemos realizar circuitos propios que podrían abatir costos y generar productividad en el tema de innovación tecnológica en nuestro país. Actualmente, este tipo de algoritmos estudiados se han implementado solamente en entornos de simulación y en nuestro caso fueron implementados en una plataforma ya existente, pero es posible también hacer manufactura propia. Una solución a esta alternativa es generar circuitos fabricados de manera independiente y sin la dependencia de patentes ni de circuitos fabricados anteriormente. Hay técnicas que parecen prometedoras, tal como es el caso del Diseño de Filtros Digitales utilizando la técnica coplanar de Capacitores conmutados. Es una técnica que de hecho ya se esta estudiando a profundidad en Europa pero que no ha sido profundizado debido a la falta de interés en el tema. En nuestra Universidad, hay equipos multidisciplinarios que han encontrado una amplia oportunidad para poder combinar esta técnica con el estudio abordado en el presente trabajo, sin embargo lo anterior está completamente fuera de nuestro alcance por el momento. La sugerencia específica es continuar el estudio de los algoritmos aquí presentados pero con manufactura propia, lo cual llevaría mayor tiempo y recursos en términos financieros y de tiempo, incluso muy probablemente un estudio que pueda ser un buen tema de tesis a nivel doctorado, dada la complejidad pero sobre todo el nivel de innovación que se alcanzaría con esto.

Finalmente, con todo lo mencionado anteriormente, es posible considerar que los objetivos planteados para el presente trabajo se cubrieron de forma satisfactoria, cuestión que se puede constatar con las implementaciones ya mencionadas y mediante lo reportado mediante este documento.

# Apéndices

# Apéndice A

# Códigos

## A.1. Código de Banco de Filtros de 5 Canales

```
%Diseno de filtros para el Banco de Filtros Multinivel
[h0 h1 g0 g1]= firpr2chfb(7,0.49999)
fm=22050;
[hh0,w]=freqz(h0,1,200);
[hh1,w]=freqz(h1,1,200);
[hg0,w]=freqz(g0,1,200);
[hg1,w]=freqz(g1,1,200);
f=fm*w/(2*pi);
figure
plot(f,20*log10(abs(hh0)),-,f,20*log10(abs(hh1)),--)
xlabel(frecuencia [Hz])
ylabel(Magnitud [dB])
figure
plot(f,20*log10(abs(hg0)),-,f,20*log10(abs(hg1)),--)
xlabel(frecuencia [Hz])
ylabel(Magnitud [dB])
%Verificacion de la reconstruccion perfecta
n=0:11;
pr=0.5*conv(g0,h0)+0.5*conv(g1,h1);
%Conversion a Banco de Filtros de 4 bandas
H02=[h0(1) 0 h0(2) 0 h0(3) 0 h0(4) 0 h0(5) 0 h0(6) 0 h0(7)];

H12=[h1(1) 0 h1(2) 0 h1(3) 0 h1(4) 0 h1(5) 0 h1(6) 0 h1(7)];
G02=[g0(1) 0 g0(2) 0 g0(3) 0 g0(4) 0 g0(5) 0 g0(6) 0 g0(7)];
G12=[g1(1) 0 g1(2) 0 g1(3) 0 g1(4) 0 g1(5) 0 g1(6) 0 g1(7)];
H00=conv(h0,H02);
H01=conv(h0,H12);
H10=conv(h1,H02);
H11=conv(h1,H12);
G00=conv(g0,G02);
G01=conv(g0,G12);
G10=conv(g1,G02);
G11=conv(g1,G12);
```

sectionImplementación

```
/*
```

```
* File: DSPprieb4can_data.c
*
* Real-Time Workshop code generated for Simulink model DSPprieb4can.
*
* Model version : 1.5
* Real-Time Workshop file version : 6.5 (R2006b) 03-Aug-2006
* Real-Time Workshop file generated on : Wed Apr 15 08:40:44 2009
* TLC version : 6.5 (Aug 3 2006)
* C source code generated on : Wed Apr 15 08:40:51 2009
*/
#include "DSPprieb4can.h"
#include "DSPprieb4can_private.h"
/* Block parameters (auto storage) */
#pragma DATA_ALIGN(DSPprieb4can_P,8)
Parameters_DSPprieb4can DSPprieb4can_P = {
/* DiscreteFilter10_A : <Root>/Discrete Filter10
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter10_C : <Root>/Discrete Filter10
*/
{ 3.0641673148813753E-001, 1.0021204742470907E-001, -7.1520341073669461E-002,
-9.1702675155071856E-002, 3.6565640701629203E-002, 1.4550456329120948E-001,
-2.5929204477517348E-001 },
/* DiscreteFilter4_A : <Root>/Discrete Filter4
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter4_C : <Root>/Discrete Filter4
*/
{ -6.1283346297627506E-001, 2.0042409484941814E-001, 1.4304068214733892E-001,
-1.8340535031014371E-001, -7.3131281403258405E-002, 2.9100912658241895E-001,
5.1858408955034696E-001 },
1.0820389672762274E+000, /* DiscreteFilter4_D : <Root>/Discrete Filter4
*/
0.0, /* Upsample4_IC : <Root>/Upsample4
*/
/* DiscreteFilter8_A : <Root>/Discrete Filter8
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter8_C : <Root>/Discrete Filter8
*/
{ 2.9100912658241895E-001, 7.3131281403258405E-002, -1.8340535031014371E-001,
-1.4304068214733892E-001, 2.0042409484941814E-001, 6.1283346297627506E-001,
1.0820389672762274E+000 },
-5.1858408955034696E-001, /* DiscreteFilter8_D : <Root>/Discrete Filter8
*/
/* DiscreteFilter11_A : <Root>/Discrete Filter11
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter11_C : <Root>/Discrete Filter11
*/
{ 1.4550456329120948E-001, -3.6565640701629203E-002, -9.1702675155071856E-002,
7.1520341073669461E-002, 1.0021204742470907E-001, -3.0641673148813753E-001,
5.4101948363811370E-001 },
```

```
2.5929204477517348E-001, /* DiscreteFilter11_D : <Root>/Discrete Filter11
*/
0.0, /* Downsample1_IC : <Root>/Downsample1
*/
/* DiscreteFilter2_A : <Root>/Discrete Filter2
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter2_C : <Root>/Discrete Filter2
*/
{ 3.0641673148813753E-001, 1.0021204742470907E-001, -7.1520341073669461E-002,
-9.1702675155071856E-002, 3.6565640701629203E-002, 1.4550456329120948E-001,
-2.5929204477517348E-001 },
5.4101948363811370E-001, /* DiscreteFilter2_D : <Root>/Discrete Filter2
*/
0.0, /* Downsample4_IC : <Root>/Downsample4
*/
0.0, /* Upsample2_IC : <Root>/Upsample2
*/
/* DiscreteFilter6_A : <Root>/Discrete Filter6
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter6_C : <Root>/Discrete Filter6
*/
{ 2.9100912658241895E-001, 7.3131281403258405E-002, -1.8340535031014371E-001,
-1.4304068214733892E-001, 2.0042409484941814E-001, 6.1283346297627506E-001,
1.0820389672762274E+000 },
-5.1858408955034696E-001, /* DiscreteFilter6_D : <Root>/Discrete Filter6
*/
/* DiscreteFilter3_A : <Root>/Discrete Filter3
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter3_C : <Root>/Discrete Filter3
*/
{ 1.4550456329120948E-001, -3.6565640701629203E-002, -9.1702675155071856E-002,
7.1520341073669461E-002, 1.0021204742470907E-001, -3.0641673148813753E-001,
5.4101948363811370E-001 },
2.5929204477517348E-001, /* DiscreteFilter3_D : <Root>/Discrete Filter3
*/
0.0, /* Downsample5_IC : <Root>/Downsample5
*/
0.0, /* Upsample3_IC : <Root>/Upsample3
*/
/* DiscreteFilter5_A : <Root>/Discrete Filter5
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter5_C : <Root>/Discrete Filter5
*/
{ -6.1283346297627506E-001, 2.0042409484941814E-001, 1.4304068214733892E-001,
-1.8340535031014371E-001, -7.3131281403258405E-002, 2.9100912658241895E-001,
5.1858408955034696E-001 },
1.0820389672762274E+000, /* DiscreteFilter5_D : <Root>/Discrete Filter5
*/
0.0, /* Upsample5_IC : <Root>/Upsample5
```

```
*/
/* DiscreteFilter9_A : <Root>/Discrete Filter9
*/
{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 },
/* DiscreteFilter9_C : <Root>/Discrete Filter9
*/
{ -6.1283346297627506E-001, 2.0042409484941814E-001, 1.4304068214733892E-001,
-1.8340535031014371E-001, -7.3131281403258405E-002, 2.9100912658241895E-001,
5.1858408955034696E-001 },
1.0820389672762274E+000 /* DiscreteFilter9_D : <Root>/Discrete Filter9
*/
};
/* File trailer for Real-Time Workshop generated code.
*
* [EOF]
*/
```

# Bibliografía


[1] Sacerdoti, Juán., «Transformada Z», *Apuntes de Matemáticas*, Departamento de Matemáticas, Universidad de Buenos Aires, págs. 5–7, 1960.

[2] B. P. Lathi, *Introducción a la Teoría y Sistemas de Comunicación*, decimonovena edición, Limusa, México DF, 2001.

[3] B. P. Lathi, *Sistemas de Comunicación*, Segunda Edición, Mc-GrawHill, México DF, 1991.

[4] J. G. Proakis, *Discrete-Time Processing of Speech Signals*, Segunda Edición, MacMillan, México, 2000.

[5] Psenicka, Bohumil, et al, *Practical Design of Digital Filters Using the Pascal Matrix*, Signal and Processing, IEEE, 2002.

[6] A. V. Oppenheim, A.S. Willsky and S.H. Nawab, *Señales y Sistemas*, Prentice Hall, 1998.

[7] W. Tomasi, *Sistemas de Comunicaciones Electrónicas*, Prentice Hall, 2003.

[8] E. B. Albertí, *Señales y Sistemas de Tiempos Discreto*, UPC, 2003.

[9] C. J. Mitchell, F. C. Piper, *A Classification of Time Element Speech Scramblers*, Journal of the Institution of Electronic and Radio Engineers, 1985.

[10] L. S. Lee, G. S. Chou, C. S. Shang, *Frecuency or time domain speech scrambling technique and system which does not require any frame synchronization*, US Patent 4591673, 1983.

[11] Phil Lapsley, Jeff Bier, Amit Shoham and Edward A. Lee, *DSP Processor Fundamentals: Architectures and Features*, Berkeley, California: Berkeley Design Technology Inc, 1996.

[12] Alan V. Oppenheim and Alan S. Willsky, *Signals and systems*, Pearson Educationm 1998. Pag 187.

[13] R. P. Areny, *Adquisión y Distribución de Señales*, Pearson Educationm 1993.

[14] H. J. Beker, C.J. Mitchell, *Permutations with restricted displacement*, Society for Industrial and Applied Mathematics, EE.UU. 1987.

[15] M. S. Ibañez, R. G. Diaz, *Generación y analisis de secuencias pseudoaleatorias*, Ediciones de la universidad Politecnica de Catalunya SL, Epaña, 1999.

[16] L. R. Rabiner, R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall Signal Processing Series, 1997.

[17] U. Zolzer, *Digital Audio Signal Processing*, Jhon Wiley  Son Ltd, 2008.

[18] B. Psenicka, F, García Ugalde y V.F Ruíz., *Practical Design of Digital Filtres Using the Pascal Matrix*, Paper en Arbitraje en IEEE, 2009.

[19] N. J. Fliege, *Multirate Digital Signal Processing*, John Wiley Sons, Ltd, New York 2000.

[20] Douglas K. Lindner, *Introducción a las senales y a los sistemas*, McGraw-Hill. Caracas, Venezuela. 2002

[21] Sanjit K. Mitra, *Digital Signal Processing. A computer-based approach*, 3rd Edition. McGraw-Hill. New York, 2006.

[22] Bohumil Psenicka, *Procesamiento de Señales. Filtros Pasivos, Activos y Digitales*, UNAM, 2002

[23] D. Esteban, C. Galand, *Application of Quadrature Mirror Filters to Split Band Voice Coding*, Schemes Proc IEEE ICASSP77, May 1997.

[24] M. Bellanger, G. Bonnerot, and M. Coudreuse, *Digital filtering by poliphase network: application to sample-rate alteration and filter banks*, IEEE Trans. Acoustics, Speech and Signal Processing, ASSP-24:109-114, April 1976.

[25] Rabiner R. and Gold B., *Theory and Applications of Digital Signal Processing*, Prentice-Hall, 1975.

[26] B. Psenicka, O. Nieto y V. López, *Prácticas de laboratorio con microprocesadordes TMS320C6711*, UNAM, 2002.