UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

"IMPLEMENTACIÓN DE LAS TÉCNICAS DE SOLAPAMIENTO-SUMA Y SOLAPAMIENTO-ALMACENAMIENTO EN UNA TARJETA DE PROCESAMIENTO DIGITAL DE SEÑALES"

PRESENTADO POR
Octavio Rafael Calderón Escobar

PARA OPTAR AL TITULO DE:
INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, FEBRERO DE 2003

UNIVERSIDAD DE EL SALVADOR

RECTORA :

Dra. Maria Isabel Rodríguez

SECRETARIA GENERAL :

Licda. Lidia Margarita Muñoz Vela

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

Ing. Alvaro Antonio Aguilar Orantes

SECRETARIO :

Ing. Saúl Alfonso Granados

ESCUELA

DIRECTOR :

Ing. Luis Roberto Chevéz Paz

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA ELECTRICA Y ARQUITECTURA
ESCUELA DE INGENIERIA ELECTRICA

Trabajo de Graduación previo a la opción al grado de:
INGENIERO ELECTRICISTA

Titulo :

"IMPLEMENTACION DE LAS TÉCNICAS DE SOLAPAMIENTO-SUMA Y SOLAPAMIENTO –ALMACENAMIENTO EN UNA TARJETA DE PROCESAMIENTO DIGITAL DE SEÑALES"

Presentado por :

Octavio Rafael Calderón Escobar

Trabajo de Graduación aprobado por:

Docente Director:

Ing. Luis Roberto Chevéz Paz

Docente Director:

Ing. Luis Roberto Chevéz Paz

San Salvador, febrero de 2003

INDICE.

INTRODUCCIÓN GENERAL.

El Tratamiento Digital de Señales (DSP, por sus siglas en Inglés), constituye otra herramienta que puede ayudar eficientemente a formular aplicaciones en diversos campos del conocimiento científico, tales como: tratamiento de señales sísmicas, exploración del subsuelo terrestre, interpretación de señales biológicas, sistemas de comunicaciones inalámbricas, determinaciones por sonar, sistemas de control por radar, formas de reproducción de imágenes y música, etc. El presente trabajo de graduación establece las bases necesarias, para diseñar técnicas generales de DSP y en particular las relativas a la obtención de filtros digitales, orientados a la convolución, filtrado y muestreo de señales. En síntesis el problema a resolver consiste en implementar, mediante programas en *MATLAB* (Lenguaje para cálculo numérico y visualización para ciencia e ingeniería), y en la DSP56L811EVM [4] (la cual incluye un microcontrolador, DSP, un sistema de codificación, reloj, memorias, etc) una convolución en tiempo real.

El contenido general del trabajo de graduación está distribuido así: El Capítulo I muestra el fundamento teórico de la transformada discreta de Fourier (DFT, por sus siglas en Inglés) y los algoritmos de la FFT, relativos al análisis de señales y filtros digitales.

En el Capítulo II se considera el diseño de filtros digitales y específicamente el método óptimo de *Parks-McClellan* combinado con el algoritmo de intercambio de *Remez* para la obtención de un filtro digital mediante el empleo de pocos coeficientes.

La convolución, en sus formas, por bloques y circular, así como las técnicas de solapamiento-suma y solapamiento-almacenamiento, son expuestas en el Capítulo III.

La descripción del desarrollo de la implementación se encuentra en el último capítulo.

Finalmente, programas y detalles mucho más específicos de la simulación, desarrollo y otras aclaraciones (que para simplificar la lectura se excluyen de los capítulos) se presentan en los anexos correspondientes.

No omito expresar, que este trabajo de graduación, no pretende, ser completo y no admitir observaciones. Si en la desarrollo del trabajo no se han obtenido resultados rígidamente coincidentes con las determinaciones teóricas; este se debe considerar como estimulo al esfuerzo del estudioso para buscar detalles complementarios.

CAPÍTULO I

TRANSFORMADA DISCRETA DE FOURIER (DFT).

1.1 INTRODUCCIÓN.

En este capítulo se desarrolla un tema de mucho interés en DSP, y me refiero a un gran poema matemático: “ la transformada de Fourier”, pero concretamente se estudia la transformada de Fourier discreta en esta se hace referencia a los algoritmos que la definen. Para empezar la DFT es una transformada propia (tal como la transformada de Fourier o las Series de Fourier) y una operación reversible muy útil. Como su nombre lo implica, tiene propiedades matemáticas que son enteramente análogas a la transformada de Fourier en tiempo continuo. Particularmente define el espectro de frecuencia de una serie o secuencia en el dominio del tiempo, también cumple la propiedad de que la multiplicación de los espectros de frecuencia de dos series o secuencias en el dominio del tiempo corresponden a la convolución de las dos series o secuencias en el dominio del tiempo.

Las técnicas de DSP se utilizan para analizar una forma de onda continua la cual hace necesario que se tomen muestras de dicha forma de onda continua (normalmente a intervalo de tiempo igualmente espaciados) para producir una secuencia de datos discretos que puedan ser alimentados a una computadora o microcontrolador DSP (ejemplo: la tarjeta de evaluación DSP56L811EVM).

Como se sabe muy bien [2], tal secuencia de datos discretos representa completamente la forma de onda continua, si dicha señal es de banda limitada en frecuencia y se toman las muestras a una frecuencia que es por lo menos dos veces la frecuencia más alta que puede representar dicha forma de onda continua. Cuando las muestras están igualmente espaciadas, se les conoce como muestras de *Nyquist*.

La DFT esta estrechamente relacionada con la transformada de Fourier de la forma de onda continua, de la cual se han tomado las muestras de datos para formar la secuencia

en el dominio del tiempo. Esto hace a la DFT particularmente útil para análisis espectral y filtros digitales.

A la familia de algoritmos utilizados para el cálculo eficiente de la DFT de secuencias de números, reales o complejos de longitud finita se le conoce como FFT. Hay varias formas de calcular la DFT, desde la forma elemental hasta la eficiente. El método de cálculo directo a partir de su definición es una solución simple para resolver la DFT, también el método de *Goertzel* es otra forma de solución, este hace notar que las propiedades de simetría, periodicidad y recurrencia pueden ayudar a reducir los cálculos. Otro método para calcular la DFT, el cual produce el mismo resultado que los métodos directos, es eficiente, reduce el tiempo de cálculo y errores de redondeo al nivel de cuantificación más cercano, a este método se le conoce como FFT.

J. W. Cooley y J.W. Tukey a quienes se les acredita por el método de la FFT, en su artículo [1] dieron a conocer un algoritmo que requería menos esfuerzo para calcular los coeficientes de Fourier.

De acuerdo a lo anterior surge la siguiente pregunta: ¿Por qué es importante la FFT? La razón del porque es importante esta técnica, es el hecho de ser muy simple de desarrollar que los otros métodos. Ahora bien, como sabrá el análisis de Fourier es de importancia extraordinaria en casi todas las áreas de las matemáticas, ciencias e ingeniería. Para desarrollar trabajos analíticos sobre señales se utilizan varios métodos tales como: La transformada de Laplace, La transformada de Fourier, Las series de Fourier, La transformada Z y La transformada de Fourier de tiempo discreto (DTFT, por sus siglas en Inglés). Pero, ninguno de estos análisis matemáticos se puede desarrollar de manera eficiente en computadoras[1], solamente la DFT resuelta eficientemente con la codificación

[1] Nos referimos a computadoras, microcontroladores, DSP's, circuitos integrados y dispositivos electrónicos.

de la FFT. Es una de las herramientas más importantes que pueden ser desarrolladas en tarjetas DSP.

1.2 TRANSFORMADA DISCRETA DE FOURIER.

1.2.1 CÁLCULO DIRECTO DE LA DFT.

La definición de la DFT viene dada por:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k = 0,1,2 \cdots N-1 \quad (1-1)$$

$$W_N = e^{-j2\pi/N} \quad (1-2)$$

Donde:

X[k]: son los coeficientes de la DFT.

x[n]: es la secuencia finita a transformar.

W_N: son las funciones bases con periodo N.

A partir de la ecuación (1-1) se calcula en forma directa la DFT, uno de los métodos se enfoca en codificar la ecuación (1-1) en un programa, como dos bucles anidados de operaciones escalares, e incluyen una multiplicación compleja y una suma o acumulación compleja. Otro método interpreta la ecuación (1-1) como una única multiplicación matricial [3], una matriz contiene la secuencia de entrada *x[n]* y la otra las funciones bases W_N.

El primero de los métodos utiliza dos bucles anidados, con la sumatoria de $x[n]*W_N^{nk}$ en el bucle interior y utilizando el índice *k* en bucle exterior. El programa *dft_fwd.m* (anexo A1.1.1) realiza la DFT de una secuencia utilizando el método anterior. El

siguiente método utiliza la multiplicación de los vectores *x[n]* y W_N, para formar una matriz, el programa *dft.m* (anexo A1.1.2) es el que ejecuta la DFT en forma matricial. Ahora se hará la evaluación de los algoritmos en términos de el tiempo de ejecución y de el número de operaciones en punto flotante.

El programa *dft_fft_t.m* (anexo A1.1.3), genera los valores de tiempo de ejecución para los métodos de cálculo directo de la DFT y la FFT, para diferentes longitudes (N) de una secuencia. Lo que se pretende con estos valores es comparar: ¿Cuál método o algoritmo es el que se va usar?, ¿Cuál es el más rápido?, ¿Cuál seria inconveniente utilizar?.

En la Figura 1.1 se puede observar que el algoritmo de la FFT para valores grandes de N es mucho más rápido que los demás, el método matricial que para valores pequeños de N puede ser utilizado convenientemente.

Los tiempos de ejecución de la Figura 1.1, fueron evaluados por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *clock y etime* de *MATLAB.*

El método de los bucles, presenta el mayor tiempo de ejecución para los valores grandes de N, esto es por que se utilizan lazos *for* los cuales en *MATLAB* (se pudo utilizar código ejecutable creado en C, C++ o ensamblador para evaluar los algoritmos) son menos eficientes. La idea básica con la Figura 1.1 es mostrar la ventaja de velocidad que proporciona la transformada rápida de Fourier y el decidir dependiendo de los valores de la DFT cual método utilizar.

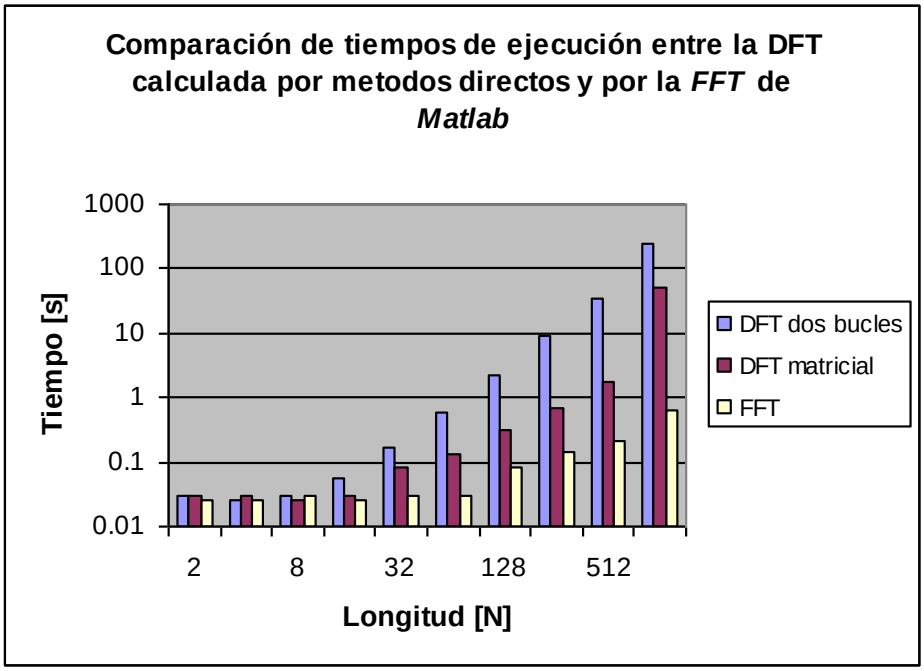

Figura 1.1 Comparación de tiempos de ejecución entre diferentes métodos de cálculo de la DFT. Estos tiempos de ejecución, fueron evaluados por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *clock y etime* de *MATLAB*.

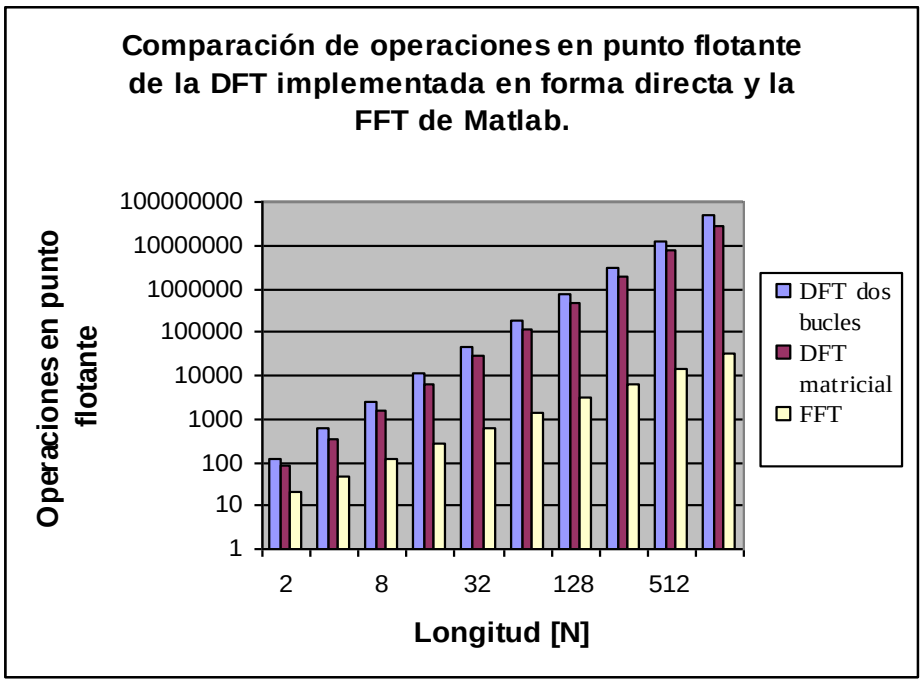

Figura 1.2 Comparación de operaciones en punto flotante entre diferentes métodos de cálculo de la DFT. Estas operaciones en punto flotante, fueron evaluadas por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *flops* de *MATLAB*.

A partir de la definición de la DFT ecuación (1-1), ya que, la DFT tiene N valores y cada valor se calcula a partir de N valores de entrada, se requerirá del orden N^2 sumas y multiplicaciones en punto flotante[2], la Figura 1.2 muestra la comparación entre operaciones en punto flotante para los métodos anteriormente mencionados.

Observando la Figura 1.2, notamos que el algoritmo de la FFT además de ser el mas rápido también es el que menos operaciones necesita, le sigue el método directo de la DFT en forma matricial y el que más operaciones requiere es el método directo de la DFT en forma de dos bucles.

A medida la longitud de la secuencia aumenta las operaciones en punto flotante también aumentan, en mayor proporción para uno que para otro método. Esto podría ser uno de los indicadores de que método seria conveniente utilizar.

Otro método directo para calcular la DFT es el algoritmo de *Goertzel*, el cual requiere de N^2 operaciones, pero en algunas ocasiones se aproxima a la mitad el número de multiplicaciones, con respecto a las utilizadas por los otros métodos directos (ejemplos el de dos bucles y matricial). La forma para calcular la DFT de longitud N, por este método se basa en la evaluación de un polinomio [3].

Debido a la naturaleza recursiva que presenta la evaluación de dicho polinomio, también se puede evaluar como un filtro con respuesta al impulso infinita (IIR, por sus siglas en Inglés) de primer orden con un polo complejo y una entrada formada con la secuencia de datos en orden inverso. El valor de la DFT es la salida del filtro después de N iteraciones.

[2]Punto flotante es una de las dos formas más común para almacenar números en una computadora. Usa una forma de notación científica, en donde una mantisa es elevada a un exponente.

La ventaja del método de *Goertzel*, radica [2] en que se logra una reducción en el número de operaciones, convirtiendo el filtro de primer orden en un filtro de segundo orden.

Existen funciones en *MATLAB* que nos ayudaran a codificar la DFT utilizando el método de *Goertzel*. Estas funciones son *polyval* y *filter*.

El archivo *polyb.m* (anexo A1.1.5) codifica la DFT utilizando la función *polyval*[3] y la evalúa para N puntos en un bucle. Mientras tanto el archivo *polym.m* (anexo 1.1.6) implementa la DFT utilizando también la función *polyval* pero en una forma que no utiliza bucle para evaluar los N puntos, en su lugar se ocupa un vector. Otra forma es utilizar la función *filter*[4], el archivo *fildft.m* (anexo A1.1.7) codifica dicha solución.

Una comparación entre diferentes métodos de cálculo de la DFT utilizando el método de *Goertzel*, tanto en tiempo de ejecución como en número de operaciones por segundo en punto flotante se presentan en las Figura 1.3 y Figura 1.4 respectivamente.

En la Figura 1.3 se observa el tiempo de ejecución de la DFT codificada con la función *polyval* en forma matricial es más rápida que los otros métodos, pero la FFT sigue siendo todavía mucho más rápida. En cuanto a la Figura 1.4 se puede observar una cierta ventaja en el número de operaciones en punto flotante, en los métodos que utilizan la función *polyval* contra la FFT y *filter*.

Ninguno, de los métodos directos anteriormente mencionados logra alcanzar la velocidad de la FFT, pero es importante mencionarlos, para hacer notar la importancia de los algoritmos que codifican la FFT.

[3] Función de MATLAB que evalúa un polinomio: polyval(P,X), P es un vector de longitud N-1 en el cual están los coeficientes de un polinomio evaluado en X. Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)

[4] filter(B,A,X), filtra los datos de X con un filtro descrito por los coeficientes A y B. El filtro es de la "Forma Directa II Transpuesta" , implementa la ecuación diferencia: a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb) -a(2)*y(n-1) - ... - a(na+1)*y(n-na)

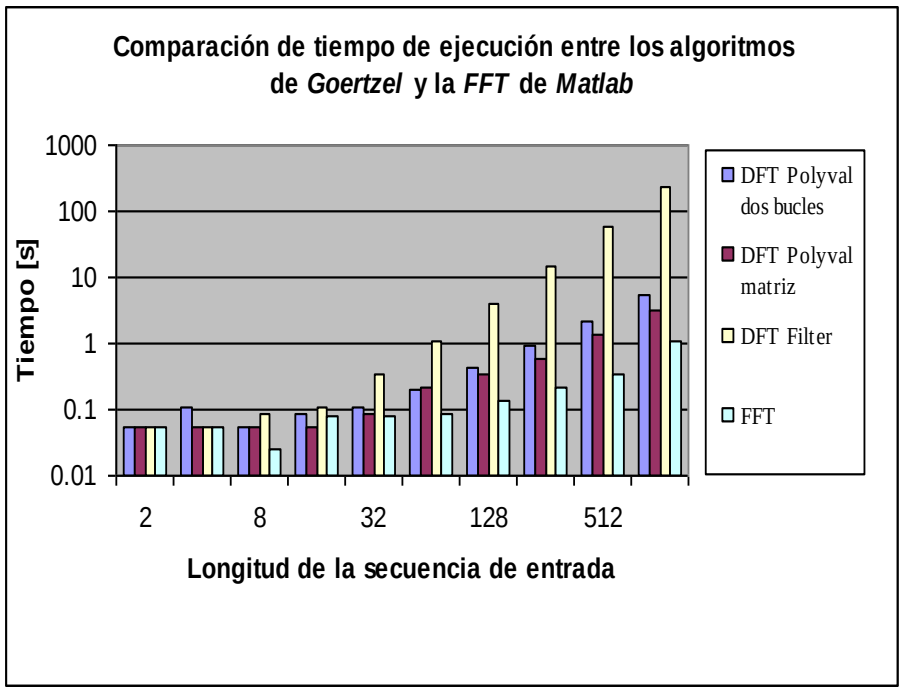

Figura 1.3 Comparación de tiempo de ejecución entre diferentes métodos de cálculo de la DFT mediante el método directo de *Goertzel*. Estos tiempos de ejecución, fueron evaluadas por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *time* y *etime* de *MATLAB.*

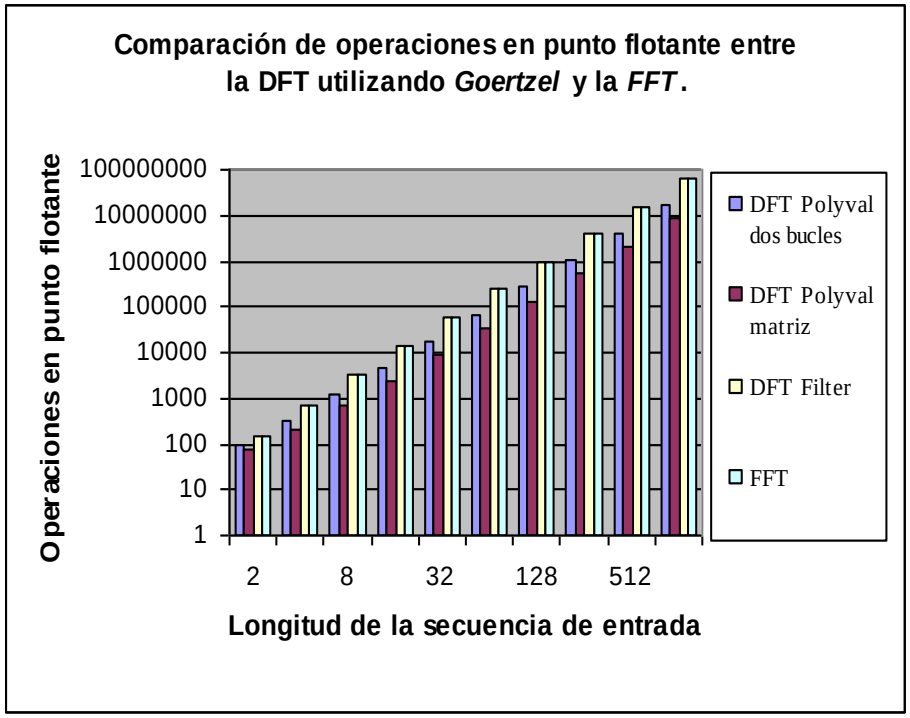

Figura 1.4 Comparación de operaciones en punto flotante entre los diferentes métodos de cálculo de la DFT mediante el método directo de *Goertzel*. Estas operaciones en punto flotante, fueron evaluadas por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *flops* de *MATLAB.*

1.2.2 TRANSFORMADA RÁPIDA DE FOURIER.

Como se ha mencionado anteriormente la FFT es un conjunto de algoritmos, que desarrollan de manera eficiente la DFT. Los cuales se basan en la descomposición de la DFT, este principio fundamental fue descrito por Cooley y Tukey en su publicación [1], de ahí es que existen un conjunto de algoritmos conocidos como la FFT de Cooley-Tukey. Estos algoritmos se basan en la eliminación de cálculos redundantes, que se hacen al evaluar la DFT directamente con la ecuación (1-1). Esto se logra descomponiendo en factores la longitud N y calculando múltiples DFT de orden menor cuyas longitudes serán iguales a la de los factores. Ya que, se permite cualquier descomposición factorial de la longitud N, dichos factores podrían ser iguales, y en tal caso $N=R^M$, donde R se denomina raíz y a la FFT correspondiente se le conoce como FFT de raíz R. A partir de como se hace la descomposición, así surgirá un algoritmo para realizar la FFT, por ejemplo FFT de raíz mixta, FFT de factor primo, etc. Además, fundamentos de la teoría de números [6], como: congruencias y teoremas de números que pueden lograr hacer algoritmos que eliminen el *"twiddle factor"*[5] o alcancen su mayor eficiencia, la cual es descomponer la longitud N en el máximo número de factores.

Para el alcance de este documento la FFT de Cooley-Tukey es suficiente para realizar de manera eficiente la DFT.

Entre los algoritmos mas conocidos se encuentran la FFT de raíz-2 y la FFT de raíz-4. Los métodos de convolución por bloques utilizan el algoritmo de la FFT y por medio de la simulación en Matlab se puede obtener los resultados de dichos métodos, y con esta idea se intentará desarrollar los programas que se puedan ejecutar en la DSP56L811EVM.

[5] Exponencial que multiplica la secuencia impar, resultante de la descomposición de la DFT en una secuencia par e impar. Dicho factor no es un cálculo propio de la DFT pero es necesario en las operaciones. Eliminando este factor se puede lograr más rapidez en la FFT.

Haciendo referencia al método de Cooley y Tukey para calcular la FFT, ellos dedujeron que la DFT de longitud N de *x[n]* puede calcularse en dos partes a partir de dos DFT's de longitud N/2, una DFT para los términos pares de *x[n]* y otra DFT multiplicada por un factor exponencial para los términos impares. Ellos nombraron esta forma particular de calcular la DFT como diezmado en tiempo (DIT, por sus siglas en Inglés), debido a que la entrada *x[n]* se divide en dos partes, una con los términos pares y la otra con los términos impares (ver anexo A1.3.1). Existe un factor exponencial que se conoce como "*twiddle factor"*, y dicho factor es parte de una operación que no es propiamente de la DFT, pero que es necesaria para considerar el desplazamiento en el tiempo. Invirtiendo los papeles de *x[n]* y *X[k]* (ver anexo A1.3.2), se obtiene una organización llamada diezmado en frecuencia (DIF, por sus siglas en Inglés).

Tanto la FFT DIT como la FFT DIF, definen una DFT de longitud 2^M en términos de dos DFT's de longitud 2^{M-1} y estas a su vez son evaluadas en términos de cuatro DFT's de longitud 2^{M-2}, este proceso se repite sucesivamente hasta que la longitud de DFT es igual a longitud de cada muestra; es posible evaluar en M pasos la DFT original, sin hacer evaluación directa de una DFT. Esta formulación es perfecta para programación recursiva, en la que un programa se llama a sí mismo.

En MATLAB se ha codificado la FFT DIT. Utilizando la propiedad de recurrencia, se ha efectuado la descomposición de la secuencia de entrada, posteriormente, hacer la síntesis del resultado de la descomposición. El archivo *dftdit2.m* (anexo A1.2.2), codificada la FFT DIT. La misma formulación de recurrencia se ha utilizado para codificar la FFT DIF. El archivo *dftdif2.m* (anexo A1.2.3), realiza dicha FFT.

Para datos haciendo una comparación entre el tiempo de ejecución y número de operaciones por segundo en punto flotante, obtenemos las Figuras 1.5 y 1.6, la FFT DIT y la FFT DIF respectivamente.

Aunque existen variaciones entre los tiempos de ejecución entre los dos métodos, se puede observar de la Figura 1.5, que a medida la longitud de la secuencia de entrada aumenta la diferencia tiende a ser menor, pudiéndose destacar el caso de N=32, en donde, el tiempo de ejecución es igual para los dos métodos. También se puede notar que para una longitud de N= 1024, el tiempo de ejecución, es menor a 1 segundo, lo que no se logro con los métodos directos para la DFT.

Observando la Figura 1.6 podemos ver que tanto la FFT DIT como la FFT DIF tienen el mismo número de operaciones en punto flotante, también para el caso de N=1024, las operaciones son mucho menores que las utilizadas por los métodos directos.

Recapitulando se deduce que cuando la DFT es calculada por los métodos directos el tiempo de ejecución es aproximadamente N^2, mientras que para la FFT es de $Nlog_2N$, lo que significa que la DFT es rápidamente calculada con la FFT.

El anexo A1.3 analiza en mayor detalle los métodos de cálculo de la DFT utilizando los algoritmos DIT y DIF, pues a partir de estos, se desarrollan los programas que realizan la FFT en la DSP56L811EVM.

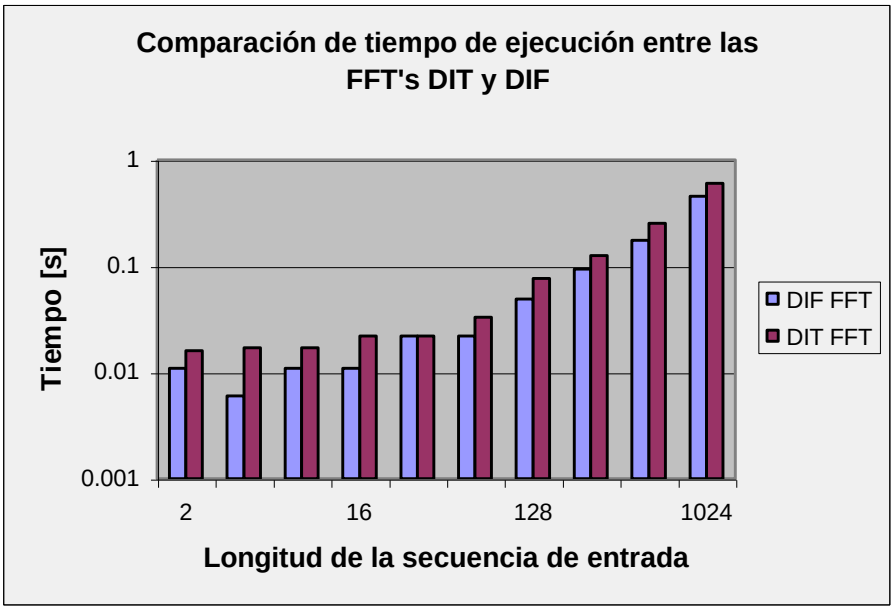

Figura 1.5 Comparación de tiempo de ejecución entre la FFT DIT y la FFT DIF. Estos tiempos de ejecución, fueron evaluadas por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *time* y *etime* de *MATLAB.*

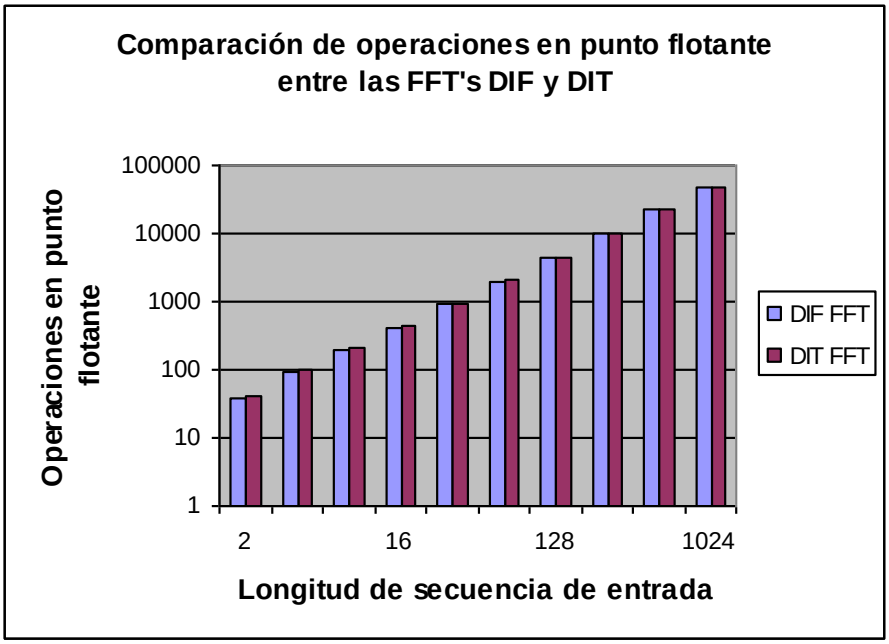

Figura 1.6 Comparación de operaciones en punto flotante entre la FFT DIT y la FFT DIF. Estas operaciones en punto flotante, fueron evaluadas por una computadora con microprocesador *Pentium* 150 Mhz, con 32 Mbytes de *RAM*. Utilizando la función *flops* de *MATLAB.*

1.3 CONCLUSIONES.

- Existen muchas formas de calcular la DFT, pero eficientemente son muy pocas, esencialmente la FFT es un conjunto de algoritmos que resuelven la DFT de manera eficiente, a lo que me refiero es que existe no solo un programa o algoritmo que resuelve la DFT sino que son varios que eficazmente calculan la DFT.
- La idea principal de este capítulo es hacer notar que es necesario una correcta compresión de los conceptos antes de incorporarlos en una implementación. Por tanto, es necesario incorporar conceptos, teoremas y propiedades fundamentales de la DFT para poder llevar a cabo un buen diseño del sistema que deseamos implementar. Pues, en dicho diseño la DFT juega un papel muy importante, o aun más, es una parte esencial.
- Es muy importante dejar claro que la FFT no es otra transformada, pues es, la manera eficiente de calcular la DFT, la cual es un periodo de las series de Fourier, cuyos coeficientes son muestras equiespaciadas de la transformada de Fourier en tiempo continuo.
- Otra parte importante de este capítulo son las mediciones del tiempo de ejecución y las operaciones en punto flotantes, son indicadores de la eficacia y rapidez de un algoritmo, como también, son los que determinan los alcances y limitaciones que se puede tener con un determinado *hardware*.

CAPÍTULO II.

DISEÑO DE FILTROS DIGITALES,

MÉTODO PARKS-McCLELLAN.

2.1 INTRUDUCCIÓN.

El diseño de filtros digitales es la orilla del mar cósmico de teorías y técnicas, que se utilizan en DSP, existen diferentes tipos de filtros digitales, pero concentraremos nuestra atención en los filtros discretos con respuesta al impulso finita(FIR, por sus siglas en Inglés). Existen también varios métodos de diseño de estos tipos de filtros. Por ejemplo el más elemental, un sistema promediador, pasando por el método de la ventana [2] y [5]*(Bartlett, Hamming, Hanning, Blackman y Kaiser)* y culminando con métodos de diseño de filtros óptimos, en este trabajo de graduación se presenta una de las muchas técnicas de diseño de filtros óptimos, nos referimos a una de las más populares, el método de Parks-McClellan [8], el cual utiliza el algoritmo de cambio de Remez [9].

Los filtros digitales de fase lineal con respuesta al impulso de duración finita (FIR), tienen varias propiedades importantes las cuales los hacen atractivos para aplicaciones de DSP. Entre estas características tenemos: la fase lineal exacta, la ausencia de algunos problemas de estabilidad encontrados en los filtros de respuesta al impulso de duración infinita (IIR, por sus siglas en Inglés) y la disponibilidad de eficientes métodos de diseño iterativos. El problema para estos filtros consiste como indicar aproximadamente una forma de magnitud idealizada en el sentido de la ponderación de Chebyshev (min-max) – criterio de error –. El problema del diseño con el criterio de Chebyshev con frecuencia de corte ha sido resuelto en el caso del filtro paso bajo, y el algoritmo de Remez fue mostrado para ser un medio muy eficiente para el cálculo de la mejor aproximación [10].

Un método para el diseño de filtros digitales de fase lineal con respuesta al impulso de duración finita (FIR) fue presentado por J. H. McClellan y T. W. Parks en su articulo [8], para tratar de una manera única, los cuatros tipos de casos posibles de filtros – tipo I, II,

III y IV –. Este método demuestra como reducir cada caso a la forma apropiada a fin de que el algoritmo de cambio de Remez, pueda ser utilizado como la mejor estrategia para calcular la respuesta en frecuencia deseada. El resultado es un método flexible y rápido para el diseño de filtros de fase lineal FIR.

El objetivo de este capítulo es mostrar el diseño de varios filtros FIR de fase lineal, por ejemplo; paso bajo, pasa banda, rechaza banda, multibanda, transformador Hilbert y diferenciador. De los cuales esencialmente utilizaremos los coeficientes calculados con la función Remez de *Matlab* para realizar la convolución.

2.2 DISEÑO DE FILTROS FIR, MÉTODO PARKS-McCLELLAN.

2.2.1 FORMULACIÓN DE LA APROXIMACIÓN DEL PROBLEMA Y SU SOLUCIÓN.

Un filtro digital FIR de tamaño N con respuesta al impulso {h(k)}, k = 0,1,…,N-1, posee una respuesta en frecuencia, la cual, es la transformada *z* evaluada en el círculo unitario ($z=e^{j\omega}$):

$$H(e^{j\omega}) = H(z)|_{z=e^{j\omega}} = \sum_{k=0}^{N-1} h(k)e^{-j\omega k}. \quad (2-1)$$

A partir de un filtro de fase lineal generalizada denotamos que la respuesta en frecuencia puede ser escrita como:

$$H(e^{j\omega})=G(e^{j\omega})e^{\,j(A+B\omega)} \quad (2\text{-}2)$$

Donde *A* y *B* son constantes y $G(e^{j\omega})$ es una función real – posiblemente bipolar – de ω. Observe que $G(e^{j\omega})$ no es la magnitud de la respuesta en frecuencia ya que puede ser negativa, pero $|G(e^{j\omega})|$ si es la magnitud.

En esta sección mostraremos del porque escribir la función $G(e^{j\omega})$ (ver anexo 2.1) en la forma general:

$$G(e^{j\omega}) = Q(e^{j\omega})P(e^{j\omega}) \quad (2-3)$$

La ecuación (2-3) conduce a una información muy clara y breve de las condiciones necesarias y suficientes para la aproximación de *Chebyshev* mejor optimizada, así también muestra un método preciso para calcular esta buena aproximación.

Primero, formulamos la aproximación del problema del diseño del filtro de fase-lineal. Luego debemos normalizar a uno la frecuencia de muestreo; a continuación la respuesta en frecuencia, es completamente especificada en el intervalo [0,1/2]. Ahora, dado un subconjunto $\Im$ compuesto de [0,1/2], una función deseada $D(e^{j\omega})$ que es definida y continua en $\Im$, una función positiva $W(\omega)$ que es definida y continua en $\Im$, y uno de los cuatro tipos de filtros de fase lineal – tipo I, II, III y IV – el cual es determinado de la forma de $G(e^{j\omega})$, entonces buscamos minimizar: $\left\|E(e^{j\omega})\right\| = maxW(\omega)\left|D(e^{j\omega}) - G(e^{j\omega})\right| \quad (2-4) \quad \omega \in \Im$, por selección de $G(e^{j\omega})$.

La formulación del problema en un subconjunto compuesto de [0,1/2] permite al diseñador hacer diseños de filtros multibanda, ya que, las especificaciones de la respuesta en frecuencia se pueden realizar en el intervalo [0,1/2]. Por ejemplo:

$$D(e^{j\omega}) = \begin{cases} 0, & \omega \in [0,0.10] \\ 1, & \omega \in [0.2,0.35] \\ 0, & \omega \in [0.425,0.5] \end{cases} \qquad (2-5)$$

La ecuación (2-5) especifica un filtro pasa banda, en el conjunto de $\Im$, el cual es: $\Im = [0,0.1]U[0.2,0.35]U[0.425,0.5]$ y $D(e^{j\omega})$ es continua en $\Im$, ya que la región de transición ha sido insertada en las bifurcaciones de $D(e^{j\omega})$. Cuando en (A2-7) $G(e^{j\omega})$ es una combinación lineal de cósenos, el teorema de la alternación establece las condiciones necesarias y suficientes que debe satisfacer a la mejor aproximación.

Sin embargo, en algunos casos el teorema de la alternación no puede ser aplicado directamente.

No obstante, usando (2-3) podemos formular una aproximación equivalente al problema, por lo tanto el teorema de la alternación puede ser aplicado. Observe que:

$$W(\omega)\left|D(e^{j\omega})-Q(e^{j\omega})P(e^{j\omega})\right|=W(\omega)Q(e^{j\omega})\left|\frac{D(e^{j\omega})}{Q(e^{j\omega})}-P(e^{j\omega})\right| \quad (2-6)$$

Excepto posiblemente al final de los puntos, donde $Q(e^{j\omega})$=0. Así, (2-6) es verdadera en ℑ'C ℑ, donde ℑ' es un subconjunto compuesto de ℑ y $Q(e^{j\omega})$>0 en ℑ'. Dejando que las estimaciones sean: $\hat{D}(e^{j\omega})=D(e^{j\omega})/Q(e^{j\omega})$ y $\hat{W}(\omega)=W(\omega)/Q(e^{j\omega})$, ahora el problema llega a ser, el de minimizar: $\|E(\omega)\|=max\hat{W}(\omega)\left|\hat{D}(e^{j\omega})-P(e^{j\omega})\right| \quad (2-7) \quad \omega \in$ ℑ', por selección de $P(e^{j\omega})$, donde $P(e^{j\omega})$ es una combinación lineal de cósenos.

El problema no es exactamente el mismo porque el conjunto de ℑ ha sido reemplazado por el subconjunto ℑ'. Por ejemplo en (2-5), si la longitud de la aproximación del filtro fuera par, entonces $Q(1/2)=0$ y ℑ podría ser modificada para obtener ℑ'=[0,0.1]U[0.20,0.35]U[0.425,(0.5-ε)], donde ε es un pequeño numero positivo. De hecho, $D(e^{j\omega})$ podría ser especificada a cero en cualquier frecuencia donde $G(e^{j\omega})=0$. En otro caso, habrá un error fijo en la aproximación de *Chebyshev* el cual no puede ser reducido. Este es el caso cuando se trata de aproximar un diferenciador de banda completa con un filtro de longitud par porque $D(1/2)=1/2$, mientras $G(1/2)=0$. La unión de nuestra aproximación resulta de (2-5) porque podemos establecer un teorema de la alternación que abarca los cuatro casos de tipos de filtros.

2.2.2 TEOREMA DE LA ALTERNACIÓN.

Si $P(e^{j\omega})$ es una combinación lineal de r funciones coseno $(esto \quad es, P(e^{j\omega}) = \sum_{k=0}^{r-1} \alpha(k)\cos\omega)$ entonces una condición necesaria y suficiente es que $P(e^{j\omega})$ será la mejor aproximación de *Chebyshev* para una función continua $D(e^{j\omega})$ en $\Im$',esto es porque la función de error $\|E(\omega)\| = \hat{W}(\omega)\left|\hat{D}(e^{j\omega}) - P(e^{j\omega})\right|$ mostrará por lo menos $r+1$ frecuencias extremas en $\Im$'. Esto es, puntos ω_i, tales que $\omega_1 < \omega_2 .... < \omega_n < \omega_{n+1}$, $E(\omega_i) = -E(\omega_{i+1})$, $i = 1,2,...,n$, y $|E(\omega_i)| = maxE(\omega)$ $\omega \in \Im$'. Considere la aplicación del teorema de la alternación en el caso de un diferenciador de banda amplia. La respuesta en frecuencia normalizada de un diferenciador ideal es $H(e^{j\omega}) = j\omega$, así que $D(e^{j\omega}) = \omega$.

También para obtener un error el cual es proporcional a $D(e^{j\omega})$, hacemos $W(\omega) = 1/\omega$. Así $E(\omega) = 1/\omega \,/\, \omega - Q(e^{j\omega})P(e^{j\omega}) \,/$. La Figura 2.1 muestra la alternación de la curva de error para N = 16 (esto es, r = 8 y 9 frecuencias extremas). La tabla 2.1 muestra un resumen de este teorema para los cuatro casos que han sido considerados (ver anexo 2.1).

Digamos para concluir con el teorema de alternación, se presenta a continuación el siguiente ejemplo: Supongamos que deseamos examinar los polinomios $P(x)$ que se aproximan a la unidad para $-1 \leq x \leq -0.1$ y a cero para $0.1 \leq x \leq 1$. Consideremos tres de esos casos que se muestran en la figura 2.2.Cada uno de esos polinomios es de grado cinco, y se desea determinar cual de ellos si es que existe, satisface el teorema de alternación. Además, los subconjuntos cerrados del eje real x a los que se refiere el teorema son $-1 \leq x \leq -0.1$ y $0.1 \leq x \leq 1$. Ponderemos igual los errores en ambas regiones, es decir $W_p(x) = 1$. De acuerdo con el teorema de alternación, el polinomio de quinto grado óptimo debe tener por

lo menos siete alternaciones del error en las regiones correspondientes al subconjunto cerrado F_p. $P_1(x)$ tiene sólo cinco alternaciones, tres en la región $-1 \leq x \leq -0.1$ y dos en la región $0.1 \leq x \leq 1$. Pues el punto con pendiente cero no toca la línea de puntos, y no es una alternación ya que, no alcanza el valor extremo positivo. Ya que, el teorema de alternación especifica que las alternaciones adyacentes deben cambiar de signo, por tanto el valor extremo x = 1, no puede ser tampoco una alternación.

Ahora, el polinomio $P_2(x)$, también tiene cinco alternaciones y por tanto no es óptimo. El valor x = 0.1 no es un valor extremo positivo, pues la alternación previa que esta en x = -0.1 es un valor extremo negativo, por lo que necesitamos un valor extremo positivo para la siguiente alternación. Además, el primer punto con pendiente cero dentro del intervalo $0.1 \leq x \leq 1$ tampoco se puede contar, ya que es un valor extremo negativo, como x = -0.1, y no alternan el signo.

Y por último, $P_3(x)$ tiene ocho alternaciones las cuales son: todos los puntos de pendiente cero, x = -1, x = -0.1, x = 0.1 y x = 1. Como con ocho alternaciones se satisface el teorema de alternación, el cual especifica un mínimo de siete, $P_3(x)$ es el único polinomio de quinto grado que produce una aproximación óptima para esta región.

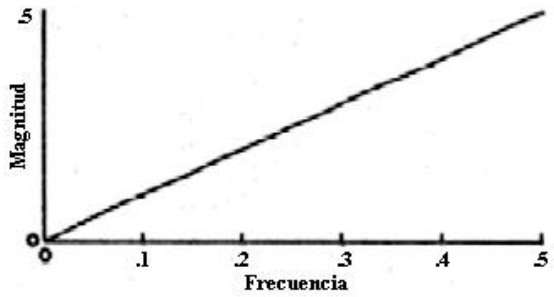

Figura 2.1 Ilustración del Teorema de la Alternación para un diferenciador de una longitud de 16. Pendiente de Error = 0.0136 con 9 frecuencias extremas

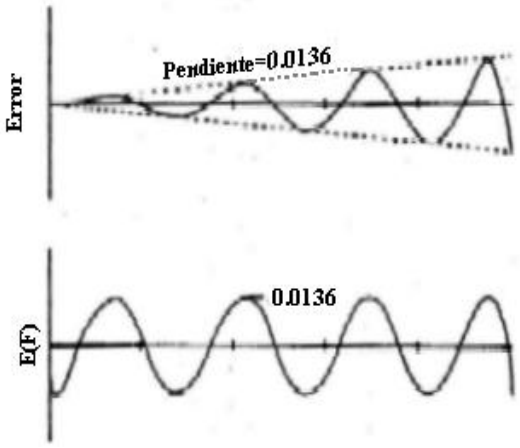

Figura 2.1(Continuación) Ilustración del Teorema de la Alternación para un diferenciador de una longitud de 16. Pendiente de Error = 0.0136 con 9 frecuencias extremas

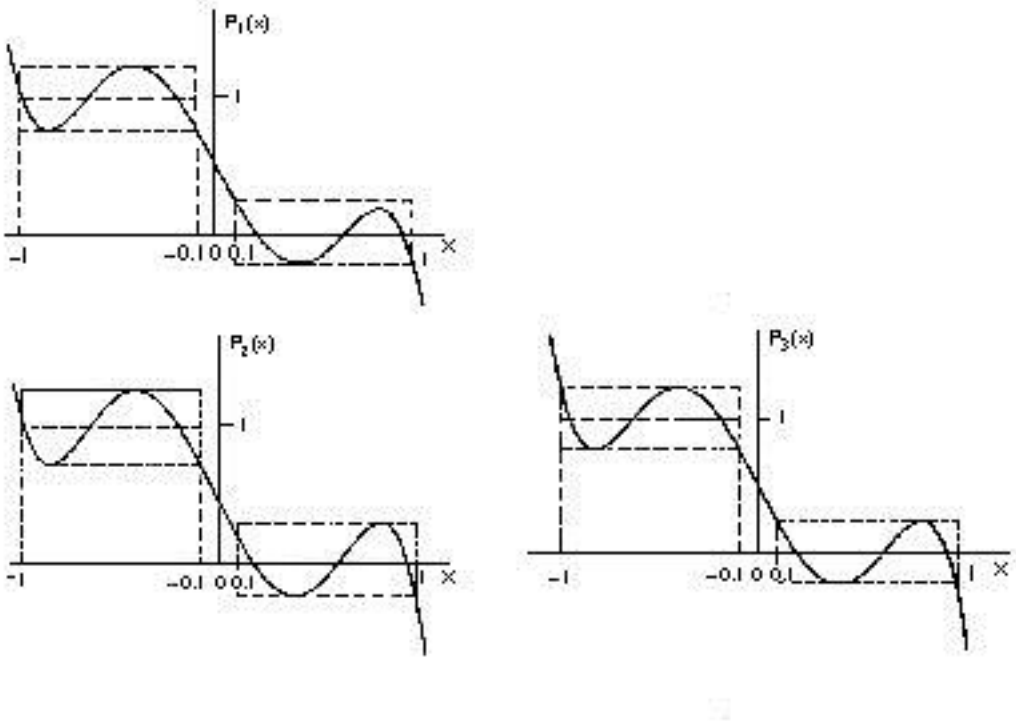

Figura 2.2 Ilustración del Teorema de la Alternación en el caso, de tres polinomios de grado cinco.

Tabla 2.1

SIMETRÍA / LONGITUD	SIMÉTRICA	ANTISIMÉTRICA
IMPAR	$G(e^{j\omega}) = \sum_{k=0}^{n} \hat{a}(k)\cos\omega k$ $n = \frac{N-1}{2}$, *al menos* $\frac{(N-1)}{2}+2$ *frecuencias extremas*	$G(e^{j\omega}) = \text{sen}\,\omega\sum_{k=0}^{n-1} \hat{c}(k)\cos\omega k$ $n = \frac{N-1}{2}, G(0) = G(1/2) = 0$ *al menos* $\frac{(N-1)}{2}+1$ *frecuencias extremas*
PAR	$G(e^{j\omega}) = \cos\frac{\omega}{2}\sum_{k=0}^{n-1} \hat{b}(k)\cos\omega k$ $n = \frac{N}{2}, G(1/2) = 0$ *al menos* $\frac{N}{2}+1$ *frecuencias extremas*	$G(e^{j\omega}) = \text{sen}\,\frac{\omega}{2}\sum_{k=0}^{n-1} \hat{d}(k)\cos\omega k$ $n = \frac{N}{2}, G(0) = 0$ *al menos* $\frac{N}{2}+1$ *frecuencias extremas*

2.2.3 CÁLCULO DE LA MEJOR APROXIMACIÓN.

El desarrollo anterior, no solamente unifica la teoría de aproximación de filtros de fase lineal, sino que también incita a unificar técnicas por computadora, tal como el algoritmo de cambio de Remez. Esto es fácil de ver, ya que, solo necesitamos un algoritmo general el cual haga aproximaciones cósenos, ya que, los otros tres casos pueden ser reducidos a este caso.

Así, que la estructura de un diseño de un programa general, consiste de una sección de entrada, formulación de la apropiada aproximación del problema, solución de la aproximación del problema usando el algoritmo Remez y cálculo de la respuesta al impulso de acuerdo (A2-1)-(A2-7). La Figura 2.3 muestra esta estructura en un diagrama de bloques.

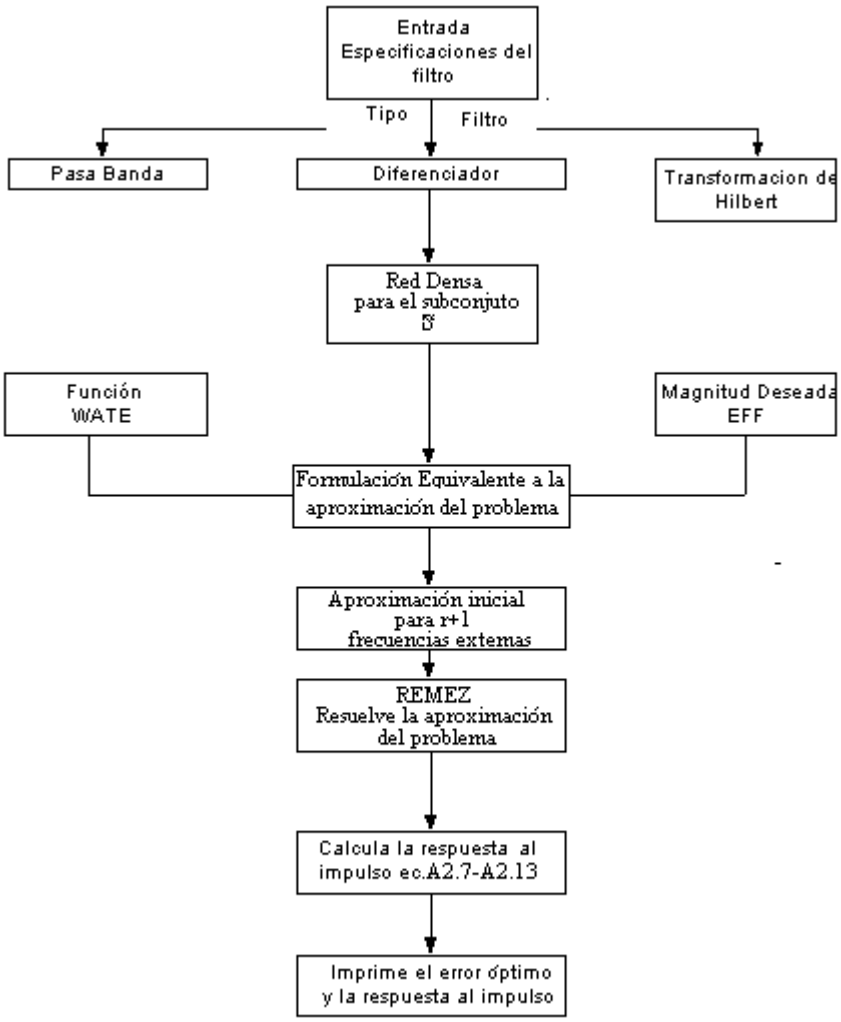

Figura 2.3 Diagrama de Bloques del diseño del programa.

2.2.4 EJEMPLOS DE DISEÑO.

Las Figuras de la 2.4 a 2.11 muestran ejemplos específicos del uso del diseño de programas para varios filtros típicos de interés. Para cada uno de estos filtros, se muestra un gráfico de la respuesta en frecuencia del filtro, en algunos casos se muestra tanto en escala lineal como en dB, pero en los casos Diferenciador y Hilbert se muestra el error. Dicho filtros fueron diseñados en *Matlab,* utilizando la función *remez.* Las Figura 2.4 es para un filtro pasa bajo con M = 24 (M, es el número de coeficientes). La Figura 2.5 es para un filtro pasa banda con M = 32.

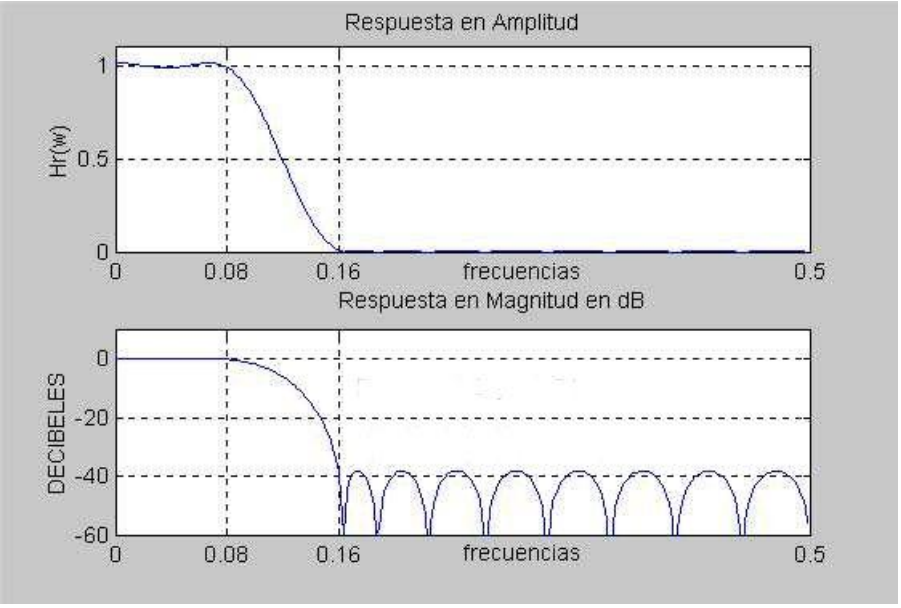

Figura 2.4 Filtro Pasa Bajo, M=24, frecuencia normalizada.

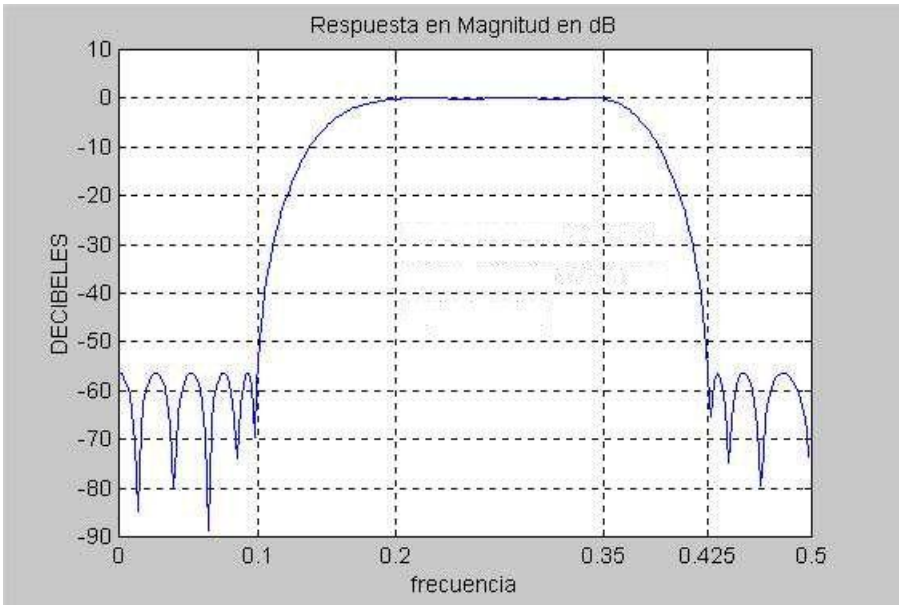

Figura 2.5 Filtro Pasa Banda, M=32, frecuencia normalizada.

La Figura 2.6 es para un filtro pasa banda con M = 50 en el cual el ancho es diferente en las dos bandas de rechazo. Así, el error pico en el limite superior del rechaza banda es más pequeño que el error pico en el limite inferior del rechaza banda. La Figura 2.7 es para un filtro rechaza banda con M=31 con igual ancho en ambas pasa banda.

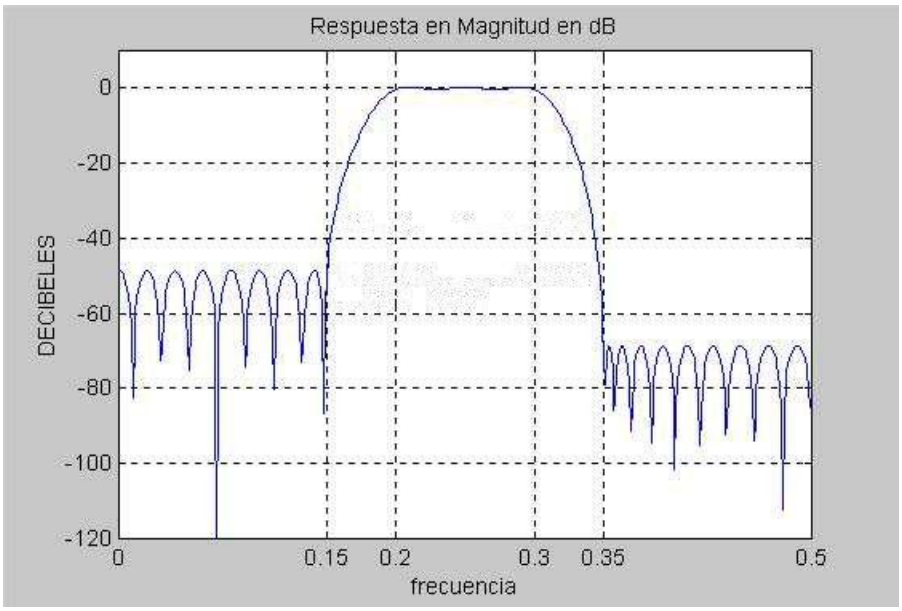

Figura 2.6 Filtro Pasa Banda, M=50, diferente atenuación en la banda de rechazo, frecuencia normalizada.

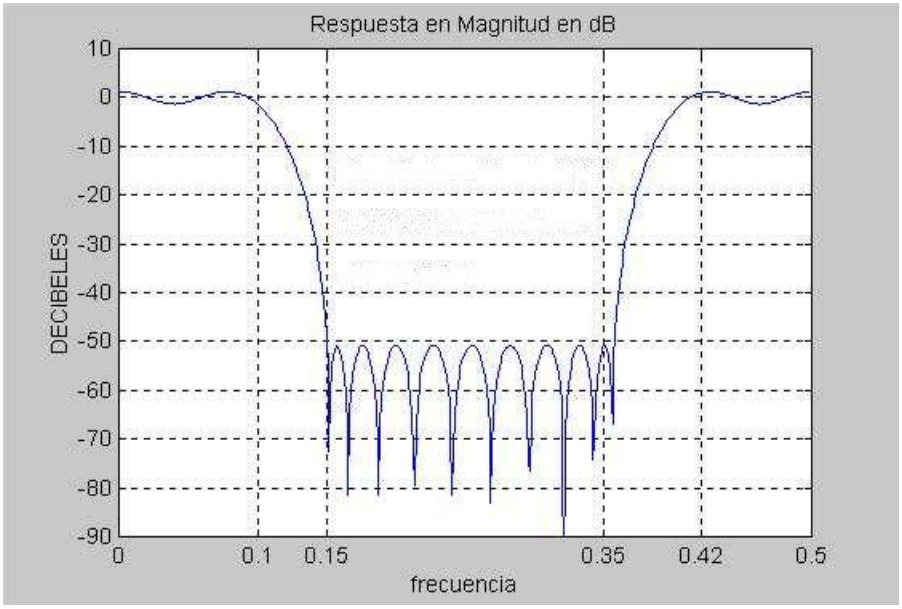

Figura 2.7 Filtro Rechaza Banda, M=31, frecuencia normalizada.

Para ilustrar la capacidad del programa se muestra un filtro multibanda, la Figura 2.8 muestra los resultados de un filtro de 5 bandas con M = 55, con tres rechaza banda y dos pasa banda. El ancho de cada una de las rechaza banda es distinto haciendo, la aproximación de la diferencia de error pico en cada una de estas bandas.

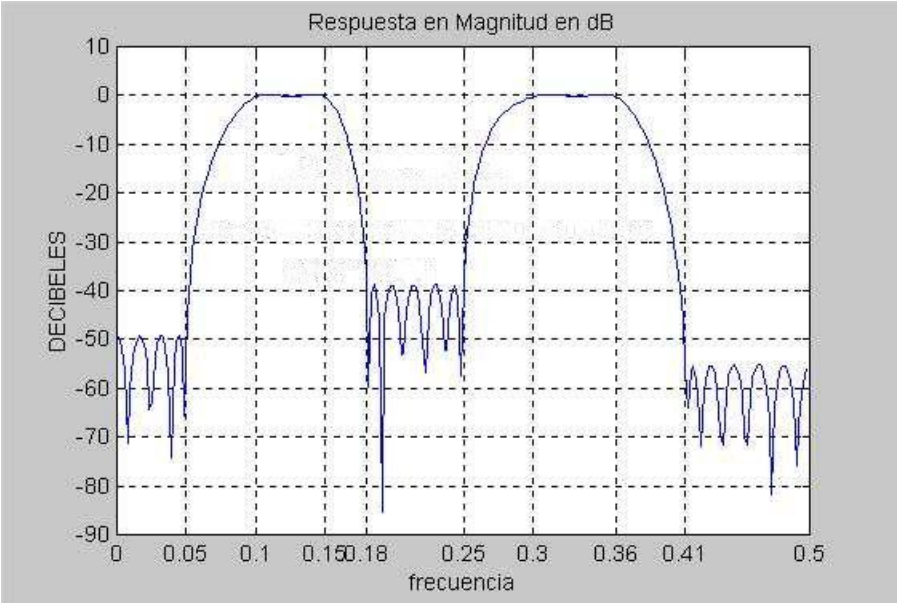

Figura 2.8 Filtro multibanda, M=55, diferente atenuación en las bandas de rechazo, frecuencia normalizada.

Las figuras 2.9 y 2.10 muestran ejemplos típicos de aproximaciones para un diferenciador y un transformador Hilbert. La Figuras 2.9 muestra los resultados para un diferenciador de banda completa con M = 32. La Figura 2.10 muestra los resultados para un transformador Hilbert, donde la frecuencia de corte superior es 0.5 y la frecuencia de corte inferior es 0.05. La aproximación del error pico es 0.0062 para el diferenciador y 0.02 para el transformador Hilbert.

Finalmente la Figura 2.11 muestran un ejemplo de un filtro pasa banda con M=128. Con una función de ancho arbitrario de la forma:

$$W(\omega)=\begin{cases}\dfrac{10}{1-9\omega}, & 0<\omega<0.1\\ 1, & 0.12<\omega<0.13\\ \dfrac{10}{9\omega-1.25}, & 0.15<\omega<0.25\\ 10, & 0.25<\omega<0.5\end{cases} \quad (2-8)$$

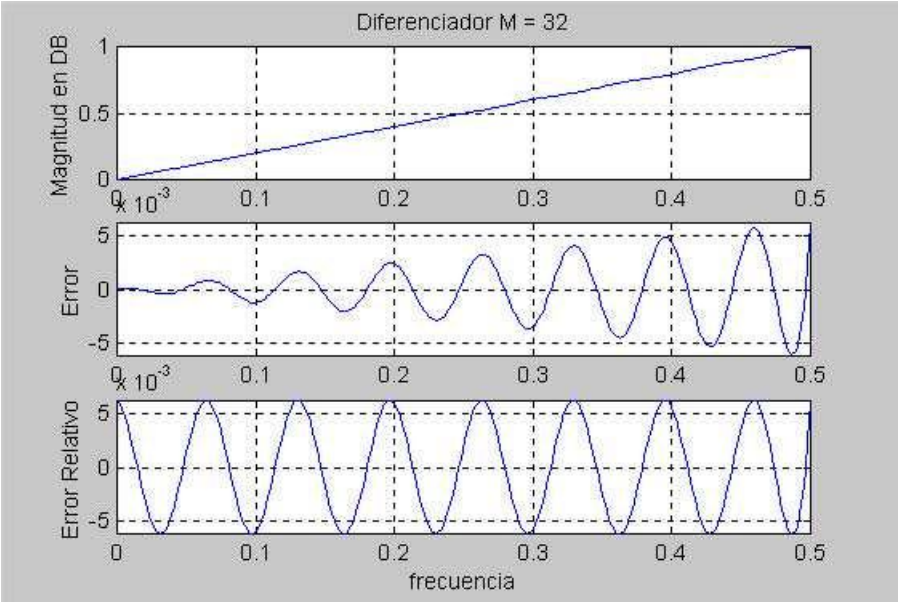

Figura 2.9 Filtro Diferenciador, M=32, Magnitud, Error y Error Relativo, frecuencia normalizada.

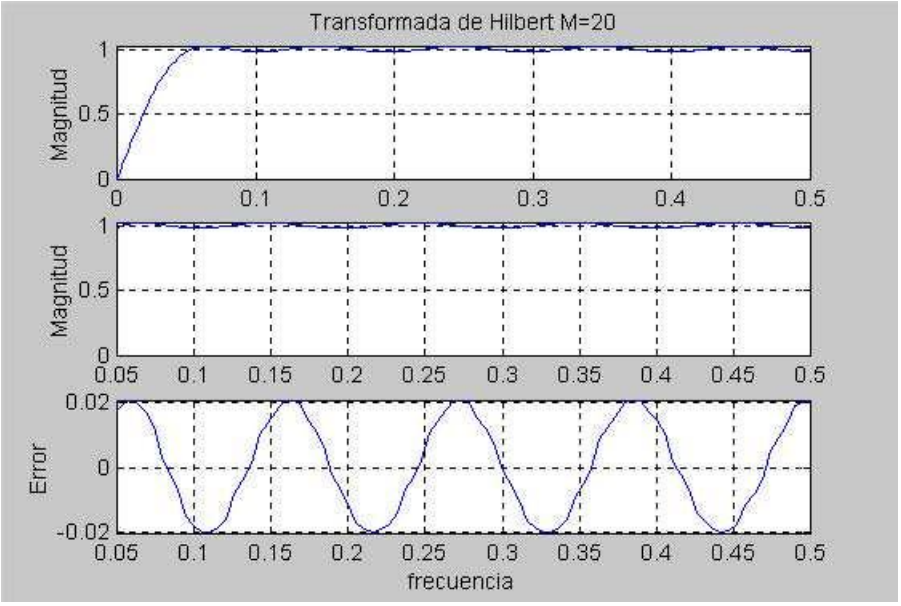

Figura 2.10 Filtro Transformada de *Hilbert*, M=20, frecuencia normalizada.

Así, el esquema de tolerancia es lineal en intervalo $0 < \omega < 0.1$ y $0.15 < \omega < 0.25$. El error en el limite de rechaza banda es 0.0005(-66dB), y el error pico es incrementado linealmente a 0.005(-46dB).

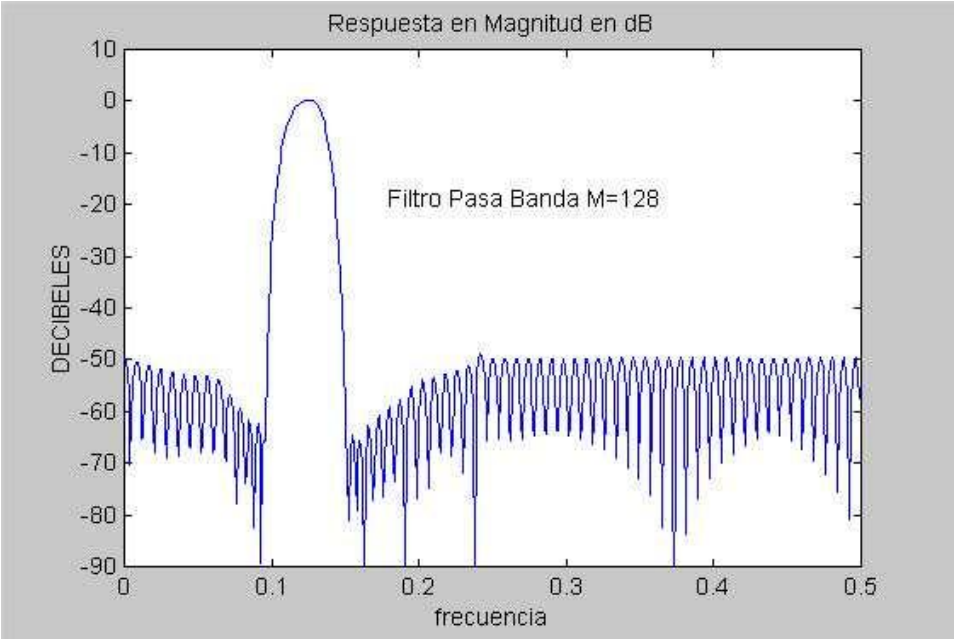

Figura 2.11 Filtro Pasa Banda, M=128, frecuencia normalizada.

En el anexo A2.2 encontraran los programas hechos en *Matlab*, para poder obtener cualquiera de las graficas anteriormente mostradas.

2.3 CONCLUSIONES.

- El propósito general del diseño de filtros FIR de fase lineal, es el de presentar la capacidad de diseño de una amplia variedad de filtros estándares; así como cualquier respuesta en magnitud deseada, la cual puede ser especificada por el usuario.
- La velocidad del algoritmo, así como, su generalidad hacen de este uno de los mas atractivos para una amplia variedad de diseños de aplicación. Y en consecuencia, se utilizan los coeficientes, para realizar la convolución, además, no es necesario un gran número de coeficientes para lograr obtener una muy buena aproximación de la respuesta en frecuencia deseada.

- Se ha presentado uno de los muchos métodos de diseño de filtros con respuesta al impulso de duración finita (FIR), pero dicho método tiene una gran ventaja, el de poder realizar filtros mediante un algoritmo que puede ser implementado eficientemente en una computadora. Además, tiene la cualidad de ser un método de diseño óptimo, lo cual, implica que el algoritmo permite la posibilidad de aproximar las características de respuestas en frecuencia arbitrarias.
- La complejidad y lo sofisticado del diseño de un filtro, no es inconveniente hoy en día, debido a que contamos con herramientas, como *Matlab* – o cualquier otro programa de análisis científico –, que además de simular los filtros, puede crear un archivo que contenga los coeficientes del filtro digital, para que posteriormente con dicho conjunto de datos se pueda implementar en un determinado hardware o computadora.
- Muchas preguntas pueden surgir, del porque diseñar o implementar filtros FIR, pero el diseño dichos filtros, nos da una ventaja o un criterio más, para decidir que tipo de filtro o método de diseño se debe realizar, a la hora de consideraciones prácticas.

CAPÍTULO III.

CONVOLUCIÓN POR BLOQUES, MÉTODOS SOLAPAMIENTO-SUMA Y SOLAPAMENTO-ALMACENAMIENTO.

3.1 INTRUDUCCIÓN.

La parte central de este trabajo de graduación, es este capítulo en el que se expondrá de manera simple y clara las teorías y los procedimientos para llevar a cabo el desarrollo de las técnicas solapamiento-suma y solapamiento-almacenamiento en una tarjeta de procesamiento digital de señales.

Este capítulo presenta tres importantes técnicas en DSP, el método solapamiento-suma, solapamiento-almacenamiento y la convolución usando la FFT. Ambos métodos de solapamiento se utilizan para dividir señales de gran longitud en pequeños segmentos para un fácil procesamiento [2]. Estos métodos de solapamiento, junto con la convolución usando la FFT hacen de esa combinación una forma eficiente de procesamiento, para realizar el desarrollo de filtros digitales en tiempo real.

La convolución usando la FFT se basa en la propiedad que la multiplicación en el dominio de la frecuencia corresponde a la convolución en el dominio del tiempo. La señal de entrada *x[n]* es transformada al dominio de la frecuencia usando la DFT *X[k]* (ver ecuación (1-1)), y dicho espectro es multiplicado por la respuesta en frecuencia del filtro, para luego regresar el resultado al dominio del tiempo usando la DFT inversa.

Estas técnicas básicas – convolución usando la DFT – anteriormente, no se les dio mucha importancia, ya que el tiempo requerido para calcular la convolución, usando la DFT era demasiado grande comparado con el tiempo para calcular la convolución por métodos convencionales.

Esto cambia, como se menciono en el Capítulo I en el año 1965 con el desarrollo de algoritmos eficientes para calcular la DFT. Al utilizar algoritmos como la FFT para calcular la DFT y la convolución – multiplicación en el dominio de la frecuencia – el resultado es

un método más rápido para calcular la convolución que los métodos convencionales en el dominio del tiempo. El resultado de la convolución es el mismo, solamente que el algoritmo ha cambiado por uno más eficiente.

Ya que obtenemos la convolución usando la FFT, se puede emplear en combinación con los métodos solapamiento-suma y solapamiento-almacenamiento, con el objetivo de realizar y desarrollar filtros digitales en tiempo real en la DSP56L811EVM.

Otro de los temas que juega un papel de gran importancia es la convolución circular [2] y [3] , debido a su utilización en la convolución por bloques (método solapamiento-suma y solapamiento-almacenamiento), pues se desea realizar un sistema lineal e invariante en el tiempo (LTI, por sus siglas en inglés) entonces se debe determinar que valores de la convolución circular corresponden con la convolución lineal, o cuando esos tipos de convolución son iguales.

La convolución circular esta relacionada también con la DFT, se debe determinar que valor de N puntos de la DFT serán necesarios para obtener una convolución lineal.

Todo lo anterior se debe tomar muy en cuenta en el desarrollo de filtros digitales en la DSP56L811EVM, pues los programas que esencialmente se deben realizar son: la FFT para N puntos, la multiplicación de espectros, la IFFT y los métodos de convolución por bloques.

Como la convolución lineal se puede realizar en el dominio del tiempo en el anexo A3.3 se presentan programas realizados en el lenguaje ensamblador del núcleo DSP56800, que desarrolla filtros digitales FIR, utilizando el Capítulo II para el diseño de filtros digitales y la estructura FIR [2] para realizar la convolución en el dominio del tiempo. Es necesario dejar definido que sí, es posible el desarrollo de filtros digitales en el *hardware* utilizado y con el fin de promover el diseño, desarrollo de otras técnicas de DSP y el

realizar mejoras a los diseños de los programas que se pretenden realizar en este trabajo de graduación.

3.2 MÉTODO SOLAPAMIENTO-SUMA.

Hay muchas aplicaciones en donde una señal debe ser filtrada en segmentos. Por ejemplo, audio digital de alta fidelidad requiere una taza de transferencia de datos cerca de los 5 Mbytes/min, mientras que video digital requiere cerca de 500 Mbytes/min. Con estas altas tazas de transferencias de datos es común para las computadoras tener insuficiente memoria para simultáneamente mantener la señal de entrada y ser procesada. También hay sistemas que procesan segmento por segmento, debido a que hay que operar en tiempo real. Por ejemplo, señales para telefonía, no pueden tener un retardo por más de unos pocos cientos de milisegundos, limitando la cantidad de datos que están disponibles para ser procesados en un instante. Y en otras aplicaciones podría ser que el "procesamiento" requiera que la señal deba ser segmentada. Un ejemplo de esto es la convolución con la FFT, uno de los temas principales de este trabajo de graduación.

3.2.1 DESCRIPCIÓN DEL MÉTODO SOLAPAMIENTO-SUMA.

Con el fin de utilizar el método solapamiento-suma para el procedimiento de filtrado damos a continuación una descripción del método. Para empezar se denomina con ese nombre por el hecho de que las secciones filtradas se solapan y se suman al construir la señal de salida. Aunque por supuesto le pudieron haber puesto otro nombre como por ejemplo "salidas solapadas", pero lo importante no es el nombre, es el método o aun mucho más el concepto.

¿Por qué se produce ese solapamiento, tan mencionado? Se produce porque la convolución lineal de cada sección, con la respuesta al impulso del filtro, es en general mayor que la longitud de la sección. A manera de aclaración y para desarrollar el procedimiento se puede consultar [2] y [3]. Supongamos que la respuesta al impulso del filtro *h[n]* es de longitud P y la señal *x[n]* de longitud mucho mayor que la de P (ver figura 3.1), pero digamos que *x[n]* = 0 para $n < 0$. Y dicha secuencia *x[n]* se puede representar como una suma de segmentos de longitud finita L es decir:

$$x[n] = \sum_{r=0}^{\infty} x_r[n - rL] \quad (3-1)$$

siendo

$$x_r[n] = \begin{cases} x[n + rL], & 0 \le n \le L-1, \\ 0, & en\ el\ resto \end{cases} \quad (3-2)$$

Como se observa de la ecuación (3-1) *x[n]* esta formada por la suma de todas las secciones involucradas, desde 0 hasta infinito. La secuencia *x[n]* tiene una longitud inmensamente grande por lo que se ha tenido que seccionar para poder ser procesada. En cambio la ecuación (3-2) muestra las secciones de longitud finita $x_r[n]$ a las cuales se les puede aplicar un determinado procesamiento.

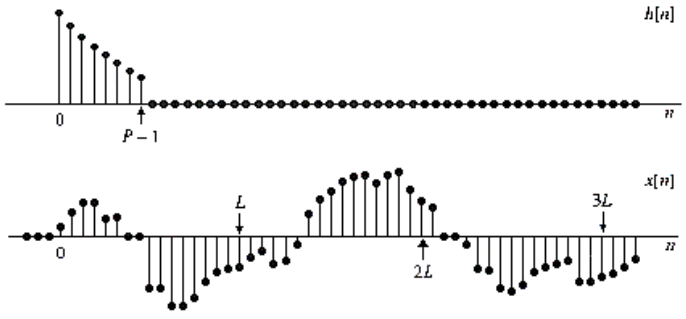

Figura 3.1 La respuesta al impulso de longitud finita *h[n]* y señal *x[n]* de longitud indefinida a filtrar.

Cabe mencionar que en el *hardware* se debe de utilizar un *buffer* para estar almacenando la secuencia *x[n]* en lo que se esta procesando una determinada sección $x_r[n]$ y a la vez devolviendo una secuencia de salida *y[n]*. Un *buffer* es una sección de memoria limitada a N localidades destinada para almacenar N cantidad de datos, puede ser lineal o modular.

La convolución es una operación lineal e invariante con el tiempo, entonces se deduce de la ecuación (3-1) que:

$$y[n] = x[n] * h[n] = \sum_{r=0}^{\infty} y_r[n - rL] \quad (3-3)$$

siendo

$$y[n] = x_r[n] * h[n] \quad (3-4)$$

Por tanto la convolución es la suma de los resultados de la convolución de cada sección(ver figura 3.2). Existe el solapamiento, por querer obtener la convolución lineal en los bloques que se procesan pues la longitud L de $x_r[n]$ es mucho menor que de la convolución lineal que debería ser N = L + P - 1.

3.2.2 IMPLEMENTACIÓN DEL MÉTODO SOLAPAMIENTO-SUMA.

Esta sección describe el procedimiento para desarrollar el método solapamiento-suma. Para empezar se necesita de un programa que realice la FFT y la IFFT, las Figuras 3.3 y 3.4 muestran parte del código ensamblador de las macros *ditfft.asm* y *ifft.asm* que se encuentran en el anexo 3.2.2 y 3.2.4 respectivamente, es necesario una macro que realice la multiplicación de los espectros (ver anexo 3.2.8, m*ulcmp.asm*) de la secuencia de entrada y

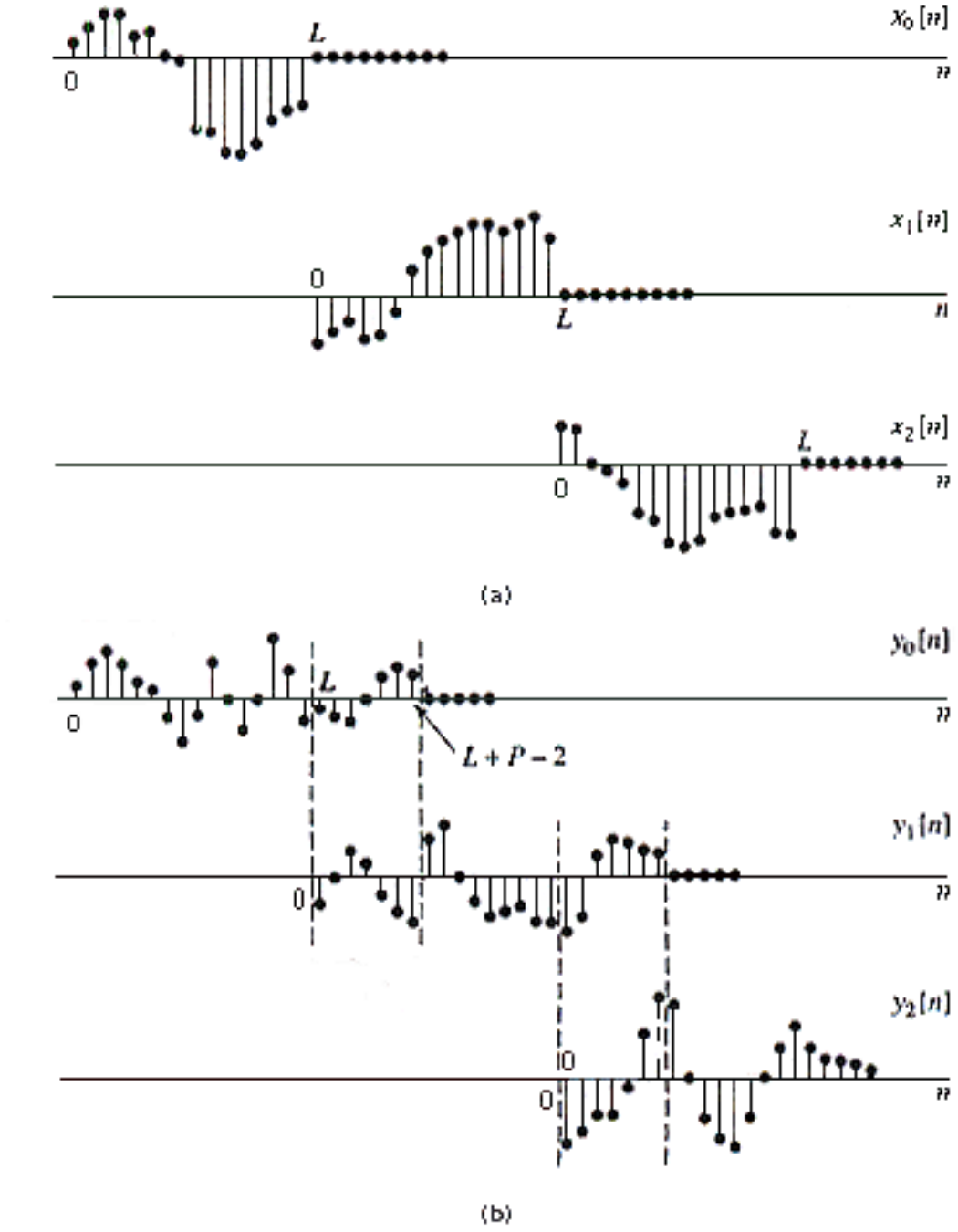

Figura 3.2 (a) Descomposición de la señal *x[n]* de la Figura 3.1 en secciones que no se solapan de longitud L

(b) Resultado de convolucionar cada sección con *h[n]*.

los coeficientes del filtro, es necesario aclarar que el programa que realiza la FFT regresa dos conjuntos de datos, un conjunto representa la parte real y el otro representa la parte imaginaria, por tanto la macro *mulcmp.asm* realiza una multiplicación compleja y en forma rectangular (x+jy = (a+jb)*(c+jd)).

```
DO     R3,ENDBFL\NMAC          ;Inicio de Butterfly
PUSH   X0
MAC    Y0,X0,B                         ;B=ar+wr*br
MOVE   X:(R1+N),X0             ;X0=bi
MACR   -Y1,X0,B                       ;B=xr
MOVE   X:(R0),A                        ;A=ar
ASL    A                       ;A=2ar
MOVE   B,X:(R0)+N
SUB    B,A                     ;A=yr
MOVE   X:(R1),B                        ;B=ai
MOVE   A,X:(R0)+N
MACR   Y0,X0,B                         ;B=ai+wr*bi
MOVE   X:(R1),A                        ;A=ai
POP    X0                      ;X0=br
MACR   Y1,X0,B                         ;B=xi
MOVE   X:(R0+N),X0
ASL    A                       ;A=2ai
MOVE   B,X:(R1)+N
SUB    B,A                     ;A=yi
MOVE   X:(R0),B
MOVE   A,X:(R1)+N
ENDBFL\NMAC
LEA    (R0)+                   ;Actualiza apuntador de reales
LEA    (R1)+                   ;Actualiza apuntador de imaginarios
```

Figura 3.3 Código que realiza la FFT directa *ditfft.asm* anexo 3.2.2.

Además, otra macro se encarga del relleno con ceros (ver anexo 3.2.14 *rellcr.asm*), luego un programa identifica y suma los puntos que se deben solapar (ver anexo 3.2.15 *sumblq.asm)* y para concluir en la Figura 3.5 se muestra el código de programa que realiza la recepción de la señal de entrada y la Figura 3.6 presenta el código para la transmisión del resultado de la convolución por bloques – en este caso del método solapamiento-suma, ecuación (3-3) –.

```
        DO     R3,BFLEND\NMAC          ;Inicio de Butterfly
        PUSH   X0
        MAC    Y0,X0,B                         ;B=ar+wr*br
        MOVE   X:(R1+N),X0          ;X0=bi
        MACR   Y1,X0,B                         ;B=xr  Y1=conjugado -wi
        MOVE   X:(R0),A                        ;A=ar
        ASL    A                    ;A=2ar
        MOVE   B,X:(R0)+N
        SUB    B,A                  ;A=yr
        MOVE   X:(R1),B                        ;B=ai
        MOVE   A,X:(R0)+N
        MACR   Y0,X0,B                         ;B=ai+wr*bi
        MOVE   X:(R1),A                        ;A=ai
        POP    X0                   ;X0=br
        MACR   -Y1,X0,B                        ;B=xi   Y1=conjugado -wi
        MOVE   X:(R0+N),X0
        ASL    A                    ;A=2ai
        MOVE   B,X:(R1)+N
        SUB    B,A                  ;A=yi
        MOVE   X:(R0),B
        MOVE   A,X:(R1)+N
BFLEND\NMAC
        LEA    (R0)+                ;Actualiza apuntador de reales
        LEA    (R1)+                ;Actualiza apuntador de imaginarios
```

Figura 3.4 Código que realiza la FFT directa *ifft.asm* anexo 3.2.4.

Pues todo estos programas, se implementan como *macros* dentro de la DSP56L811EVM y solo se hacen llamadas a dichas macros cuando sea necesario, un ejemplo de macro se muestra en la Figura 3.7 en esta figura se muestra como se crea una macro Figura 3.7(a) y también como se hace la llamada de la macro dentro del programa principal Figura 3.7(b). Todas estas macros y otros programas se encuentran en el anexo 3.

Además se necesita un programa principal el cual inicializa todos los registros a utilizar dentro de la arquitectura de la DSP56L811EVM, como lo son los registros de la ALU, AGU y el BCR también los registros que manejan la interfaz serie sincrona (SSI, por sus siglas en inglés) y como si esto fuera poco también hay que inicializar el PLL pues es la parte que genera el reloj del dispositivo en cuestión, la Figura 3.8 muestra parte del código para inicializar los registros.

```
;Servicio de interrupción de lectura sin excepción
LECISR:
      MOVEP   X:SSRTSR,Y1
      MOVEP   X:INPUT,X0
      MOVE    X0,X:(R0)+
      BFCLR   #$A000,X:SCR2
      BFSET   #$5000,X:SCR2
      RTI
;Servicio de interrupción de lectura con excepción
LECEISR
      MOVEP   X:SSRTSR,Y1
      MOVEP   X:INPUT,X0
      MOVE    X0,X:(R0)+
      BFCLR   #$A000,X:SCR2
      BFSET   #$5000,X:SCR2
      RTI
```

Figura 3.5 Parte del código que realiza la lectura de muestras.

```
;Servicio de interrupción de escritura con excepción
ESCEISR
      MOVE    X:INPUT,Y0
      MOVE    X:(R1)+,X0
      MOVEP   X:SSRTSR,Y1
      MOVEP   X0,X:OUTPUT
      BFCLR   #$5000,X:SCR2
      BFSET   #$A000,X:SCR2
      RTI
;Servicio de interrupción de escritura sin excepción
ESCISR:
      MOVE    X:INPUT,Y0
      MOVE    X:(R1)+,X0
      MOVEP   X0,X:OUTPUT
      BFCLR   #$5000,X:SCR2
      BFSET   #$A000,X:SCR2
      RTI
```

Figura 3.6 Parte del código que realiza la transmisión de datos.

```
PUSH    MACRO   REG                     ;coloca el valor REG
PUSH    IDENT   1,1                     ;en la posición de memoria
                LEA     (SP)+           ;apuntada por el puntero SP
                MOVE    REG,X:(SP)
                ENDM
```

(a)

```
    ...
PUSH R0
PUSH Y0
     …
```

(b)

Figura 3.7 (a) Definición o creación de una macro. (b) Llamada a la macro dentro del programa principal.

Luego se procederá con la lectura de las muestras, a continuación se mueven las muestras a la sección de procesamiento la cual será *in-place*, lo que significa, que ese mismo bloque de memoria se utilizara para realizar cualquier otro procesamiento. Por ultimo se moverá el resultado de la convolución por bloques al buffer de transmisión, para así tener desarrollada la convolución utilizando el método solapamiento-suma.

```
ORG    P:START
       MOVE    #$2000,SP              ;el apuntador de stack esta en la direccion $2000
       MOVE    #0000,X:BCR            ;cero estado de espera en el bus de control.
       MOVEP   #$0600,X:PCR1          ;configurando el reloj físico
       MOVEP   #$0020,X:PCR0          ;configurando PLL
      NOP                             ;retardo
      NOP
      NOP
       BFSET   #$4000,X:PCR1          ;habilita el PLL
       BFSET   #$0F00,X:PCC           ;habilita el puerto C para la SSI
       MOVEP   #FSTX,X:SCRTX          ;configurando el registro de transmisión
       MOVEP   #$5180,X:SCR2          ;configurando el registro de control de la SSI
;      MOVE    #MEDPON-1+$8000,M01
       MOVE    #ENTRA,R0
       MOVE    #DATOSR,R1
       BFSET   #$0010,X:SCR2          ;habilita transmisión
       MOVEP   #$0200,X:IPR           ;configurando registro de prioridad de interrupción
       BFCLR   #$0200,SR              ;habilita interrupciones
```

Figura 3.8 Inicialización de registro.

Para despejar cualquier duda de lo anterior en las Figuras 3.9 y 3.10 se muestra el diagrama general de la convolución por bloques y el método solapamiento-suma.

En las Figura 3.11 se presenta parte del código del programa *ovladd98.asm* el cual desarrolla la técnica de solapamiento-suma, solo se muestra las llamadas a las macros respectivas.

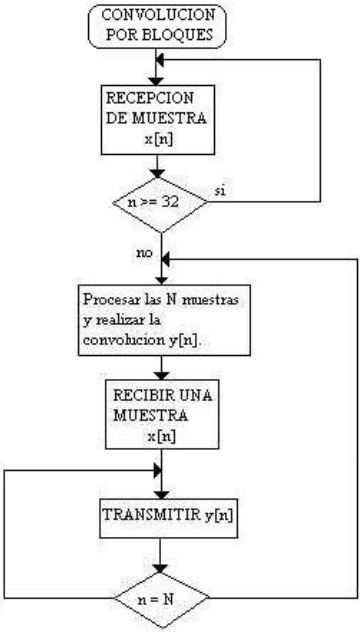

Figura 3.9 Diagrama general de la convolución por bloques.

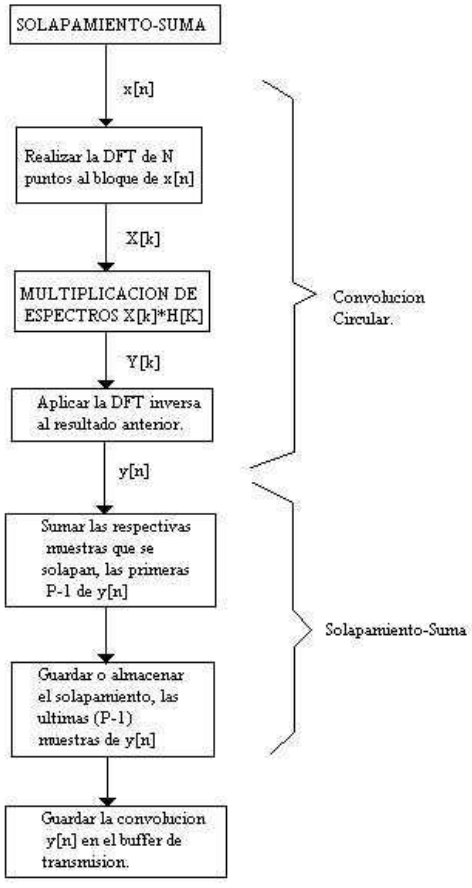

Figura 3.10 Diagrama del método solapamiento-suma.

```
RELLCR  1,POINTS,DATOSR,DATOSI,CNTRLL            ;rellena con ceros la entrada de datos
MULLP   1,POINTS,DATOSR,DATOSI,CNTLP,ESCENT      ;escala la entrada 1/10
DITFFT  1,POINTS,DATOSR,DATOSI          ; Macro que realiza la FFT directa
MULCMP  1,POINTS,DATOSR,FILTR           ;multiplicación de los espectros
IFFT    1,POINTS,DATOSR,DATOSI          ; Macro que realiza la IFFT
MULLP   2,POINTS,DATOSR,DATOSI,CNTLP,NUMERO      ;divide entre N
DIVI    1,POINTS,DATOSR,DATOSI,CNTLZ,CNTR1,CNTR2,ESCENT
                                                          ;escala la salida
SUMBLK  1,POINTS,NCOEF,DATOSR,SLPADD,CNTSUM
                              ;suma bloques de convolución con bloque almacenado
MOVBLK  1,POINTS,NCOEF,DATOSR,SLPADD,CNTBLK
                              ;guarda bloque de solapamiento
```

Figura 3.11 Secuencia de código para realizar la convolución por bloque utilizando el método solapamiento-suma, el programa es *ovladd98.asm.*

3.3 MÉTODO SOLAPAMIENTO-ALMACENAMIENTO.

Este método consiste en realizar una convolución circular de L puntos de los coeficientes del filtro *h[n]* de P puntos con un segmento de L puntos $x_r[n]$, luego se debe identificar la parte de la convolución circular que corresponde a la convolución lineal.

¿Cómo se forma la salida? Pues los segmentos resultantes de salida se pegan juntos para formar la salida. Concretamente si una secuencia de L puntos se convoluciona circularmente con una secuencia de P puntos (P<L) los primeros (P-1) puntos resultantes son incorrectos – no concuerdan con la convolución lineal – y los restantes puntos son idénticos a los que se obtiene con la convolución lineal.

La ecuación (3.3) define las secciones, cuando se divide *x[n]* en secciones de longitud L de forma que cada sección de entrada se solape con la sección precedente (P-1) puntos.

$$x_r[n] = x[n + r(L - P + 1) - P + 1] \quad 0 \le n \le L - 1 \ \ (3-5)$$

En la Figura 3.12 (a) se muestra el método solapamiento-almacenamiento. La convolución circular de cada sección con *h[n]* se denomina $y_{rp}[n]$ donde el subíndice extra

p indica que $y_{rp}[n]$ es el resultado de una convolución circular con solapamiento temporal. La Figura 3.12 (b) muestra estas secuencias. La parte de cada sección de salida en la región $0 \leq n \leq P-2$ es la parte que hay que descartar. Las restantes muestras de las secciones sucesivas se van colocando una detrás de otra para formar la salida filtrada. La ecuación (3.6) es la unión de las secciones correctas resultante de la convolución circular.

$$y[n] = \sum_{r=0}^{\infty} y_r[n - r(L - P + 1) + P - 1] \quad (3-6)$$

siendo

$$y_r[n] = \begin{cases} y_{rp}, & P-1 \leq n \leq L-1 \\ 0, & en\,el\,resto \end{cases} \quad (3-7)$$

Lo importante de este método es que usamos directamente la convolución circular y no existe un procedimiento intermedio – como la suma de bloques – para obtener la salida filtrada.

3.3.1 IMPLEMENTACIÓN DEL MÉTODO SOLAPAMIENTO-ALMACENAMIENTO.

Esencialmente, el proceso de implementación es el mismo que en el caso de solapamiento-suma, pues una buena parte del mencionado proceso se utiliza. Para comenzar se utilizan tanto la FFT y la IFFT las cuales son las macros *ditfft.asm* y *ifft.asm*, también se debe realizar la multiplicación de los espectros de la secuencia de entrada y los coeficientes del filtro por tanto se ocupa la macro *mulcmp.asm*, el relleno con ceros es necesario así que se utiliza también la macro *rellcr.asm*, también la identificación de los puntos malos y buenos es necesaria, esto se hace con la macro *movbl2.asm* .

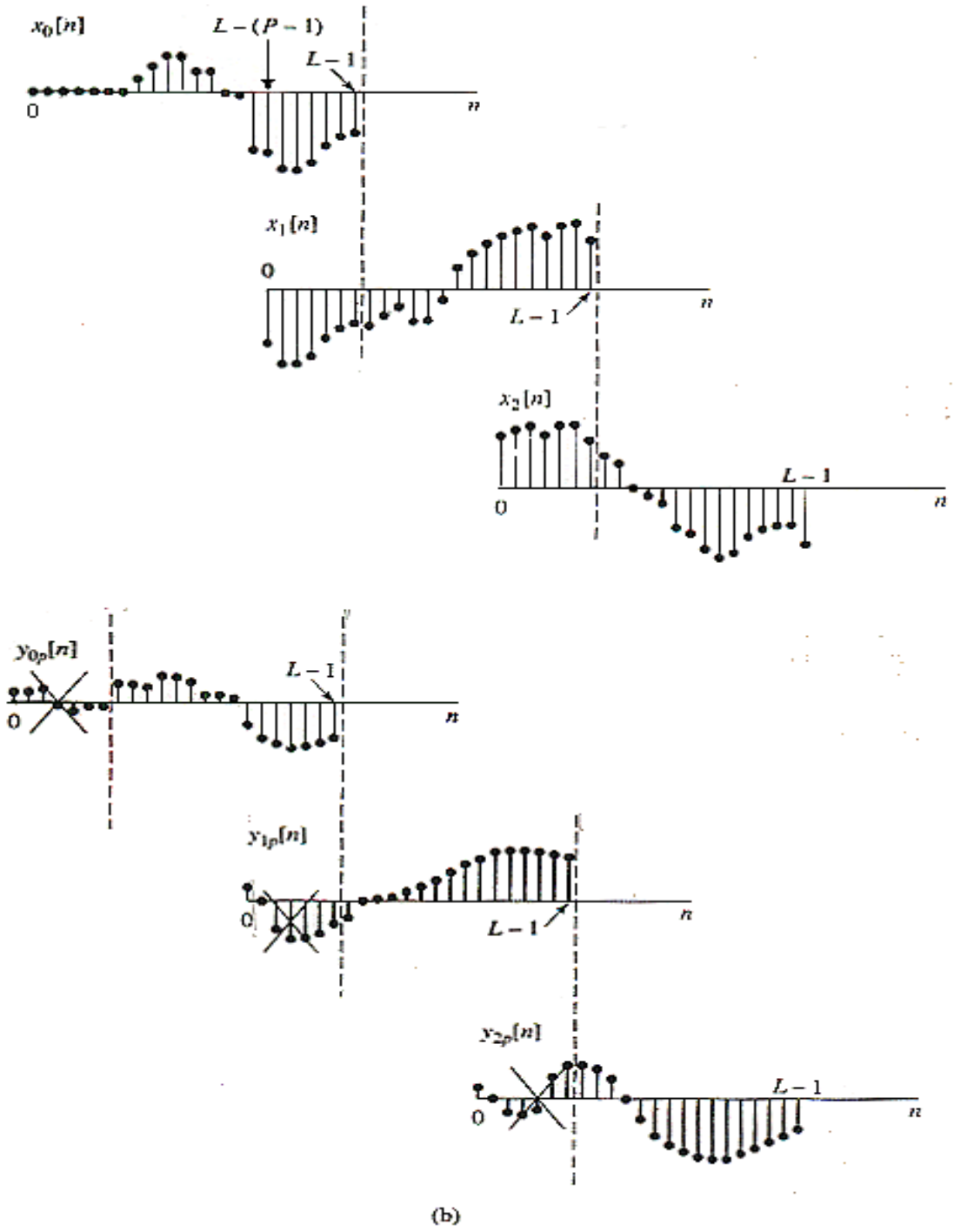

Figura 3.12 (a) Descomposición de la señal *x[n]* de la Figura 3.1 en secciones que se solapan de longitud L (b) Resultado de convolucionar cada sección con *h[n]*. Se indican las porciones de cada sección que se descartan para formar la convolución lineal.

A diferencia del anterior método los puntos identificados como buenos pasan directamente al *buffer* de salida. También, estos programas, se desarrollaron como *macros*. El programa principal es el mismo que se utiliza para el método solapamiento-suma.

Después, de la lectura de las muestras, se deben mover a la sección de memoria donde serán procesadas, ya cuando se obtiene las muestras de salida, se almacena en el *buffer* de transmisión esto se realiza con las macros *sumblq.asm* y *movblq.asm*. Dentro del proceso, existe un almacenamiento de los últimos P-1 muestras de *x[n]*. Recordando las secciones de entrada deben tener un solapamiento. Para tener una mejor claridad de lo anterior se presenta en la Figura 3.13 el diagrama de bloques del método solapamiento-almacenamiento y en la Figura 3.14 se muestra parte del código *ovlsa98.asm* que desarrolla el anterior método descrito mas detalles se encuentran en el anexo 3.

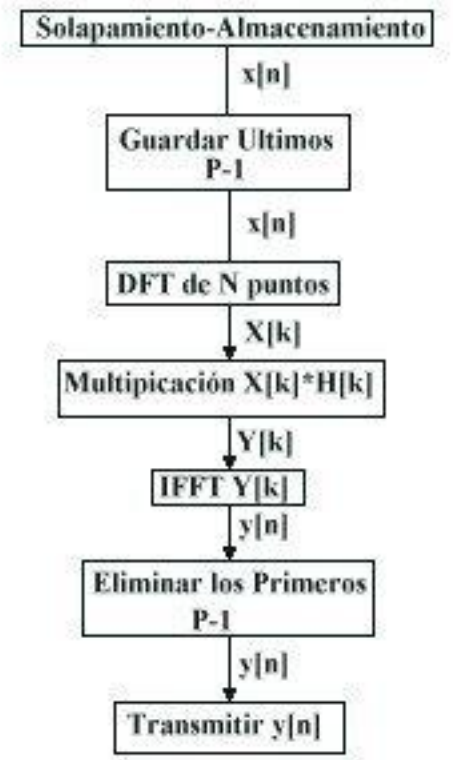

Figura 3.13 Diagrama del método solapamiento-almacenamiento.

```
                    ;rellena con ceros la entrada de datos
REL1CR  1,POINTS,DATOSR,DATOSI,CNTRLL
                    ;guarda bloque de solapamiento
MOVBLQ  1,POINTS,NCOEF,ENTRA,DATOSR,CNTBLK
                    ;suma bloques de convolución con bloque almacenado
SUMBLQ  1,POINTS,NCOEF,DATOSR,SLPADD,CNTSUM
                    ;guarda bloque de solapamiento
MOVBL2  1,POINTS,NCOEF,DATOSR,SLPADD,CNTBLK
                    ;escala la entrada 1/10
MULLP   1,POINTS,DATOSR,DATOSI,CNTLP,ESCENT
                    ;Macro que realiza la FFT directa
DITFFT  1,POINTS,DATOSR,DATOSI
                    ;multiplicación de los espectros
MULCMP  1,POINTS,DATOSR,FILTR
                    ;Macro que realiza la IFFT
IFFT    1,POINTS,DATOSR,DATOSI
                    ;divide entre N
MULLP   2,POINTS,DATOSR,DATOSI,CNTLP,NUMERO
                    ;escala la salida
DIVI    1,POINTS,DATOSR,DATOSI,CNTLZ,CNTR1,CNTR2,ESCENT
```

Figura 3.14 Código que realiza el método solapamiento-almacenamiento, utilizando el programa *ovlsa98.asm.*

3.4 CONCLUSIONES.

- Ambos métodos resuelven los problemas de almacenamiento y procesamiento de una secuencia de longitud infinita. Aunque el método de solapamiento-almacenamiento presenta la forma más eficaz de realizar la convolución por bloques.
- La convolución utilizando la FFT es la parte fundamental para poder llevar a cabo dichos métodos, pues el procesamiento de la convolución en el dominio de la frecuencia facilita el procesamiento para poder llevar a cabo el filtrado de una señal.
- Aunque puede no resultar aparente la utilidad de los métodos de convolución por bloques, aun así tenemos o contamos con una herramienta o criterio con el cual se puede decidir que método ocupar a la hora de filtrar una señal de longitud muy grande.

- Es de hacer resaltar que tanto la DFT como el diseño de filtro digitales – descritos en este trabajo de graduación – son parte fundamental para cada uno de los métodos de convolución por bloque, aunque en el caso de la DFT son aplicaciones.

REFERENCIAS BIBLIOGRÁFICAS.

[1] J. W. Cooley and J.W. Tukey, "An algorithm for the machine calculation of complex Fourier Series," *Mathematics Computation,* Vol. 19, pp. 297-301, Abril 1965

[2] A. V. Oppenheim and R.. W. Schafer, "Discrete Time Signal Processing," *Prentice Hall,* 1989.

[3] Burrus, C. Sidney et al., "Ejercicios de Tratamiento Digital de la Señal utilizando MATLAB V.4," *Prentice Hall,* Madrid, 1998.

[4] Motorola, "DSP56L811 Evaluation Module User's Manual," *Motorola,* Inc., 1997.

[5] Blanco G. E. y Calderón E. O., "Diseño de Filtros FIR utilizando la tarjeta DSP56L811EVM," *Proyecto de Ingeniería UES*, 2001.

[6] Burton W. Jones., "Teoría de los Números," *F. Trillas S.A.,* México, 1969.

[7] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader y P. D. Welch, "What is the Fast Fourier Transform?," *IEEE Trans Audio Electroacoust. (G-AE Subcommittee on Measurement Concepts),* vol. AU-15(2), pp. 45-55, Junio 1967.

[8] J. H. McClellan y T. W. Parks, "A Unified Approach to the Design of Optimum FIR Linear-Phase Digital Filters", *IEEE Trans. Circuit Theory, (Design of Digital Filters),* vol. CT-20 (6), pp. 697-701, Noviembre 1973.

[9] J. H. McClellan, T. W. Parks y L. R. Rabiner, "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters", *IEEE Trans. Audio Electroacoust. (Linear Phase Digital Filters),* vol. AU-21 (6), pp. 506-526, Diciembre 1973.

[10] J. H. McClellan y T. W. Parks, "Chebyshev approximation for nonrecursive digital filters with linear phase," *IEEE Trasn. Circuit Theory,* vol. CT-19, pp. 189-194, Marzo de 1972.

ANEXO 1.

- **PROGRAMAS PARA SIMULACIÓN DE LA DFT.**
- **ANALISIS DE LA FFT DE COOLEY-TUKEY.**

A1.1 PROGRAMAS PARA CALCULAR LA TRANSFORMADA DISCRETA DE FOURIER POR MÉTODOS DIRECTOS.

A1.1.1 MÉTODO DIRECTO UTILIZANDO DOS BUCLES ANIDADOS.

Nombre de archivo: *dft_forwd.m*

```
function [Xk]=dft_forwd(xn,N)
%Cálculo de la DFT a partir de su Definición
%Utilizando dos bucles
%xn: secuencia de duración finita de N puntos
%N: Longitud de la DFT
%Xk: coeficientes de la DFT
for k1 = 1:N%bucle externo índice k
    X(k1) = 0;
  k = k1-1;
  for n1 = 1:N;%bucle interno índice n
    n = n1-1;
    X(k1) = X(k1) + xn(n1)*exp(-j*2*pi*k*n/N);
    %ec. (1-1) de [3] (Capítulo 9)
    end
end
```

A.1.2 MÉTODO DIRECTO UTILIZANDO UNA MATRIZ

Nombre de archivo: *Dft.m*

```
function [Xk] = dft(xn,N)
% Cálculo de la DFT a partir de su definición
```

```
% Utilizando la forma matricial
% [Xk] = dft(xn,N)
% Xk = coeficientes de la DFT 0 <= k <= N-1
% xn = secuencia de duración finita de N puntos
%  N = Longitud de la DFT
n = [0:1:N-1]; % vector fila para n
k = [0:1:N-1]; % vector fila para k
WN = exp(-j*2*pi/N); % Factor Wn
nk = n'*k; % crea una matriz de NxN de nk valores
WNnk = WN .^ nk; % matriz DFT
Xk = xn * WNnk;  % vector de los coeficientes de la DFT
```

A1.1.3 CÁLCULO DEL TIEMPO DE EJECUCIÓN ENTRE LOS MÉTODOS DIRECTOS Y LA FFT.

Nombre de archivo: *dft_fft_t.m*

```
Nmax=10;
%vector de tiempos para DFT de dos bucles
dft_time=zeros(1,Nmax);
mdft_time=zeros(1,Nmax);%vector de tiempos para DFT matricial
fft_time=zeros(1,Nmax);%vector de tiempos para FFT
for n = 1:1:Nmax
   l=2^n;
   %hasta longitud de 1024 puntos
     %disp(l)
```

```
    for i=1:2 %promedio
     x = ones(1,l); %secuencia de entrada
     %tiempo de ejecución para DFT de dos bucles
     td0=clock,dft_forwd(x,l),t1=etime(clock,td0);
     %tiempo de ejecución para DFT matricial
     tmd0=clock,dft(x,l),t2=etime(clock,tmd0);
     %tiempo de ejecución para FFT
     tf0=clock,fft(x,l),t3=etime(clock,tf0);
  end
  dft_time(n)=t1/2;
  mdft_time(n)=t2/2;
  fft_time(n)=t3/2;
end
%salvando vectores de tiempo de ejecución en un archivo
save dfttime.mat dft_time mdft_time fft_time;
```

A1.1.4 CÁLCULO DE OPERACIONES EN PUNTO FLOTANTE ENTRE LOS MÉTODOS DIRECTOS Y LA FFT.

Nombre de archivo: *dft_fft_op.m*

```
%Medición de número de operaciones de la DFT
%Cálculo de la DFT ec.(1-1) de [3] (Capítulo 9)
Nmax = 10;
%vector de operaciones de la DFT con dos bucles
dft_op=zeros(1,Nmax);
```

```
%vector de operaciones de la DFT matricial
mdft_op=zeros(1,Nmax);
%vector de operaciones de la FFT
fft_op=zeros(1,Nmax);
for n = 1:1:Nmax;
  l=2^n;
  %Longitud de la secuencia de entrada(potencia de dos)
    fd = 0; fm = 0; ff = 0;
  for I=1:2%promedio de valores
    x = [ones(1,l)]; %secuencia de entrada
    %operaciones en punto flotante para DFT dos bucles
     f0=flops,dft_forwd(x,l),f1=flops-f0;
    fd=fd+f1;
    %operaciones en punto flotante para DFT matricial
    fx=flops,Dft(x,l),f2=flops-fx;
    fm=fm+f2;
    %operarciones en punto flotante para FFT
    fy=flops,fft(x,l),f3=flops-fy;
    ff=ff+f3;
  end
  dft_op(n)=fd/2;
  mdft_op(n)=fm/2;
  fft_op(n)=ff/2;
end
```

```
%salvando vectores en un archivo
save dft_operac.mat dft_op mdft_op fft_op
```

A1.1.5 MÉTODO DIRECTO UTILIZANDO EL ALGORITMO DE GOERTZEL, USANDO UN POLINOMIO EN UN SOLO BUCLE.

Nombre de archivo: *polyb.m*

```
function Xk=polyb(xn,N)
%xn: secuencia de duración finita
%N: N puntos de frecuencia
%Xk: DFT de xn de longitud N
% xn: secuencia de entrada de duración finita
% N: Longitud de la DFT
xn=fliplr(xn);%secuencia de entrada invertida
for k1 = 1:N
    X(k1) = 0;
    k = k1-1;%indice k
    z=exp(-j*2*pi*k/N);%factor WN visto como un polinomio
  X(k1) = X(k1) + polyval(xn,z);%DFT utilizando polyval
end
Xk=X;
```

A1.1.6 MÉTODO DIRECTO UTILIZANDO EL ALGORITMO DE GOERTZEL, USANDO UN POLINOMIO EN UNA MATRIZ.

Nombre de archivo: *polym.m*

```
function Xk=polym(xn,N)
%xn: secuencia de duración finita
%N: N puntos de frecuencia
%Xk: DFT de xn de longitud N
% xn: secuencia de entrada de duración finita
% N: Longitud de la DFT
        xn=fliplr(xn);%secuencia de entrada invertida
        k=0:N-1;%indice k
    z=exp(-j*2*pi*k/N);%polinomio del factor WN
    Xk=polyval(xn,z);
          %coeficientes de la DFT utilizando polyval
```

A1.1.7 MÉTODO DIRECTO UTILIZANDO EL ALGORITMO DE GOERTZEL, USANDO LA FUNCIÓN FILTER PARA DESARROLLAR LA DFT A PARTIR DE LA SALIDA DE UN FILTRO.

Nombre de archivo: *fildft.m*

```
function y=fildft(xn,N)
%xn: secuencia de duración finita
%N: N puntos de frecuencia
%Xk: DFT de xn de longitud N
  for k1 = 1:N
```

```
     X(k1) = 0;
    k = k1-1;%indice externo k
    for n1 = 1:N;
      n = n1-1;%indice interno n
      z=exp(-j*2*pi*k*n/N);
      %el factor WN se ve como un polinomio
      X(k1) = X(k1) + filter(1,z,xn(n1));
      %DFT utilizando filter
     end
  end
y=X;
%yabs=abs(X);
%k=0:Nmax-1;
%stem(k,yabs)
```

A1.1.8 CÁLCULO DEL TIEMPO DE EJECUCIÓN ENTRE LOS MÉTODOS DIRECTOS UTILIZANDO EL ALGORITMO DE GOERTZEL Y LA FUNCIÓN FFT(X,N).

Nombre de archivo: *gorz_t.m*

```
%Tiempos de ejecución de la DFT utilizando el Algoritmo
% de Goertzel(usando la función polyval y filter) con la FFT%
Nmax=10;
polybt=zeros(1,Nmax);
%vector de tiempos de ejecución de la DFT usando la función polyval
```

```
polymt=zeros(1,Nmax);
%vector de tiempo de ejecución de la DFT utilizando la función polyval
filtt=zeros(1,Nmax);
%vector de tiempo de ejecución de la DFT usando filter
fftt=zeros(1,Nmax);
%vector de tiempo de ejecución de la DFT usando la FFT
for n = 1:1:Nmax
        l=2^n;
%longitud de la secuencia de entrada (potencia de dos)
  for i=1:2%promedio
    x = ones(1,l);%secuencia de duración finita
   t0=clock,polyb(x,l),t1=etime(clock,t0);
        %tiempo para polyval
   tfl0=clock,polym(x,l),t2=etime(clock,tfl0);
        %tiempo para polyval
   tf0=clock,fildft(x,l),t3=etime(clock,tf0);
        %tiempo para filter
   tff=clock,fft(x,l),t4=etime(clock,tff);
        %tiempo para FFT
  end
  polybt(n)=t1/2;
  polymt(n)=t2/2;
  filtt(n)=t3/2;
  fftt(n)=t4/2;
```

```
end
disp('fin de gorz_t');
save gortzt.mat polybt polymt filtt fftt n;
%Salvando tiempos en un archivo
```

A1.1.9 CÁLCULO DE OPERACIONES EN PUNTO FLOTANTE ENTRE LOS MÉTODOS DIRECTOS UTILIZANDO EL ALGORITMO DE GOERTZEL Y LA FUNCIÓN FFT(X,N).

Nombre de archivo: *gorz_op.m*

```
%Medición de número de operaciones de la DFT
%Cálculo de la DFT ec.(2-1) de [3] (Capítulo 9)
Nmax = 10;
polybop=zeros(1,Nmax);
%vector de tiempos de ejecución de la DFT usando la función polyval
polymop=zeros(1,Nmax);
%vector de tiempos de ejecución de la DFT usando la función polyval
filtop=zeros(1,Nmax);
%vector de tiempo de ejecución de la DFT utilizando la función filter
fftop=zeros(1,Nmax);
%vector de tiempo de ejecución de la DFT usando la FFT
for n = 1:1:Nmax;
    l=2^n;
    %Longitud de la secuencia de entrada(potencia de dos)
    fd = 0; fm = 0;fl = 0;  ff = 0;
```

```
 for I=1:2%promedio de valores
                x = [ones(1,l)];%secuencia de entrada
  f0=flops,polyb(x,l),f1=flops-f0;
  fd=fd+f1;
      %operaciones para DFT polyval dos bucles
  fx=flops,polym(x,l),f2=flops-fx;
  fm=fm+f2;
      %operaciones para DFT polyval matricial
    fy=flops,fildft(x,l),f3=flops-fy;
  fl=fl+f3;
      %operarciones en punto flotante para DFT con filter
  fz=flops,fildft(x,l),f4=flops-fz;
  ff=ff+f4;%operarciones en punto flotante para FFT
 end
 polybop(n)=fd/2;
 polymop(n)=fm/2;
 filtop(n)=fl/2;
 fftop(n)=ff/2;
end
save gortzop.mat polybop polymop filtop fftop
%salvando vectores en un archivo
```

A1.2 PROGRAMAS PARA CALCULAR LA TRANSFORMADA DISCRETA DE FOURIER POR MÉTODOS DE LA TRANSFORMADA RAPIDA DE FOURIER.

A1.2.1 BIT-REVERSED

Nombre de archivo: *imppar.m*

```
%función que realiza la descomposición de la secuencia
%de entrada de manera recursiva
%realiza el bit-reversed de una secuencia
%x: es la secuencia de entrada
function xk=imppar(x)
N=length(x);
     if (length(x) ~= 1)
     xp=x(2:2:N);
          xi=x(1:2:N-1);
    xk=[imppar(xi) imppar(xp)];
  else
    xk=x;
  end
```

A1.2.2 FFT RADIX-2, DIEZMADO EN EL TIEMPO

Nombre de archivo: *dftdit2.m*

```
%Función que implementa en forma recursiva, el algoritmo de diezmado en el tiempo de la
DFT.
%x: vector de longitud N=2^M, inicialmente contiene al vector de entrada (tiempo n) y
finalmente a la DFT (frecuencia k)
```

```
function x=dftdit2(x,M)
N=2^M;
%Genera X[k] impares y pares
x=imppar(x);
%x
%suma de DFT de N/2
for L=1:M
  NM=2^L;
  NM1=NM/2;
  W=exp(-sqrt(-1)*pi/NM1);
%   W=(cos(pi/NM1)-(sqrt(-1)*sin(pi/NM1)))
    WN=1;
 for j=1:NM1
   for i=j:NM:N
      imp=i+NM1;
      T=x(imp)*WN;
      x(imp)=x(i)-T;%ecuación (1-7) de [3](Capitulo 9)
      x(i)=x(i)+T;%ecuación (1-6) de [3](Capitulo 9)
   end
   WN=WN*W;
 end
end
```

A1.2.3 FFT RADIX-2, DIEZMADO EN LA FRECUENCIA

Nombre de archivo: *dftdif2.m*

```
%función que desarrolla en forma recursiva, el algoritmo de %diezmado en la frecuencia de
la DFT.
%x: vector de longitud N=2^M, inicialmente contiene al vector de entrada (tiempo n) y
finalmente la DFT (frecuencia k)
function x=dftdif2(x,M)
N=2^M;
%diezmado en la frecuencia
for L=1:M
  LE=2^(M+1-L);
  LE1=LE/2;
  WN=1;
  W=(cos(pi/LE1)-(sqrt(-1)*sin(pi/LE1)));
  for j=1:LE1
    for i=j:LE:N
      ip=i+LE1;
      T=x(i)+x(ip);
            x(ip)=(x(i)-x(ip))*WN;%ecuación (1-9)de [3](Capitulo 9)
      x(i)=T;%ecuación (1-8) de [3] (Capitulo 9)
    end
    WN=WN*W;
  end
end
```

```
%bit-reversed
x=imppar(x);
```

A1.2.4 CÁLCULO DEL TIEMPO DE EJECUCIÓN ENTRE LOS MÉTODOS DIT Y DIF UTILIZANDO EL ALGORITMO COOLEY-TUKEY.

Nombre de archivo: *dit_tf.m*

```
%   Tiempo de ejecución de la FFT Cooley-Tukey
%   Diezmado en el tiempo, radix-2
Nmax=10;
fft_f_r2=zeros(1,Nmax);fft_t_r2=zeros(1,Nmax);
%Vectores que contendrán los tiempos de ejecución
for n = 1:1:Nmax
     l=2^n;%longitud de la secuencia de entrada
   for i=1:10
     x = ones(1,l);%secuencia de entrada
     tf0=clock,dftdif2(x,n),t1=etime(clock,tf0);
           %medición de los tiempos de ejecución
     tt0=clock,dftdit2(x,n),t2=etime(clock,tt0);%
   end
   fft_f_r2(n)=t1/10;%resultados promediados
   fft_t_r2(n)=t2/10;
end
disp('Listo!!!');
pause(1);
```

```
%save filt_t.txt dft_t_filt dft_t_fft n -ascii -tabs;
save timeditf.mat fft_f_r2 fft_t_r2;
%guardando los tiempos en un archivo
```

A1.2.5 CÁLCULO DE OPERACIONES EN PUNTO FLOTANTE ENTRE LOS MÉTODOS DIT Y DIF UTILIZANDO EL ALGORITMO COOLEY-TUKEY.

Nombre de archivo: *ditr2_op.m*

```
% Medición de número de operaciones de la DFT
%  Operaciones en punto flotante de la FFT Cooley-Tukey
%   Diezmado en frecuencia, radix-2
Nmax = 10;
dit_op=zeros(1,Nmax);dif_op=zeros(1,Nmax);
%Vectores que contienen el número de operaciones
for n = 1:1:Nmax;
  l=2^n;%Longitud de la secuencia de entrada
  fd = 0; fm = 0;%contadores iniciales
  for I=1:10
    x = [ones(1,l)];%secuencia de entrada
       f0=flops,dftdit2(x,m),f1=flops-f0; %medición de número de
       fd=fd+f1;                                  %operaciones
    fx=flops,dftdif2(x,m),f2=flops-fx;
    fm=fm+f2;
  end
  dit_op(n)=fd/10;%resultados promediados
```

```
  dif_op(n)=fm/10;
end
save opeditf.mat dit_op dif_op;
%guardando resultados en un archivo
```

A1.2.6 GENERACIÓN DE COEFICIENTES "TWIDDLE FACTOR"

Nombre de archivo: *coef**dft.m***

```
%genera coeficientes de la FFT DIT en archivo *.txt
function Wy=coefdft(M)
%M: es número de etapas
N=2^M;
%Wx=zeros(1,M);
Wy=zeros(1,N-1);
%generando coeficientes para la FFT DIT
for L=1:M
   NM=2^L;
   NM1=NM/2;
   W=exp(-sqrt(-1)*pi/NM1)
      WN=1;
      for j=1:NM1
     for i=j:NM:N
       Wy(i+(j-1))=WN;%actualizando vector de coeficientes
      end                          %en el orden correcto
      WN=WN*W;
```

```
    end
end
archdc(Wy,'twidd.txt');
%Salvando coeficientes en un archivo de texto
```

A1.2.7 GENERACIÓN DE ARCHIVOS DE TEXTO QUE SERÁN DESCARGADOS COMO DATOS EN LA DSP56L811.

Nombre de archivo: *archdc**.m***

```
%Convirtiendo los valores de la variable "y" %en datos que pueda leer la tarjeta
EVM56L811
function archdc(y,arch)
%function archdc(y,arch)
%y: secuencia de datos
%arch: nombre del archivo que contiene a “y”
L=length(y); s11='DC       '; fid=fopen(arch,'w');
for k=1:L-1
    ry=real(y(k)); iy=imag(y(k));
        if (abs(ry)<0.01) %parte real
      ry=0.0;
      s11=strcat(s11,num2str(ry,3),',');
       else
        s11=strcat(s11,num2str(ry,3),',');
    end
        if (abs(iy)<0.01) %parte imaginaria
```

```
        iy=0.0;
        s11=strcat(s11,num2str(iy,3),',');
      else
          s11=strcat(s11,num2str(iy,3),',');
    end
end
      ry=real(y(L)); iy=imag(y(L));
      if (abs(ry)<0.01) %parte real
        ry=0.0; s11=strcat(s11,num2str(ry,3),',');
        else
          s11=strcat(s11,num2str(ry,3),',');
      end
       if (abs(iy)<0.01) %parte imaginaria
        iy=0.0; s11=strcat(s11,num2str(iy,3));
          else
          s11=strcat(s11,num2str(iy,3));
       end
fprintf(fid,'\t%s\r\n',s11);
fclose(fid);
```

A1.3 TRANSFORMADA RÁPIDA DE FOURIER (FFT).

La importancia de la DFT estriba en que es posible utilizar un algoritmo, llamado FFT, que la realiza de forma eficiente y rápida.

La DFT de una secuencia *x[n]* es:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad k = 0,1,2 \cdots N-1 \quad (A1-1)$$

$$W_N = e^{-j2\pi/N} \quad (A1-2)$$

Una primera aproximación al cálculo de la DFT requeriría la suma compleja de N multiplicaciones complejas para cada uno de las salidas. En total, N^2 multiplicaciones complejas y N^2 sumas complejas para realizar una DFT de N puntos.

Lo que consigue el algoritmo FFT es simplificar enormemente el cálculo de la DFT introduciendo "atajos" matemáticos para reducir drásticamente el número de operaciones (ver Figuras:1.2, 1.4 y 1.6).

La optimización del proceso de cálculo de la DFT está basado en las siguientes ideas:

- Simetría y periodicidad de los términos W_N.

$$W_N^{n+N} = W_N^n \quad (A1-3) \qquad W_N^{n+N/2} = -W_N^n \quad (A1-5)$$

$$W_N^{Nk} = 1 \quad (A1-4) \qquad W_N^2 = W_{N/2} \quad (A1-6)$$

- Elegimos el valor de N de forma que $N = R^M$. Al factor R se le denomina radix y su valor más habitual es *2*, de forma que $N = 2^M$ y el algoritmo se le denomina FFT *radix-2*.

A1.3.1 FFT RADIX-2 DIEZMADO EN EL TIEMPO (DIT).

Para este tipo de algoritmo dividimos la secuencia de datos de entrada *x[n]* en dos grupos, uno de índices par y el otro de índices impar. Con estas subsecuencias se realiza la DFT de *N/2* puntos y sus resultados se combinan para formar la DFT de *N* puntos.

$$X[k]=\sum_{n=0}^{N/2-1}x[2n]W_N^{2nk}+\sum_{n=0}^{N/2-1}x[2n+1]W_N^{(2n+1)k}=\sum_{n=0}^{N/2-1}x[2n]W_N^{2nk}+W_N^k\sum_{n=0}^{N/2-1}x[2n+1]W_N^{2nk}$$

Sustituimos

$$x_1[n]=x[2n]\quad x_2[n]=x[2n+1]$$

$$W_N^{2nk}=W_{N/2}^{nk}$$

$$X[k]=\sum_{n=0}^{N/2-1}x_1[n]W_{N/2}^{nk}+W_N^k\sum_{n=0}^{N/2-1}x_2[n]W_{N/2}^{nk}=Y[k]+W_N^kZ[k]\quad k=0,1,2\cdots N-1\quad (A1-7)$$

La ecuación (A1-7) muestra que la DFT de *N* puntos es la suma de dos DFT's de *N/2* puntos (*Y[k],Z[k])* realizadas con las secuencias par e impar de la secuencia original

$$X[k]=Y[k]+W_N^kZ[k]$$

$$X[k+N/2]=Y[k]-W_N^kZ[k]\quad k=0,1,2\cdots N/2-1\quad (A1-8)$$

x[n]. Cada termino *Z[k]* es multiplicado por un factor W_N^k , llamado "*twiddle factor"*. Debido a la ecuación (A1-5) y a la periodicidad de *Y[k]* y *Z[k]* (periodo *N/2*) podemos poner *X[k]* como:

Las dos DFT de *N/2* puntos (Figura A1.1) se puede dividir a su vez dividir para formar 4 DFT's de *N/4* puntos, lo que produce las siguientes ecuaciones:

$$Y[k]=U[k]+W_N^{2k}V[k]\quad (A1-9)$$

$$Y[k+N/4]=U[k]-W_N^{2k}V[k]\quad (A1-10)$$

$$para\ k=0,1,2\cdots N/4-1$$

$$Z[k]=R[k]+W_N^{2k}S[k]\quad (A1-11)$$

$$Z[k+N/4]=R[k]-W_N^{2k}S[k]\quad (A1-12)$$

$$para\ k=0,1,2\cdots N/4-1$$

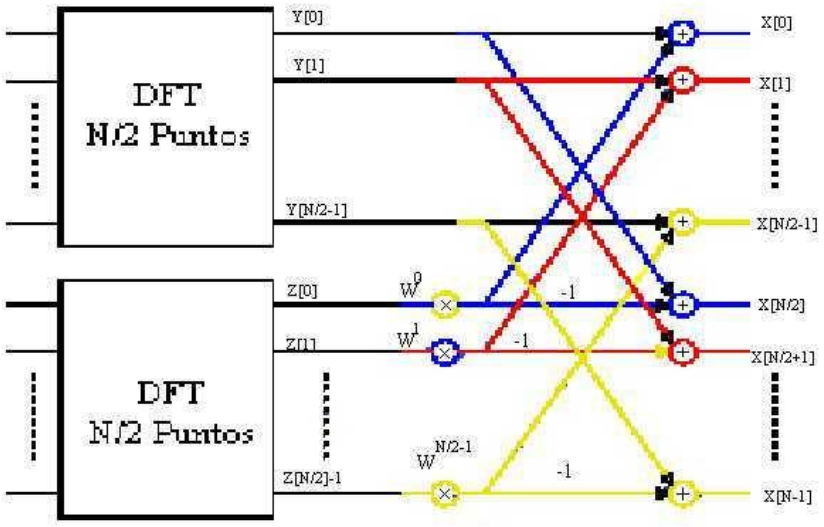

Figura A1.1 DFT de N puntos en dos DFT's de N/2 puntos, utilizando diezmado en el tiempo..

El proceso puede repetirse hasta llegar a calcular la DFT de dos valores de *x[n]*, en concreto *x[k] y x[k+N/2]*, para k = 0,1,2... N/2-1.Como ejemplo la Figura A1.2 muestra un esquema para calcular una DFT de *N=8* puntos.

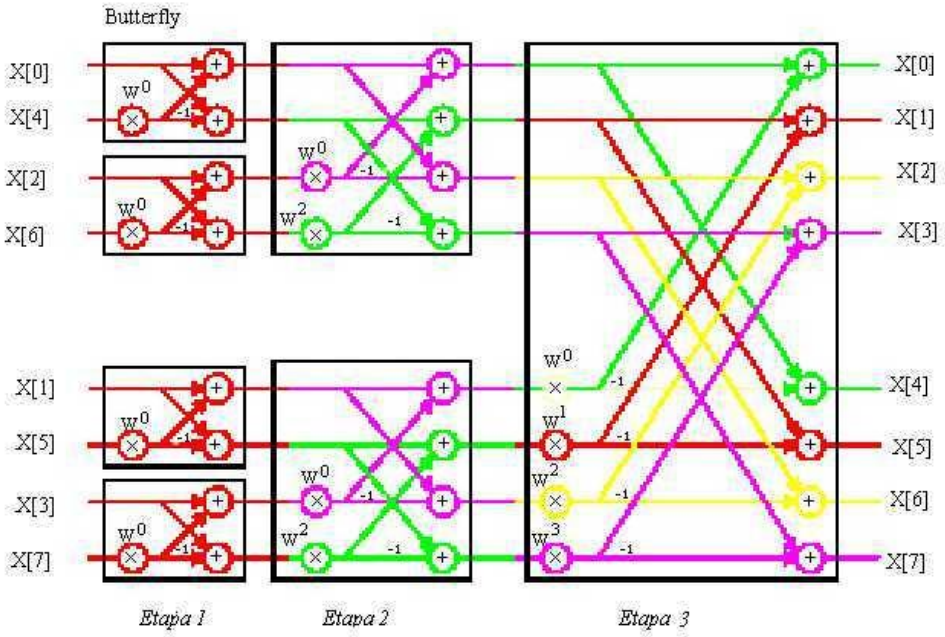

Figura A1.2 DFT de N=8, utilizando diezmado en el tiempo.

Las características de una FFT de *N* puntos diezmados en el tiempo se resumen en la Tabla A1.1.

Tabla A1.1

	Etapa 1	Etapa 2	Etapa3	***Etapa log_2N***
Número de Grupos	*N/2*	*N/4*	*N/8*	*1*
Butterflies por Grupo	*1*	*2*	*4*	*N/2*
Exponentes ***Twiddle Factors***	*(N/2)k, k=0*	*(N/4)k, k=0,1*	*(N/8)k, k=0,1,2,3*	*k, k=0,1...N/2-1*

Resumiendo, por cada *butterfly* tenemos una multiplicación y dos sumas complejas, hay *N/2 butterfly* por etapa y log_2N etapas. Por tanto, el número total de multiplicaciones es $N/2*log_2N$ y el número total de sumas es $N*log_2N$. Para valores pequeños de *N*, la diferencia puede parecer pequeña, pero para valores grandes la diferencia es enorme, esto se puede observar en las Figuras 1.2 y 1.4, donde el número de operaciones es menor utilizando la FFT que la DFT.

A1.3.2 FFT RADIX-2 DIEZMADO EN LA FRECUENCIA.

Ahora expresemos la FFT como suma de las FFT de dos secuencias, la primera con los *N/2* primeros datos y la segunda con los *N/2* últimos.

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{nk}=\sum_{n=0}^{N/2-1}x[n]W_N^{nk}+\sum_{n=N/2}^{N-1}x[n]W_N^{nk}$$
$$=\sum_{n=0}^{N/2-1}x[n]W_N^{nk}+\sum_{n=0}^{N/2-1}x[n+N/2]W_N^{(n+N/2)k}$$
$$=\sum_{n=0}^{N/2-1}x[n]W_N^{nk}+(-1)^k\sum_{n=0}^{N/2-1}x[n+N/2]W_N^{nk}$$
$$=\sum_{n=0}^{N/2-1}[x[n]+(-1)^k x[n+N/2]]W_N^{nk}\quad k=0,1,2\cdots N-1\quad (A1-13)$$

El diezmado en frecuencia se obtiene dividiendo la secuencia de salida (*X[k])* en dos ecuaciones, una para los índices pares y otro para los impares.

$$X[2k]=\sum_{n=0}^{N/2-1}[x[n]+x[n+N/2]]W_N^{2nk}$$
$$=\sum_{n=0}^{N/2-1}[x[n]+x[n+N/2]]W_{N/2}^{nk}\quad k=0,1,2\cdots N/2-1\quad (A1-14)$$
$$X[2k+1]=\sum_{n=0}^{N/2-1}[x[n]-x[n+N/2]]W_N^{n(2k+1)}$$
$$=\sum_{n=0}^{N/2-1}[[x[n]-x[n+N/2]]W_N^{n}]W_{N/2}^{nk}\quad k=0,1,2\cdots N/2-1\quad (A1-15)$$

Notemos que *X[2k]* y *X[2k+1]* son los resultados de la DFT de *N/2* puntos realizado con las sumas y la diferencia entre la primera y segunda mitades de la secuencia de entrada.

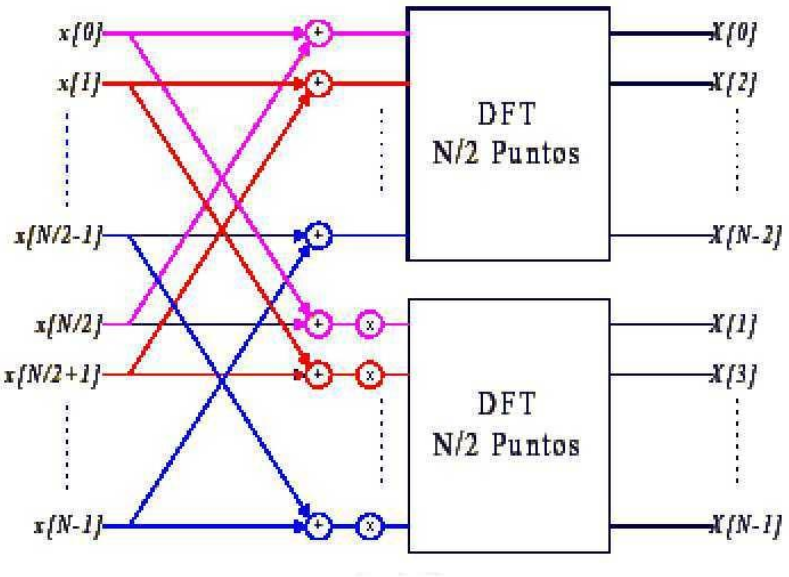

Figura A1.3 DFT de N/2 puntos utilizando, diezmado en la frecuencia.

Para el mismo caso de $N=8$ se muestra en la Figura A1.4 un esquema de diezmado en frecuencia.

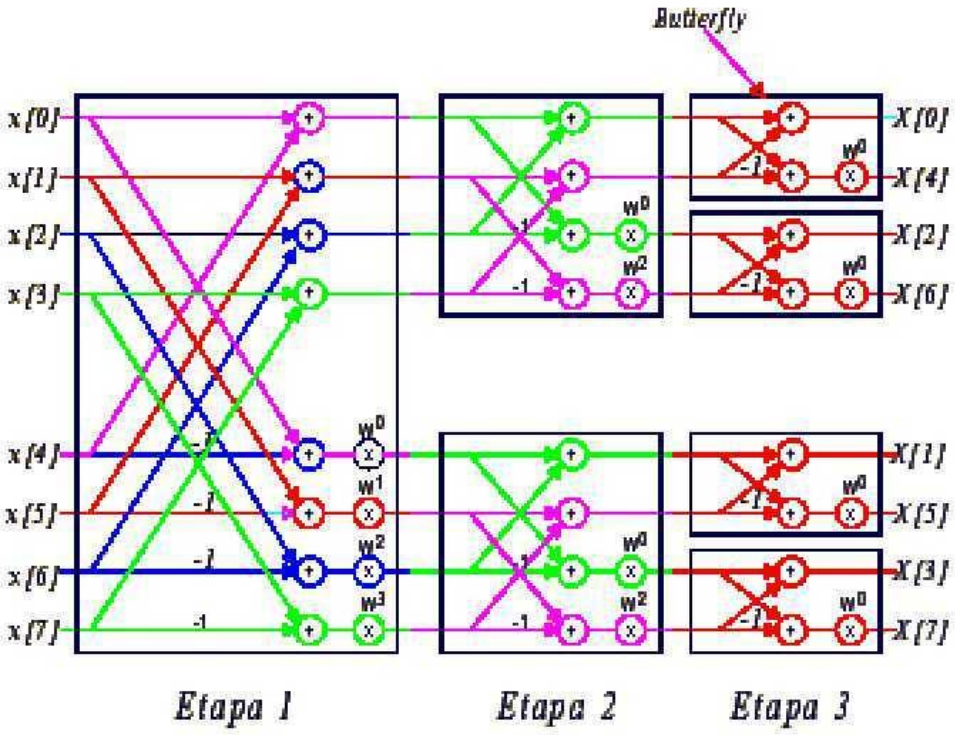

Figura A1.4 DFT de N=8 puntos, utilizando diezmado en la frecuencia

La *Tabla A1*.2 resume las características de la FFT diezmado en frecuencia.

Tabla A1.2

	Etapa 1	Etapa 2	Etapa3	***Etapa log_2N***
Número de Grupos	*1*	*2*	*4*	*N/2*
Butterflies por Grupo	*N/2*	*N/4*	*N/8*	*1*
Exponentes ***Twiddle Factors***	*n, n=0,1...N/2-1*	*2n, n=0,1..N/4-1*	*4n, n=0,1...N/8-1*	*(N/2)n n=0*

Se puede observar que en el caso de diezmado en el tiempo, la secuencia de entrada debe ser ordenada, mientras que la salida aparece en el orden correcto. Pero, en el caso de diezmado en la frecuencia, la secuencia esta en orden, mientras que la salida habrá que reordenarla. Se da la situación que esa reordenación es simplemente invertir el índice en binario. Por ejemplo en la misma posición que *x[1]* aparece *x[4]*, y 001 invertido es 100. A tal proceso, se conoce como *bit-reversed.*

ANEXO 2.

- **CUATRO TIPOS DE FILTROS DE FASE LINEAL FIR**
- **PROGRAMAS PARA SIMULACIÓN DE FILTROS DIGITALES, USANDO EL ALGORITMO DE REMEZ.**

A2.1 CUATRO TIPOS DE FILTROS DE FASE LINEAL FIR.

Un filtro digital FIR de tamaño N con respuesta al impulso {h(k)}, k=0,1,…,N-1, posee una respuesta en frecuencia la cual es la transformada *z* evaluada en el círculo unitario ($z=e^{j\omega}$): $H(e^{j\omega}) = H(z)|_{z=e^{j\omega}} = \sum_{k=0}^{N-1} h(k)e^{-j\omega k}$. $(A2-1)$

A partir de un filtro de fase lineal generalizada denotamos que la respuesta en frecuencia puede ser escrita como: $H(e^{j\omega})=G(e^{j\omega})e^{j(A+B\omega)}$ (A2-2)

Donde *A* y *B* son constantes y $G(e^{j\omega})$ es una función real (posiblemente bipolar) de ω. Observe que $G(e^{j\omega})$ no es la magnitud de la respuesta en frecuencia ya que puede ser negativa, pero $|G(e^{j\omega})|$ es la magnitud.

Primero se muestra que $A = 0$ o $\pi/2$. Podemos limitar *A* para tender entre 0 y π porque un signo menos puede ser incorporado dentro de $G(e^{j\omega})$. Puesto que $|G(e^{j\omega})|$ es la respuesta en magnitud, $|G(e^{j\omega})|$ es una función par. Así $G(e^{j\omega})$ es ya sea una función par o impar. Recordemos que $H^*(e^{j\omega})=H(e^{-j\omega})$. Si $G(e^{j\omega})$ es par, entonces:

$$G(e^{-j\omega})\, e^{j(A-B\omega)}=H(e^{-j\omega})=H^*(e^{j\omega})=G(e^{j\omega})\, e^{-j(A+B\omega)}. \quad \text{(A2-3)}$$

Esto implica que $e^{jA} = e^{-jA}$, y así $A = 0$. Si $G(e^{j\omega})$ es impar, entonces $-e^{jA} = e^{-jA}$ y $A = \pi/2$. Note que cuando $A \neq 0$ tenemos un atraso establecido, pero no, fase lineal en el sentido usual.

Ahora se demuestra que $B = -(N-1)/2$ y que hay dos tipos de simetría para la respuesta al impulso(simetría conjugada y antisimetría conjugada), dependiendo si $G(e^{j\omega})$ es par o impar. Si $G(e^{j\omega})$ es par, entonces definiendo:

$$H_0(z) = z^{-B}H(z), H_0(e^{j\omega}) = H_0(z) \quad (A2-4)$$

$H_0(e^{j\omega})$ es netamente real porque $H_0(e^{j\omega}) = G(e^{j\omega})$. Así $H_0(e^{j\omega})=H_0^*(e^{j\omega})=H_0(e^{-j\omega})$ y

escribiendo esto:
$$\sum_{k=0}^{N-1} h(k) z^{-B-k} = \sum_{k=0}^{N-1} h(k) z^{B+k}$$

$$= z^{2B+N-1} \sum_{k=0}^{N-1} h(N-1-k) z^{-B-k} \quad (A2-5) \text{ para } z = e^{j\omega}$$

Así $B=-(N-1)/2$ por lo que igualando los coeficientes se produce $h(0) = h(N-1)$ … $h(k) = h(N-1-k)$, para $k=0,1, \ldots, N-1$. Esta simetría de la respuesta al impulso será referida como *simetría conjugada*.

Si $G(e^{j\omega})$ es impar, $A = \pi/2$ y con $H_0(z) = z^{-B} H(z)$, $H_0(e^{j\omega})$ es netamente imaginario puesto que $H_0(e^{j\omega}) = jG(e^{j\omega})$. Así $H_0(e^{j\omega}) = -H_0^*(e^{j\omega}) = -H_0(e^{-j\omega})$, lo cual es escrito como

$$\sum_{k=0}^{N-1} h(k) z^{-B-k} = -\sum_{k=0}^{N-1} h(k) z^{B+k} \quad (A2-6) \text{ para } z = e^{j\omega}$$

De nuevo $B = -(N-1)/2$, pero ahora la simetría es diferente, ya que: $h(0) = -h(N-1)$, $h(1) = -h(N-2)$, … , etc. Esta simetría será llamada *antisimetría conjugada*.

Ahora examinemos la forma de la función $G(e^{j\omega})$. Diferentes funciones resultan dependiendo de si N es par o impar y si la simetría requerida para fase lineal es simétrica o antisimétrica.

A2.1.1 Simetría Conjugada – Longitud impar.

$$G(e^{j\omega}) = \sum_{k=0}^{n} a(k) \cos(\omega k) \quad (A2-7)$$

Donde $n = (N-1)/2$, $a(0) = h(n)$, y $a(k) = 2h(n-k)$, para $k = 1,2, \ldots, n$. Este es el caso el cual fue considerado en un articulo anteriormente mencionado [7].

A2.1.2 Simetría Conjugada – Longitud par.

$$G(e^{j\omega}) = \sum_{k=1}^{n} b(k)\cos(\omega(k-\frac{1}{2})) \quad (A2-8)$$

Donde $n = N/2$ y $b(k) = 2h(n-k)$, para $k = 1,\ldots,n$. Rabiner y Herrman aplicaron programación lineal en este caso [9].

A2.1.3 Antisimetría Conjugada – Longitud impar.

$$G(e^{j\omega}) = \sum_{k=1}^{n} c(k)sin(\omega k) \quad (A2-9)$$

Donde $n = (N-1)/2$ y $c(k) = 2h(n-k)$, para $k = 1,\ldots,n$. En virtud de que $G(e^{j\omega})$ sea impar, debemos tener $h(n)=0$. Herrman [10] usó este tipo en el diseño de filtros de transformada Hilbert.

A2.1.4 Antisimetría Conjugada – Longitud par.

$$G(e^{j\omega}) = \sum_{k=1}^{n} d(k)sin(\omega(k-\frac{1}{2})) \quad (A2-10)$$

Donde $n = N/2$ y $d(k) = 2h(n-k)$, para $k = 1,\ldots,n$. Anteriormente, diferenciadores han sido diseñados para este caso usando programación lineal [8].

Una estrategia única para la aproximación del problema resulta cuando $G(e^{j\omega})$. Es rescrita usando (A2-11)-(A2-13) en la forma $G(e^{j\omega})=Q(e^{j\omega})P(e^{j\omega})$, donde $P(e^{j\omega})$ es una combinación lineal de funciones cósenos:

$$\sum_{k=1}^{n} b(k)\cos(\omega(k-\frac{1}{2})) = \cos(\tfrac{\omega}{2})\sum_{k=0}^{n-1} \tilde{b}(k)\cos(\omega k) \quad (A2-11)$$

$$\sum_{k=1}^{n} c(k)sin(\omega k) = sin(\omega)\sum_{k=0}^{n-1} \tilde{c}(k)\cos(\omega k) \quad (A2-12)$$

$$\sum_{k=1}^{n} d(k)sin(\omega(k-\frac{1}{2})) = sin(\tfrac{\omega}{2})\sum_{k=0}^{n-1} \tilde{d}(k)\cos(\omega k) \quad (A2-13)$$

Note que $G(e^{j\omega})$ es forzada a ser cero en ya sea ω=0 y/o ω=0.5 por (A2-11)-(A2-13).

A2.1.5 Comprobación de (A2-11).

Recordar la identidad trigonométrica: *cos A cos B = (1/2) [cos(A+B) + cos(A-B)]*.

Sustituyendo esto dentro del lado derecho de (A2-11), obtenemos

$$\sum_{k=0}^{n-1} \tilde{b}(k)\cos(\tfrac{\omega}{2})\cos(\omega k) = \frac{1}{2}\sum_{k=0}^{n-1} \tilde{b}(k)\left[\cos(\omega(k+\frac{1}{2})) + \cos(\omega(k-\frac{1}{2}))\right]$$

$$= \frac{1}{2}\sum_{k=1}^{n} \tilde{b}(k-1)\cos(\omega(k-\frac{1}{2})) + \frac{1}{2}\sum_{k=0}^{n-1} \tilde{b}(k)\cos(\omega(k-\frac{1}{2}))$$

$$= \frac{1}{2}\tilde{b}(0)\cos(\omega(-\frac{1}{2})) + \frac{1}{2}\sum_{k=1}^{n-1}\left[\tilde{b}(k)+\tilde{b}(k-1)\right]\cos(\omega(k-\frac{1}{2})) + \frac{1}{2}\tilde{b}(n-1)\cos(\omega(n-\frac{1}{2})).$$

Así $\quad b(1) = \tilde{b}(0) + \frac{1}{2}b(1)$

$$b(k) = \frac{1}{2}(\tilde{b}(k-1) + \tilde{b}(k)), \qquad k = 2,3,...,n-1$$

$$b(n) = \frac{1}{2}\tilde{b}(n-1)$$

A2.1.6 Comprobación de (A2-12).

Usando la identidad trigonométrica: *cos A sen B = (1/2) [sen(A+B) - sen(A-B)]*

y procediendo como en la comprobación de (A2-11) para obtener:

$$c(1) = \tilde{c}(0) - \frac{1}{2}\tilde{c}(2)$$

$$c(k) = \frac{1}{2}\left[\tilde{c}(k-1) - \tilde{c}(k+1)\right], \qquad k = 2,3,...,n-2$$

$$c(n-1) = \frac{1}{2}\tilde{c}(n-2)$$

$$c(n) = \frac{1}{2}\tilde{c}(n-1)$$

La ecuación (A2-13) es derivada de la misma manera, resultando en las siguientes relaciones entre los coeficientes.

$$d(1) = \tilde{d}(0) - \frac{1}{2}\tilde{d}(1)$$

$$d(k) = \frac{1}{2}\left[\tilde{d}(k-1) - \tilde{d}(k)\right], \qquad k = 2,3,...,n-1$$

$$d(n) = \frac{1}{2}\tilde{d}(n-1)$$

A2.2 PROGRAMAS PARA SIMULACIÓN DE FILTROS DIGITALES, USANDO EL ALGORITMO DE REMEZ.

A2.2.1 Filtro Pasa Bajo, M=24 (Figura 2.3) .

Nombre de archivo: *remezpb24.m*

```
% Filtro Paso Bajo M=24
clear
clc
cla
% frecuencia de muestreo 8000hz
fp = 0.08;
```

```
fs = 0.16;
wp = fp*pi;%banda de paso
ws = fs*pi;%banda de rechazo
M = 24;%Coeficientes
N= M-1;
wt=[1 1];
f = [0 2*wp/(pi) 2*ws/(pi) 1];%Rango de frecuencias
m = [1 1 0 0];%rango de magnitud
h = remez(N,f,m,wt);%Respuesta al impulso
%se crean dos archivos de texto
%el primero contiene los coeficientes
%el segundo contiene el número de coeficientes
archDC(M,h,'remez\remezp24.txt','remez\longrmzb24.txt')
[H omega]=freqz(h,1,256);%respuesta en amplitud
Hr = abs(H);
db= 20*log10(Hr);
% Graficos
clf
figure(1);
subplot(2,1,1)
plot(omega/(2*pi),Hr);
title('Respuesta en Amplitud');
axis([0 0.5 0 1.1]);
text(0.25,-0.1,'frecuencias');
```

```
ylabel('Hr(w)')
set(gca,'XTickMode','manual','XTick',[0,wp/pi,ws/pi,0.5])
grid
subplot(2,1,2);
plot(omega/(2*pi),db);
text(0.16,20,'Respuesta en Magnitud en dB');
axis([0,0.5,-60,10]);
text(0.25,-66,'frecuencias');
ylabel('DECIBELES')
set(gca,'XTickMode','manual','XTick',[0,wp/pi,ws/pi,0.5])
grid
gtext('Filtro Paso Bajo M=24');
gtext('f[0 0.8 0.16 0.5]');
gtext('m[1 1 0 0]');
%figure(2)
%semilogy(omega/(2*pi),Hr)
%xlabel('frecuencia')
%ylabel('Magnitud log')
```

A2.2.2 Filtro Pasa Banda, M=32 (Figura 2.4) .

Nombre de archivo: *remezpbda32.m*

```
% Filtro Pasa Banda M=32
fs1 = 0.1;
fp1 = 0.2;
```

```
fp2 = 0.35;

fs2 = 0.425;

ws1 = fs1*pi;

wp1 = fp1*pi;

wp2 = fp2*pi;

ws2 = fs2*pi;

M=32;%coeficientes

N=M-1;

f = [0 2*ws1/pi 2*wp1/pi 2*wp2/pi 2*ws2/pi 1];

m = [0 0 1 1 0 0];

weights = [10 1 10];

[h,error,res] = remez(N,f,m,weights,{32});

%se crean dos archivos de texto

%el primero contiene los coeficientes

%y el segundo el número de coeficentes

archDC(M,h,'remez\remezpbda32.txt','remez\longrmzpbda32.txt')

[H omega]=freqz(h,1,256);%respuesta en amplitud

Hr = abs(H);

db = 20*log10(Hr);

% Graficos

clf

plot(omega/(2*pi),db);

title('Respuesta en Magnitud en dB');

axis([0,0.5,-90,10]);
```

```
xlabel('frecuencia');
ylabel('DECIBELES')
set(gca,'XTickMode','manual','XTick',[0,ws1/pi,wp1/pi,wp2/pi,ws2/pi,0.5])
grid
gtext('Filtro Pasa Banda M=32');
gtext('f[0 0.1 0.2 0.35 0.425 0.5]');
gtext('m[0 0 1 1 0 0]');
```

A2.2.3 Filtro Pasa Banda, M=50 (Figura 2.5) .

Nombre de archivo: *remezpbda50.m*

```
% Filtro Pasa Banda M=50
fs1 = 0.15;
fp1 = 0.2;
fp2 = 0.3;
fs2 = 0.35;
ws1 = fs1*pi;
wp1 = fp1*pi;
wp2 = fp2*pi;
ws2 = fs2*pi;
M=50%coeficientes
N=M-1;
f = [0 2*ws1/pi 2*wp1/pi 2*wp2/pi 2*ws2/pi 1];
m = [0 0 1 1 0 0];
weights = [10 1 100]
```

```
[h,error,res] = remez(N,f,m,weights);
%archivos de textos que contienen
%coeficientes y número de coeficientes
archDC(N,h,'remez\remezpbda50.txt','remez\longrmzpbda50.txt')
[H omega]=freqz(h,1,501);%respuesta en amplitud
Hr = abs(H);
db = 20*log10(Hr);%respuesta en magnitud
% Graficos
clf
plot(omega/(2*pi),db);
title('Respuesta en Magnitud en dB');
axis([0,0.5,-120,10]);
xlabel('frecuencia');
ylabel('DECIBELES')
set(gca,'XTickMode','manual','XTick',[0,ws1/pi,wp1/pi,wp2/pi,ws2/pi,0.5])
grid
gtext('Filtro Pasa Banda M=50');
gtext('f[0 0.15 0.2 0.3 0.35 0.5]');
gtext('m[0 0 1 1 0 0]');
```

A2.2.4 Filtro Rechaza Banda, M=31 (Figura 2.6) .

Nombre de archivo: *remezrbda31.m*

```
% Filtro Rechaza Banda M=31
clear
```

```
clc

cla

fp1 = 0.1;

fs1 = 0.15;

fs2 = 0.35;

fp2 = 0.42;

wp1 = fp1*pi;

ws1 = fs1*pi;

ws2 = fs2*pi;

wp2 = fp2*pi;

M=31%coeficientes

N=M-1;

f = [0 2*wp1/pi 2*ws1/pi 2*ws2/pi 2*wp2/pi 1];

m = [1 1 0 0 1 1];

weights = [1 50 1];

h = remez(N,f,m,weights,{32});

%se crean dos archivos de texto

%el primero contiene los coeficientes

%y el segundo el número de coeficentes

archDC(N,h,'remez\remezpbda31.txt','remez\longrmzpbda31.txt');

[H omega]=freqz(h,1,501);%respuesta en amplitud

Hr = abs(H);

db = 20*log10(Hr);

% Graficos
```

```
clf
plot(omega/(2*pi),db);
title('Respuesta en Magnitud en dB');
axis([0,0.5,-90,10]);
xlabel('frecuencia');
ylabel('DECIBELES')
set(gca,'XTickMode','manual','XTick',[0,wp1/pi,ws1/pi,ws2/pi,wp2/pi,0.5])
grid
gtext('Filtro Rechaza Banda M=31');
gtext('f[0 0.1 0.15 0.35 0.42 0.5]');
gtext('m[1 1 0 0 1 1]');
```

A2.2.5 Filtro Multibanda Banda, M=55 (Figura 2.7) .

Nombre de archivo: *remezmbda55.m*

```
% Filtro Multi Pasa Banda M=55
clear
clc
cla
fs1 = 0.05;
fp1 = 0.1;
fp2 = 0.15;
fs2 = 0.18;
fs3 = 0.25;
fp3 = 0.3;
```

```
fp4 = 0.36;

fs4 = 0.41;

ws1 = fs1*pi;

wp1 = fp1*pi;

wp2 = fp2*pi;

ws2 = fs2*pi;

ws3 = fs3*pi;

wp3 = fp3*pi;

wp4 = fp4*pi;

ws4 = fs4*pi;

M=55%coeficientes

N=M-1;

f = [0 2*ws1/pi 2*wp1/pi 2*wp2/pi 2*ws2/pi 2*ws3/pi 2*wp3/pi 2*wp4/pi 2*ws4/pi 1];

m = [0 0 1 1 0 0 1 1 0 0];

weights = [10 1 3 1 20];

h = remez(N,f,m,weights);

%contiene archivos de texto de los coeficientes

%y número de coeficientes

archDC(M,h,'remez\remezmbda55.txt','remez\longrmzbda55.txt')

[H omega]=freqz(h,1,256);%respuesta en amplitud

Hr = abs(H);

db = 20*log10(Hr);

% Graficos

clf
```

```
plot(omega/(2*pi),db);
title('Respuesta en Magnitud en dB');
axis([0,0.5,-90,10]);
xlabel('frecuencia');
ylabel('DECIBELES')
set(gca,'XTickMode','manual','XTick',[0,ws1/pi,wp1/pi,wp2/pi,ws2/pi,ws3/pi,wp3/pi,wp4/p
i,ws4/pi,0.5])
grid
gtext('Filtro Multi-Banda M=55');
gtext('f[0 0.05 0.1 0.15 0.18 0.25 0.3 0.36 0.41 0.5]');
gtext('m[0 0 1 1 0 0]');
```

A2.2.6 Filtro Diferenciador, M=32 (Figura 2.8) .

Nombre de archivo: *remezdif32.m*

```
% Filtro Differenciador M=32
clear
clc
cla
M = 32;%Coeficientes
N = M-1;
f = [0 1];
m = [0 1];
weights = [1];
[h,error,res] = remez(N,f,m,weights,'differentiator');
```

```
%Dos archivos de texto son generados, el
%primero contiene los coeficientes y el
%segundo el número total de coeficientes
archDC(N,h,'remez\remezdif32.txt','remez\longrmzdif32.txt')
fs=linspace(0,0.5,256);
[H,w]=freqz(h,1,256);
clf
subplot(311)
plot(fs,abs(H))
title('Diferenciador M = 32');
ylabel('Magnitud en DB');
grid
subplot(312)
plot(0:0.5/255:0.5,res.error)
ylabel('Error');
axis([0 0.5 -0.0062 0.0062])
grid
er=((res.error)./abs(res.H));
subplot(313)
plot(0:.5/255:.5,er)
ylabel('Error Relativo');
xlabel('frecuencia');
axis([0 0.5 -0.0062 0.0062])
grid
```

A2.2.7 Filtro Transformada de Hilbert , M=20 (Figura 2.9) .

Nombre de archivo: *remezhilbert20.m*

```
% Filtro Transformada de Hilbert M=20
clear
clc
cla
%T=64;
f=linspace(0.05,0.5,T)*2;
m=ones(1,T);
weights=ones(1,T/2);
M=20;%Coeficientes
N=M-1;
[h,error,res]=remez(N,f,m,weights,'hilbert');
%archivos de texto, los cuales contienen los
%coeficientes y número total de coeficientes
archDC(N,h,'remez\remezhilbert20.txt','remez\longrmzhilbert.txt')
fs=linspace(0,0.5,256);
[H,w]=freqz(h,1,256);
clf
subplot(311)
plot(0:0.5/255:0.5,abs(H))
title('Transformada de Hilbert M=20');
ylabel('Magnitud')
```

```
axis([0 0.5 0 1.0205])
grid
subplot(312)
plot(0:0.5/255:0.5,abs(H))
ylabel('Magnitud')
axis([0.05 0.5 0 1.0205]);
grid
subplot(313)
er=((res.error)./abs(res.H));
ler=size(er)
plot((0:0.5/95:0.5),er)
ylabel('Error')
xlabel('frecuencia')
axis([0.05 0.5 -0.0205 0.0205]);
grid
```

A2.2.8 Filtro Pasa Banda, M=128 (Figura 2.10) .

Nombre de archivo: *remezpbda128.m*

```
% Filtro Pasa Banda M=128
clear
clc
cla
M=128;%Coeficientes
N=M-1;
```

```
f1 = (0:0.00416666666667:0.1);

w1 = [10./(1-9*f1)];

f2 = (0.15:0.00333333333:0.25);

w2 = [10./(9*f2-1.25)];

f=[f1 0.12 0.13 f2 0.25 0.251 0.5]*2;

m=[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

weights = [w1(1:2:24) 1 w2(1:2:30) 10 10 ];

[h,error,res] = remez(N,f,m,weights);

%Genera archivo de coeficientes, para la DSP56L811EVM.

archDC(N,h,'remez\remezpbda120.txt','remez\longrmzpbda120.txt')

[H omega]=freqz(h,1,501);%respuesta en amplitud

Hr = abs(H);

db = 20*log10(Hr);

% Graficos

clf

plot(omega/(2*pi),db);

title('Respuesta en Magnitud en dB');

axis([0,0.5,-90,10]);

xlabel('frecuencia');

ylabel('DEC);

gtext('Filtro Pasa Banda M=128);
```

ANEXO 3.

PROGRAMAS PARA IMPLEMENTACIÓN DE LA CONVOLUCIÓN EN TIEMPO REAL EN LA DSP56L811EVM.

A3.1 IMPLEMENTACIÓN DE LA FFT RADIX-2 DIEZMADO EN EL TIEMPO EN LA DSP56l811EVM.

La forma de implementar la FFT en la DSP56L811EVM, se basa en la DIT y en algoritmos in-place[1]. La figura A3.1 muestra la gráfica de flujo de señal para implementar la *butterfly*, y a partir de esta forma básica poder construir la FFT DIT.

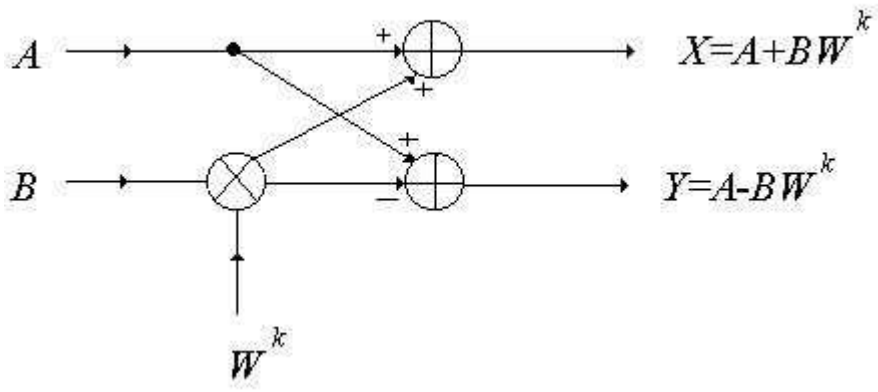

Figura A3.1 Gráfica de flujo de señal de una *butterfly*.

El hecho es que la Figura A3.1 representa un cálculo *in-place*, lo cual esta ligado a que los nodos de entrada están asociados con los nodos de almacenamiento y que los nodos de entrada de la *butterfly* son los mismos nodos de salida de dicha estructura.

Ahora, se necesita que los datos se encuentren en una región de memoria, por ejemplo, los datos A en un espacio X:$0000 y los datos B en otro X:$0100. La Figura 8.5 como es el cálculo básico utilizado para calcular la FFT, debe tener la habilidad de trabajar con números complejos, en otras palabras la *butterfly* cambia dos números complejos en

[1] Algoritmos In-place : Consiste en utilizar el mismo espacio de memoria para los datos de entrada, almacenamiento de cálculos intermedios y datos de salida, con este algoritmo se logra mayor rapidez en el procesamiento de los datos.

otros dos números complejos. Lo cual significa que A y B son complejos, por tanto el mapa de memoria podría ser el que se muestra en la Figura A3.2.

Los registros X0, Y0,Y1 y los acumuladores A y B, nos ayudaran a desempeñar de manera eficiente el cálculo de la *butterfly* (ver Figura A3.3). Otro aspecto importante son los coeficientes W^k los cuales se calculan con la ayuda de MATLAB (*coefdft.m*), estos se almacenan en memoria como se observa en la Figura A3.2 y se apunta con R2, en este caso, ya que, la FFT es DIT los coeficientes se almacenan en orden normal pero en el caso DIF los coeficientes se deberán almacenar en *bit-reversed*. A partir del *twiddle factor* $W^k = wr + j\, wi = \cos(2\pi k/N) + j \sin(2\pi k/N)$, de $A = ar + j\, ai$ y $B = br + j\, bi$, tenemos que la ecuación A3-1 muestra como se calcula la parte real de X (*xr*), la ecuación A3.2 calcula la parte imaginaria de X (*xi*). Obtenidos, *xr* y *xi,* los utilizamos para calcular *yr* y *yi*, por medio de las ecuaciones A3-3 y A3-4 respectivamente. Parte del archivo fuente fftdit2.asm, se presenta en el anexo A3.2, este programa utiliza tres lazos anidados para calcular la FFT DIT.

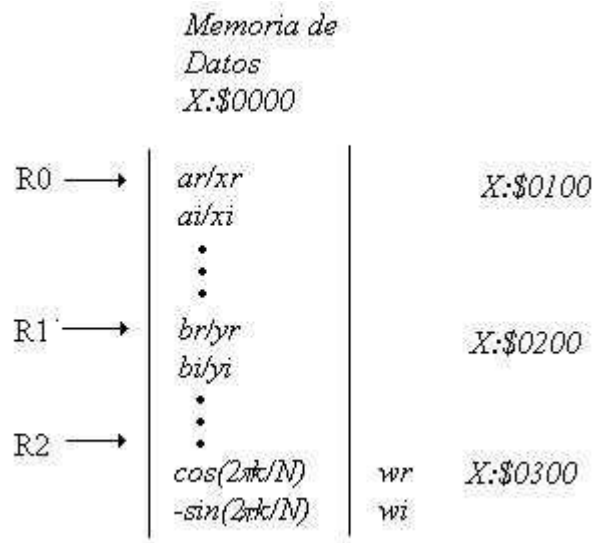

Figura A3.2 Mapa de memoria para *butterfly*.

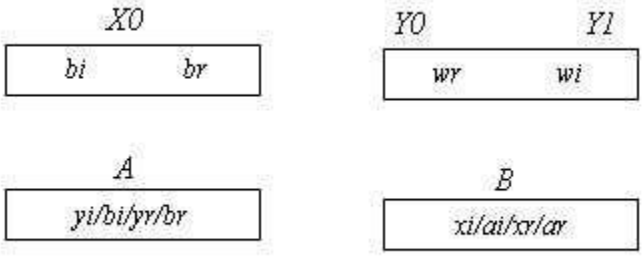

Figura A3.3 Registros X0, Y0 y Y1, acumuladores A y B utilizados para implementar *butterfly*.

$$xr = ar + wr * br - wi * bi \quad (A3-1)$$
$$xi = ai + wi * br + wr * bi \quad (A3-2)$$
$$yr = ar - wr * br + wi * bi = 2 * ar - xr \quad (A3-3)$$
$$yi = ai - wi * br - wr * bi = 2 * ai - xi \quad (A3-4)$$

A3.2 PROGRAMAS O MACROS UTILIZADAS PARA LA IMPLEMENTACION DE CONVOLUCIÓN POR BLOQUES EN LA DSP56L811EVM.

Todos las macros que a continuación se presentarán se encuentran en un subdirectorio llamado [MACRO], pues a través del código de operación MACLIB [ruta de macros o librerías], se incluyen o importan dichos programas para ser utilizados en los programas principales.

A3.2.1 MACRO QUE DESARROLLA EL BIT-REVERSED

Nombre de archivo: *bitrv**.asm (macro)***

```
;-------------------------------------------------------------
;BITRV  --- INVERSION DE BIT(bit-reversed)
;-------------------------------------------------------------

BITRV       MACRO   NMAC,PUNTOS,DATAR,DATAI
BITRV       IDENT   1,2
            MOVE    #DATAR,R0       ;apunta a inicio de datos reales
            MOVE    #DATAI,R2       ;apunta a inicio de datos imaginarios
            MOVE    R0,R1           ;R0 apunta a pares R1 a impares reales
            MOVE    #1,X0           ;inicializador de comparación
            MOVE    R2,R3           ;R2 apuntara a pares R3 a impares imaginarios
            MOVE    #PUNTOS/2,N     ;Mitad del grupo de datos
            MOVE    N,Y0            ;el registro Y0 contiene el valor de N
            MOVE    #PUNTOS-2,N     ;Offset de datos
```

```
        MOVE    N,X:INVBITS     ; X:INVBITS = offset
LAZO\NMAC
        CMP     X0,Y0           ;compara si Y0>=X0,cierto ir a VERDAD
        BGT     VERDA\NMAC      ;falso continuar
        MOVE    Y0,N            ;inicia intercambio de muestras reales
        LEA     (R1)+N          ;actualiza apuntador R1
        MOVE    X:(R1),A        ;valor apuntado por R1 se mueve A
        MOVE    X0,N            ;actualizara nuevo offset
        LEA     (R0)+N          ;actualiza R0
        MOVE    X:(R0),B        ;intercambio de valores apuntados por
        MOVE    B,X:(R1)        ;los apuntadores R0 y R1
        MOVE    A,X:(R0)        ;termina intercambio de muestras reales
        MOVE    Y0,N            ;inicia intercambio de muestras imaginarias
        LEA     (R3)+N          ;actualiza el apuntador R3 con el offset N
        MOVE    X:(R3),A        ;acumulador A obtiene valor apuntado por R3
        MOVE    X0,N            ;nuevo offset
        LEA     (R2)+N          ;actualiza R2
        MOVE    X:(R2),B        ;intercambio entre los valores
        MOVE    B,X:(R3)        ; apuntados por R2 y R3
        MOVE    A,X:(R2)        ; termina intercambio de muestras imaginarias
VERDA\NMAC
        MOVE    #PUNTOS/2,N     ;el registro Y1 contendra
        MOVE    N,Y1            ;la mitad de puntos de datos
LA2ZO\NMAC
```

```
        CMP     Y0,Y1           ;compara si Y1>Y0,cierto ir a VERDA2
        BGT     VER2DA\NMAC     ;falso seguir
        SUB     Y1,Y0           ;ajuste para encontrar las nuevas
        ASR     Y1              ;muestras que se van a intercambiar
        BRA     LA2ZO\NMAC
VER2DA\NMAC
        ADD     Y1,Y0           ;ajuste para una posible siguiente
        INC     X0              ; comparación, para el nuevo par de datos
        MOVE    #DATAR,R0       ;R0 apunta a DATAR,datos reales
        MOVE    #DATAI,R2       ;R1 apunta a DATAI,datos imaginarios
        MOVE    R0,R1           ;intercambio de apuntadores
        MOVE    R2,R3           ;R0->R1,R2->R3, -> significa asignamiento
        DECW    X:INVBITS       ; decrementa la variable anteriormente
                                ;mencionada
        JGT     LAZO\NMAC       ;sigue hasta que X:INVBITS=$0000
        ENDM
```

A3.2.2 MACRO QUE DESARROLLA LA FFT DIRECTA DE N PUNTOS

Nombre de archivo: *ditfft.asm* (macro)

```
;-------------------------------------------------------------------------------------------------
;DITFFT --- TRANSFORMADA RAPIDA DE FOURIER DIRECTA, DIEZMADO EN EL TIEMPO
;       --- RADIX-2
;-------------------------------------------------------------------------------------------------
```

```
DITFFT      MACRO   NMAC,POINTS,DATOSR,DATOSI
DITFFT      IDENT   1,2
BITRV       NMAC,POINTS,DATOSR,DATOSI  ;Macro que realiza el bit-reversed
            CLR     A
            MOVE    #POINTS*2,N         ;Iniciando Offset de WN
            MOVE    N,X:OFFSET
            MOVE    #POINTS/2,N         ;Iniciando Loop de butterfly
            MOVE    N,X:BUTTER
            MOVE    #1,X:GRUPO          ;Iniciando Loop de grupo
            MOVE    #1,X:NEXTWI         ;Offset entre real de WN e imaginario de WN
            MOVE    #@CVI(@LOG(POINTS)/@LOG(2)+0.5),X:CNTETP
            ;Inicio de ETAPA
            ETAPA\NMAC
            MOVE    X:GRUPO,N
            MOVE    #DATOSR,R0
            MOVE    #(POINTS-1)+$8000,M01 ;POINTS-1 de buffer circular
            MOVE    #DATOSI,R1
            MOVE    #COEF,R2
            NOP
            MOVE    N,X:CNTPSS              ;Inicio de PASS
            PASS\NMAC
            MOVE    X:(R2),Y0               ;Y0=Wr
            MOVE    X:(R0+N),X0             ;X0=br
```

```
MOVE    X:(R0),B                ;B=ar
PUSH    N
MOVE    X:NEXTWI,N
MOVE    X:(R2+N),Y1             ;Y1=Wi
MOVE    X:OFFSET,N
LEA     (R2)+N                  ;Apunta a siguiente función base
POP     N
MOVE    X:BUTTER,R3
DO      R3,ENDBFL\NMAC          ;Inicio de Butterfly
PUSH    X0
MAC     Y0,X0,B                 ;B=ar+wr*br
MOVE    X:(R1+N),X0             ;X0=bi
MACR    -Y1,X0,B                ;B=xr
MOVE    X:(R0),A                ;A=ar
ASL     A                       ;A=2ar
MOVE    B,X:(R0)+N
SUB     B,A                     ;A=yr
MOVE    X:(R1),B                ;B=ai
MOVE    A,X:(R0)+N
MACR    Y0,X0,B                 ;B=ai+wr*bi
MOVE    X:(R1),A                ;A=ai
POP     X0                      ;X0=br
MACR    Y1,X0,B                 ;B=xi
MOVE    X:(R0+N),X0
```

```
ASL     A                       ;A=2ai
MOVE    B,X:(R1)+N
SUB     B,A                     ;A=yi
MOVE    X:(R0),B
MOVE    A,X:(R1)+N
ENDBFL\NMAC
LEA     (R0)+                   ;Actualiza apuntador de reales
LEA     (R1)+                   ;Actualiza apuntador de imaginarios
DECW    X:CNTPSS
JGT     PASS\NMAC
;FIN DE GRUPO
MOVE    X:BUTTER,A              ;Actualiza Butterfly
MOVE    X:GRUPO,B               ;Actualiza Pass
ASR     A
ASL     B
MOVE    A,X:BUTTER
MOVE    B,X:GRUPO
MOVE    X:OFFSET,A              ;Actualiza el offset
ASR     A                       ;del siguiente coeficiente WN
MOVE    A,X:OFFSET
DECW    X:CNTETP
JGT     ETAPA\NMAC
ENDM
```

A3.2.3 MACRO QUE REALIZA LA OPERACIÓN DE DIVISIÓN.

Nombre de archivo: ***divi.asm*** (macro)

```
;---------------------------------------------------------
;DIVI   --- MACRO DIVISION DE UN BLOQUE POR UN NUMERO FRACCIONARIO
;
DIVI    MACRO   NMAC,POINTS,DATOSR,DATOSI,CNTLZ,CNTR1,CNTR2,ESCENT
DIVI    IDENT   1,2


        MOVE    X:ESCENT,X0         ;escalamiento en ESCENT=divisor=X0
        MOVE    #DATOSR,R3          ;apunta a inicio de datos reales
        MOVE    #POINTS,N           ;lazo de N puntos,total de datos
        MOVE    N,X:CNTR1
LZDV\NMAC
                                    ;división real
        MOVE    X:(R3),B            ;B=dividendo=dato real
        MOVE    B1,A                ;guarda bit de signo del dividendo(B1) en A1
        MOVE    B1,N                ;guarda el bit de signo del dividendo(B1) en N
        ABS     B                   ;obliga a división positiva
        EOR     X0,Y1               ;guarda bit de signo del cociente en el bit
                                    ;N(bandera negativa) del SR
        BFCLR   #$0001,SR           ;limpia bit C (acarreo),necesario para la
                                    ;primera instrucción de DIV
                                    ;División
        REP     #$10                ;divide 16 veces
```

```
        DIV     X0,B                ;se construye el cociente positivo en B0
        MOVE    B0,Y1               ;Y1 contiene el cociente
        BFTSTH  #$8000,N
        BCS     DONE\NMAC           ;asigna signo a Y1 correspondiente
        NOT     Y1
        INCW    Y1
DONE\NMAC
        MOVE    Y1,X:(R3)+          ;mover el cociente a la posición de memoria
                                    ;actualizar apuntador
        DECW    X:CNTR1             ; CNTR1>0
        JGT     LZDV\NMAC
                                    ;división imaginaria
        MOVE    #DATOSI,R3          ;apunta a inicio de datos imaginarios
        MOVE    #POINTS,N           ;lazo de N puntos,total de datos
        MOVE    N,X:CNTR2
LZ2DV\NMAC
        MOVE    X: (R3), B          ;B=dividendo=dato imaginario
        MOVE    B1,A                ;guarda  bit de signo del dividendo(B1) en A1
        MOVE    B1,N                ;guarda el bit de signo del dividendo(B1) en N
        ABS     B                   ;obliga a división positiva
        EOR     X0,Y1               ;guarda bit de signo del cociente en el bit
                                    ;N(bandera negativa) del SR
        BFCLR   #$0001,SR           ;limpia el bit C (acarreo),necesario para la
                                    ;primera instrucción de DIV
```

```
                                        ;División
        REP     #$10                    ;repites 16 veces
        DIV     X0,B                    ;se construye el cociente positivo en B0
        MOVE    B0,Y1
        BFTSTH  #$8000,N
        BCS     DO2NE\NMAC              ; asigna signo a Y1 correspondiente
        NOT     Y1
        INCW    Y1
DO2NE\NMAC
        MOVE    Y1,X:(R3)+      ;mover el cociente a la posición de memoria
        DECW    X:CNTR2         ;CNTR2>0
        JGT     LZ2DV\NMAC
        ENDM
```

A3.2.4 MACRO QUE DESARROLLA LA FFT INVERESA DE N PUNTOS

Nombre de archivo: ***itfft.asm*** (macro)

```
;---------------------------------------------------------
;IFFT --- TRANSFORMADA RAPIDA DE FOURIER INVERSA
;     --- RADIX-2
;
IFFT    MACRO   NMAC,PUNTOS,DATAR,DATAI
IFFT    IDENT   1,2
M\NMAC  EQU     NMAC+4
BITRV   M\NMAC,PUNTOS,DATAR,DATAI
```

```
        CLR     A
        MOVE    #PUNTOS*2,N         ;Iniciando Offset de WN
        MOVE    N,X:OFFSET
        MOVE    #PUNTOS/2,N         ;Iniciando Loop de butterfly
        MOVE    N,X:BUTTER
        MOVE    #1,X:GRUPO          ;Iniciando Loop de pass
        MOVE    #1,X:NEXTWI         ;Offset entre real de WN e imaginario de WN
        MOVE    #@CVI(@LOG(PUNTOS)/@LOG(2)+0.5),X:CNTETP
        ;Inicio de ETAPA
ET2PA\NMAC
        MOVE    X:GRUPO,N
        MOVE    #DATAR,R0
        MOVE    #(PUNTOS-1)+$8000,M01 ;POINTS-1 de buffer circular
        MOVE    #DATAI,R1
        MOVE    #COEF,R2
        NOP
        MOVE    N,X:CNTPSS              ;Inicio de PASS
PA2SS\NMAC
        MOVE    X:(R2),Y0               ;Y0=Wr
        MOVE    X:(R0+N),X0             ;X0=br
        MOVE    X:(R0),B                ;B=ar
        PUSH    N
        MOVE    X:NEXTWI,N
        MOVE    X:(R2+N),Y1             ;Y1=Wi
```

```
MOVE    X:OFFSET,N
LEA     (R2)+N                  ;Apunta al siguiente función base
POP     N
MOVE    X:BUTTER,R3
DO      R3,BFLEND\NMAC          ;Inicio de Butterfly
PUSH    X0
MAC     Y0,X0,B                 ;B=ar+wr*br
MOVE    X:(R1+N),X0             ;X0=bi
MACR    Y1,X0,B                 ;B=xr  Y1=conjugado -wi
MOVE    X:(R0),A                ;A=ar
ASL     A                       ;A=2ar
MOVE    B,X:(R0)+N
SUB     B,A                     ;A=yr
MOVE    X:(R1),B                ;B=ai
MOVE    A,X:(R0)+N
MACR    Y0,X0,B                 ;B=ai+wr*bi
MOVE    X:(R1),A                ;A=ai
POP     X0                      ;X0=br
MACR    -Y1,X0,B                ;B=xi   Y1=conjugado -wi
MOVE    X:(R0+N),X0
ASL     A                       ;A=2ai
MOVE    B,X:(R1)+N
SUB     B,A                     ;A=yi
MOVE    X:(R0),B
```

```
        MOVE    A,X:(R1)+N
BFLEND\NMAC
        LEA     (R0)+                   ;Actualiza apuntador de reales
        LEA     (R1)+                   ;Actualiza apuntador de imaginarios
        DECW    X:CNTPSS
        JGT     PA2SS\NMAC
;FIN DE GRUPO
        MOVE    X:BUTTER,A              ;Actualiza Butterfly
        MOVE    X:GRUPO,B               ;Actualiza Pass
        ASR     A
        ASL     B
        MOVE    A,X:BUTTER
        MOVE    B,X:GRUPO
        MOVE    X:OFFSET,A              ;Actualiza el offset
        ASR     A                       ;del siguiente coeficiente WN
        MOVE    A,X:OFFSET
        DECW    X:CNTETP
        JGT     ET2PA\NMAC
        ENDM
```

A3.2.5 MACRO QUE MUEVE BLOQUES DE DATOS EN MEMORIA EN EL MÉTODO SOLAPAMIENTO-ALMACENAMIENTO.

Nombre de archivo: ***movbl2.asm*** (macro)

```
;-----------------------------------------------------------------------------------------------------
;MOVBL2 --- MUEVE UN BLOQUE DE MEMORIA DE DATOS  A OTRO BLOQUE DE MEMORIA
;          EN EL METODO SOLAPAMIENTO-ALMACENAMIENTO
MOVBL2  MACRO   NMAC,POINTS,NCOEF,BLKA,BLKB,CNTBLK
MOVBL2  IDENT   1,2

        MOVE    #BLKA,R0                ;inicio de datos de entrada
        MOVE    #BLKB,R3                ;inicio de datos de solapamiento temporal=0
        MOVE    #POINTS-NCOEF+1,X:CNTBLK        ;P-1 la longitud de datos de solapamiento
       MOVE    #NCOEF-1,N
        LEA     (R0)+N
ENDBLQ\NMAC
        MOVE    X:(R0)+,Y1              ;y1=ar
        MOVE    Y1,X:(R3)+              ;a=br
        DECW    X:CNTBLK
        JGT     ENDBLQ\NMAC
        ENDM
```

A3.2.6 MACRO QUE MUEVE BLOQUES DE DATOS EN MEMORIA EN EL MÉTODO SOLAPAMIENTO-SUMA.

Nombre de archivo: ***movblk.asm*** (macro)

```
;---------------------------------------------------------
;MOVBLK --- MUEVE UN BLOQUE DE MEMORIA DE DATOS  A OTRO BLOQUE DE MEMORIA
;          EN EL METODO SOLAPAMIENTO-SUMA
MOVBLK  MACRO   NMAC,POINTS,NCOEF,BLKA,BLKB,CNTBLK
MOVBLK  IDENT   1,2

      MOVE    #BLKA,R0                ;inicio de datos de convolución
      MOVE    #BLKB,R3                ;inicio de datos de solapamiento temporal=0

      MOVE    #NCOEF-2,X:CNTBLK       ;P-1 la longitud de datos de solpamiento

      MOVE    #POINTS/2,N
      LEA     (R0)+N                  ;apunta al inicio de los P-1 solapamiento
ENDBLK\NMAC
      MOVE    X:(R0)+,Y1              ;y1=ar
      MOVE    Y1,X:(R3)+              ;a=br
      DECW    X:CNTBLK
      JGT     ENDBLK\NMAC
      ENDM
```

A3.2.7 MACRO QUE MUEVE BLOQUES DE DATOS EN MEMORIA EN EL MÉTODO SOLAPAMIENTO-ALMACENAMIENTO SOLAMENTE LA PARTE DE SOLAPAMIENTO.

Nombre de archivo: ***movblq.asm*** (macro)

```
;---------------------------------------------------------
;MOVBLQ --- MUEVE UN BLOQUE DE MEMORIA DE DATOS  A OTRO BLOQUE DE MEMORIA   PERO SOLO LA PARTE DE SOLAPAMIENTO EN EL METODO SOLAPAMIENTO-ALMACENAMIENTO.
MOVBLQ  MACRO   NMAC,POINTS,NCOEF,BLKA,BLKB,CNTBLK
MOVBLQ  IDENT   1,2
            MOVE    #BLKA,R0            ;inicio de datos de convolución circular
            MOVE    #BLKB,R3            ;inicio de datos de bloque de transmisión
                                        ;de los valores buenos o iguales a la
                                        ;convolución lineal.
            MOVE    #POINTS-NCOEF+1,X:CNTBLK ;cantidad de valores buenos.
            MOVE    #NCOEF-1,N
            LEA     (R0)+N              ;apunta al inicio de los valores buenos.
ENDBLK\NMAC
            MOVE    X:(R0)+,Y1          ;y1=valor real, valor bueno de la
                                        ;convolución circular.
            MOVE    Y1,X:(R3)+          ;a=valor real, valor a transmitir.
            DECW    X:CNTBLK
            JGT     ENDBLK\NMAC
            ENDM
```

A3.2.8 MACRO QUE DESARROLLA LA MULTIPLICACIÓN COMPLEJA.

Nombre de archivo: ***mulcmp.asm*** (macro)

```
;---------------------------------------------------------
;MULCMP --- MULTIPLICACION COMPLEJA
;
MULCMP  MACRO  NMAC,POINTS,DATOSA,DATOSB
MULCMP  IDENT   1,2
                MOVE    #DATOSA,R0
                MOVE    #DATOSB,R3
                MOVE    #DATOSA,R1
                MOVE    #POINTS,N
                MOVE    X:(R3),B                ;b=br
                MOVE    N,X:MULTI
ENDMUL\NMAC
                MOVE    X:(R0),Y1
                MOVE    X:(R3),X0               ;y1=ar y x0=br
                MPY     Y1,X0,A                 ;a=ar*br
                MOVE    X:(R0+N),Y0             ;y0=ai
                MPY     Y0,X0,B
                MOVE    X:(R3+N),X0              ;b=ai*br y x0=bi
                MACR    -Y0,X0,A                ;a=ar*br-ai*bi
                MACR    Y1,X0,B
                MOVE    A,X:(R1)                ;b=ai*br+ar*bi
                MOVE    B,X:(R1+N)
```

```
        LEA     (R0)+
        LEA     (R1)+
        LEA     (R3)+
        DECW    X:MULTI
        JGT     ENDMUL\NMAC
        ENDM
```

A3.2.9 MACRO QUE DESARROLLA LA MULTIPLICACIÓN DE UN NUMERO FRACCIONARIO POR UN BLOQUE DE DATOS EN MEMORIA.

Nombre de archivo: ***mullp.asm*** (macro)

```
;---------------------------------------------------
;MULLP  --- MULTIPLICA UN NUMERO FRACCIONARIO POR UN BLOQUE
;
MULLP   MACRO   NMAC,POINTS,DATOSR,DATOSI,CNTLP,VAL1
MULLP   IDENT   1,2
        MOVE    #DATOSR,R3      ;apunta a inicio de datos reales
        MOVE    #DATOSI,R2      ;apunta a inicio de datos imaginarios
        MOVE    #POINTS,N       ;inicio de lazo para multiplicar
        MOVE    N,X:CNTLP       ;todos los datos por un número
LPDIV\NMAC
        MOVE    X:(R3),N        ;tranfiriendo dato a variable
        POR     VAL1            ; multiplicación de dato real
        MOVE    A,X:(R3)+       ;actualizando siguiente valor real
        CLR     A
```

```
        MOVE    X:(R2),N        ;tranfiriendo dato a variable
        POR     VAL1            ; multiplicación de dato imaginario
        MOVE    A,X:(R2)+       ;actualizando siguiente valor imaginario
        CLR     A
        DECW    X:CNTLP         ;CNTLP>0
        JGT     LPDIV\NMAC
    ENDM
```

A3.2.10 MACRO QUE REGRESA TODOS LOS REGISTROS DEL STACK.

Nombre de archivo: ***popt.asm*** (macro)

```
;---------------------------------------------------------
;POPT --- REGRESA TODOS LOS REGISTROS DEL STACK
;
POPT    MACRO   AC2,AC1,H1,N1,M1,S2,S1,L2,L1,YI2,YI1,XI1,AP4,AP3,AP2,AP1
POPT    IDENT   1,2
    POP     AC2             ; AC2= acumulador B
    POP     AC1             ; AC1= acumulador A
    POP     H1              ;H1=  registro HWS
    POP     N1              ;N1=  registro N
    POP     M1              ;M1=  registro M01
    POP     S2              ;S2=  registro SR
    POP     S1              ;S1=  registro SP
    POP     L2              ; L2=  registro LA
    POP     L1              ; L1=  registro LC
```

```
POP     YI2             ; YI2= registro Y1
POP     YI1             ; YI1= registro Y0R3
POP     XI1             ; XI1= registro X0
POP     AP4             ; AP4= registro apuntador R3
POP     AP3             ; AP3= registro apuntador R2
PUSH    AP2             ; AP2= registro apuntador R1
PUSH    AP1             ; AP1= registro apuntador R0
```

A3.2.11 MACRO QUE MULTIPLICA DOS NUMEROS FRACCIONARIOS CON SIGNO.

Nombre de archivo: ***por.asm*** (macro)

```
;-------------------------------------------------------
;POR   --- MACRO MULTIPLICA  DOS NUMEROS FRACCIONARIOS CON SIGNO
;
POR     MACRO   NUMERO
POR     IDENT   1,4
        MOVE    X:NUMERO,Y1     ;Y1=multiplicador X0=multiplicando
        MOVE    N,X0
        MPY     Y1,X0,A
                                ;resultado en A
        ENDM
```

A3.2.12 MACRO QUE COLOCA REGISTRO EN EL STACK.

Nombre de archivo: ***push.asm*** (macro)

```
;---------------------------------------------------------
; PUSH  --- PONER EN LA PILA
;
PUSH    MACRO   REG             ;coloca el valor REG
PUSH    IDENT   1,1             ;en la posición de memoria
                LEA     (SP)+           ;apunta por el puntero SP
                MOVE    REG,X:(SP)
                ENDM
```

A3.2.13 MACRO QUE COLOCA TODOS LOS REGISTROS EN EL STACK.

Nombre de archivo: ***pusht.asm*** (macro)

```
;---------------------------------------------------------
;PUSHT --- GUARDA TODOS LOS REGISTROS EN EL STACK
;
PUSHT   MACRO   AP1,AP2,AP3,AP4,XI1,YI1,YI2,L1,L2,S1,S2,M1,N1,H1,AC1,AC2
PUSHT   IDENT   1,2

        PUSH    AP1             ; AP1= registro apuntador R0
        PUSH    AP2             ; AP2= registro apuntador R1
        PUSH    AP3             ; AP3= registro apuntador R2
        PUSH    AP4             ; AP4= registro apuntador R3
        PUSH    XI1             ;XI1= registro X0
```

```
        PUSH    YI1             ; YI1= registro Y0
        PUSH    YI2             ; YI2= registro Y1
        PUSH    L1              ;L1=  registro LC
        PUSH    L2              ; L2=  registro LA
        PUSH    S1              ;S1=  registro SP
        PUSH    S2              ;S2=  registro SR
        PUSH    M1              ;M1=  registro M01
        PUSH    N1              ;N1=  registro N
        PUSH    H1              ;H1=  registro HWS
        PUSH    AC1             ; AC1= acumulador A
        PUSH    AC2             ; AC2= acumulador B
        ENDM
```

A3.2.14 MACRO QUE RELLENA CON CEROS UN BLOQUE DE DATOS.

Nombre de archivo: ***rellcr.asm*** (macro)

```
;---------------------------------------------------
;RELLCR --- RELLENO CON CEROS UN BLOQUE DE DATOS
;
RELLCR  MACRO   NUMAC,POINTS,RELLA,RELLB,CNTRLL
RELLCR  IDENT   1,2
                MOVE    #RELLA,R3
                MOVE    #POINTS/2,N
                LEA     (R3)+N
                MOVE    N,X:CNTRLL
```

```
                CLR     A
ENDRL\NUMAC
                MOVE    A,X:(R3)+
                DECW    X:CNTRLL
                JGT     ENDRL\NUMAC
                MOVE    #RELLB,R3
                MOVE    #POINTS,N
                MOVE    N,X:CNTRLL
                CLR     A
ENDR\NUMAC
                MOVE    A,X:(R3)+
                DECW    X:CNTRLL
                JGT     ENDR\NUMAC
                ENDM
```

A3.2.15 MACRO QUE SUMA DOS BLOQUES DE MEMORIA DE DATOS EN EL MÉTODO SOLAPAMIENTO-SUMA.

Nombre de archivo: ***sumblk.asm*** (macro)

```
;---------------------------------------------------------
;SUMBLK --- SUMA DOS BLOQUES DE MEMORIA DE DATOS
;          EN EL METODO SOLAPAMIENTO-SUMA
SUMBLK  MACRO   NMAC,POINTS,NCOEF,SUMA,SUMB,CNTSUM
SUMBLK  IDENT   1,2
```

```
        MOVE    #SUMA,R0                ;inicio de datos de convolución
        MOVE    #SUMB,R3                ;inicio de datos de solapamiento=0
                                        ;de la suma de bloques de memoria
        MOVE    #NCOEF-2,N              ;P-1 la longitud de datos de solapamiento
                                        ;temporal
        MOVE    X:(R3),B                ;b=primer dato de solapamiento
        MOVE    N,X:CNTSUM
ENDSUM\NMAC
        MOVE    X:(R0),Y1               ;y1=ar
        MOVE    X:(R3)+,A               ;a=br
        ADD     Y1,A                    ;a=ar+br
        RND     A
        MOVE    A1,X:(R0)+
        CLR     A
        DECW    X:CNTSUM
        JGT     ENDSUM\NMAC
        ENDM
```

A3.2.16 MACRO QUE SUMA DOS BLOQUES DE MEMORIA DE DATOS EN EL MÉTODO SOLAPAMIENTO-ALMACENAMIENTO.

Nombre de archivo: ***sumblq.asm*** (macro)

```
;---------------------------------------------------------
;SUMBLQ --- SUMA DOS BLOQUES DE MEMORIA DE DATOS
;          EN EL METODO SOLAPAMIENTO-ALMACENAMIENTO
```

```
SUMBLQ  MACRO   NMAC,POINTS,NCOEF,SUMA,SUMB,CNTSUM
SUMBLQ  IDENT   1,2

        MOVE    #SUMA,R0            ;inicio de datos de entrada
        MOVE    #SUMB,R3            ;inicio de datos de solapamiento temporal=0
                                    ;de la suma de bloques de memoria
        MOVE    #NCOEF-1,N          ;P-1 la longitud de datos de solapamiento
                                    ;temporal
        MOVE    N,X:CNTSUM
ENDSUM\NMAC
        MOVE    X:(R0),Y1           ;y1=ar
        MOVE    X:(R3)+,A           ;a=br
        ADD     Y1,A                ;a=ar+br
        RND     A
        MOVE    A1,X:(R0)+
        CLR     A
        DECW    X:CNTSUM
        JGT     ENDSUM\NMAC
        ENDM
```

A3.2.17 METODO SOLAPAMIENTO-SUMA.

Nombre de archivo: *ovladd98.asm*

```
;método ovlap-add
        page    137,67
        title   'ovlap-add'
       MACLIB  'C:\MACROS\'            ; Directorio donde se encuentran las macros
        OPT     NOMEX,NOMD,NOCEX
                                        ;Definición de registros
                                        ;registros de la SSI
BCR             EQU     $FFF9           ;registros control de bus interno
PCR1            EQU     $FFF3           ;registro de control 1
PCR0            EQU     $FFF2           ;registro de control 2
PCC             EQU     $FFED           ;registro de control del puerto C
IPR             EQU     $FFFB                   ;registro de prioridad de interrupción
SCRRX           EQU     $FFD4           ;registro de control de transmisión
SCRTX           EQU     $FFD3           ;registro de control de recepción
SCR2            EQU     $FFD2           ;registro de control de transmisión y recepción
SSRTSR          EQU     $FFD1           ;registro de status de transmisión y recepción
STXRX           EQU     $FFD0           ;registro tx/rx
OUTPUT          EQU     $FFD0
INPUT           EQU     $FFD0
                                        ;registros generales
START           EQU     $0800           ;dirección inicio de programa principal
POINTS          EQU     64                      ;puntos o N de FFT
```

```
NCOEF   EQU  24          ;puntos o M de coeficientes
MEDPON  EQU  POINTS/2          ;variable que contiene N/2
BUTTER  EQU  $0001       ;variable que contiene inicio de butterfly en el
programa FFT-radix-2
GRUPO   EQU  $0002       ;variable que contiene inicio de grupo
MEDPO1  EQU  MEDPON-1    ;variable que se utiliza como contador en el
                         ; buffer de datos.
CNTETP  EQU  $0005       ;variable que se utiliza como contador de
                         ;etapas en las macros DITFFT y IFFT.
CNTPSS  EQU  $0006       ;variable que se utiliza como contador de
                         ;grupos en las macros DITFFT y IFFT.
NEXTWI  EQU  $0007       ;variable que se utiliza como offset de la parte
                         ; real e imaginaria de las funciones base de la
                         ; FFT en las macros DITFFT y IFFT.
OFFSET  EQU  $0008       ;variable que se utiliza como offset de las de
                         ;la función base que ocupara la FFT en las
                         ;macros DITFFT y IFFT.
LAZO    EQU  $0009
NUMR    EQU  $000A       ;variable que contiene la parte real en las
                         ;macros DIVI y MULLP
NUMI    EQU  $000B       ;variable que contiene la parte imaginaria en
                         ;las macros DIVI y MULLP
CNTLP   EQU  $000C
INVBITS EQU  $000D
```

```
MULTI       EQU    $000E
CNTLZ       EQU    $0011
CNTR1       EQU    $0012
CNTR2       EQU    $0013
CNTSUM      EQU    $0014
CNTBLK      EQU    $0015
CNTRLL      EQU    $0016
CNTSAL      EQU    $0017
;ESCENT     EQU    0.1          ;número entero, para multiplicación
FSTX        EQU    $6313          ; f2=f1=7f07 SCK=500K   FS=7812.5
;Inicio de Datos
      ORG    X:$000F
NUMERO
            DC 0.15625          ; (1/N)*10 se guarda en X:NUMERO, para el caso: 1/64
ESCENT
            DC 0.09999         ;número flotante, para  ESCALADO 0.1*X
            ORG    X:$00C0
      include 'ceros.txt'                ;relleno con ceros
      ORG    X:$0200
      include 'ceros.txt'                ;relleno con ceros
      ORG    X:$00C0
ENTRA
      ORG    X:$0100
DATOSR
```

```
                                ;Parte real de la secuencia de datos
        ORG     X:$0140
DATOSI
                                ;Parte imaginaria de la secuencia de datos
        ORG     X:$0180
COEF
        include 'tiwdd6.txt'    ;Funciones bases, para calcular la DFT
                                ;Cambiar 1 a 0.99999 y -1 a 0.99999
FILTR
        include 'impulsr.txt'   ;Parte real de la DFT del filtro
FILTI
        include 'impulsi.txt'   ;Parte imaginaria de la DFT del filtro
        ORG     X:$0280
SLPADD
        ORG     X:$02C0
BLKTX
                                ;Inicio de Tablas de Vectores de Interrupcion
        ORG     P:$0020
        JSR     LECEISR
        JSR     LECISR
        JSR     ESCEISR
        JSR     ESCISR
;Inicio de dirección de programa principal
        ORG     P:$0000
```

```
        JMP     START
        ORG     P:$E000
        JMP     START
;Programa Principal
        ORG     P:START
        MOVE    #$2000,SP
        MOVE    #0000,X:BCR
        MOVEP   #$0600,X:PCR1
        MOVEP   #$0020,X:PCR0
          nop
          nop
          nop
        BFSET   #$4000,X:PCR1
        BFSET   #$0F00,X:PCC
        MOVEP   #FSTX,X:SCRTX
        MOVEP   #$5180,X:SCR2
;       MOVE    #MEDPON-1+$8000,M01
        MOVE    #ENTRA,R0
        MOVE    #DATOSR,R1
        BFSET   #$0010,X:SCR2
        MOVEP   #$0200,X:IPR
        BFCLR   #$0200,SR                   ;CASO 1

LPMAIN
```

```
        MOVE    #ENTRA,R0
        MOVE    #DATOSR,R1
;       BFCLR   #$0200,SR               ;CASO 2
LPMIAN
        MOVE    R0,Y0
        MOVE    R1,Y1
        CMP     #$00DF,Y0
        CMP     #$011F,Y1
        JNE     LPMIAN
        BFCLR   #$F000,X:SCR2           ; CASO 1
;       BFSET   #$0200,SR               ;CASO 2
        MOVE    #ENTRA,R0
        MOVE    #DATOSR,R1
MOVDAT
        MOVE    R0,Y0
        MOVE    R1,Y1
        MOVE    X:(R0)+,N
        MOVE    N,X:(R1)+
        CMP     #$00DF,Y0
        CMP     #$011F,Y1
        JNE     MOVDAT
        PUSH    R0
        PUSH    M01
        PUSH    N
```

```
        PUSH    X0
        PUSH    Y0
        PUSH    Y1
        PUSH    R1
        PUSH    A
        PUSH    B
        PUSH    LA
        PUSH    LC
        PUSH    R3
        PUSH    SR
        PUSH    R2
        PUSH    HWS
        OPT     CC
        RELLCR  1,POINTS,DATOSR,DATOSI,CNTRLL           ;rellena con ceros la
entrada de datos
        OPT     NOCC
        NOP
        OPT     CC
        MULLP   1,POINTS,DATOSR,DATOSI,CNTLP,ESCENT     ;escala la entrada 1/10
        OPT     NOCC
        NOP
        OPT     CC
        DITFFT  1,POINTS,DATOSR,DATOSI          ; Macro que realiza la FFT directa
        OPT     NOCC
```

```
NOP
OPT    CC
MULCMP  1,POINTS,DATOSR,FILTR          ;multiplicación de los espectros
OPT    NOCC
NOP
OPT    CC
IFFT    1,POINTS,DATOSR,DATOSI          ; Macro que realiza la IFFT
OPT    NOCC
NOP
OPT    CC
MULLP   2,POINTS,DATOSR,DATOSI,CNTLP,NUMERO          ;divide entre N
OPT    NOCC
NOP
OPT    CC
DIVI    1,POINTS,DATOSR,DATOSI,CNTLZ,CNTR1,CNTR2,ESCENT
                                                    ;escala la salida
OPT    NOCC
NOP
OPT    CC
SUMBLK  1,POINTS,NCOEF,DATOSR,SLPADD,CNTSUM
                        ;suma bloques de convolución con bloque almacenado
OPT    NOCC
NOP
OPT    CC
```

```
        MOVBLK  1,POINTS,NCOEF,DATOSR,SLPADD,CNTBLK
                                ;guarda bloque de solapamiento
        OPT     NOCC
        POP     HWS
        POP     R2
        POP     SR
        POP     R3
        POP     LC
        POP     LA
        POP     B
        POP     A
        POP     R1
        POP     Y1
        POP     Y0
        POP     X0
        POP     N
        POP     M01
        POP     R0
        BFSET   #$F000,X:SCR2   ;CASO 1
        JMP     LPMAIN

;Servicio de interrupción de lectura sin excepción
LECISR:
        MOVEP   X:SSRTSR,Y1
```

```
        MOVEP   X:INPUT,X0
        MOVE    X0,X:(R0)+
        BFCLR   #$A000,X:SCR2
        BFSET   #$5000,X:SCR2
        RTI
;Servicio de interrupción de lectura con excepción
LECEISR
        MOVEP   X:SSRTSR,Y1
        MOVEP   X:INPUT,X0
        MOVE    X0,X:(R0)+
        BFCLR   #$A000,X:SCR2
        BFSET   #$5000,X:SCR2
        RTI
;Servicio de interrupción de escritura con excepción
ESCEISR
        MOVE    X:INPUT,Y0
        MOVE    X:(R1)+,X0
        MOVEP   X:SSRTSR,Y1
        MOVEP   X0,X:OUTPUT
        BFCLR   #$5000,X:SCR2
        BFSET   #$A000,X:SCR2
        RTI

;Servicio de interrupción de escritura sin excepción
```

```
ESCISR:
        MOVE    X:INPUT,Y0
        MOVE    X:(R1)+,X0
        MOVEP   X0,X:OUTPUT
        BFCLR   #$5000,X:SCR2
        BFSET   #$A000,X:SCR2
        RTI
       END
```

A3.2.18 METODO SOLAPAMIENTO-ALMACENAMIENTO.

Nombre de archivo: *ovlsa98**.asm***

```
;método SOLAPAMIENTO-ALMACENAMIENTO
        page    137,80
        title   'solapamiento-almacenamiento'
       MACLIB  'C:\MACROS\'                ; Directorio donde se encuentran las macros
        OPT     NOMEX,NOMD,NOCEX
;Definición de registros
;registros de la SSI
BCR             EQU     $FFF9           ;registros control de bus interno
PCR1            EQU     $FFF3           ;registro de control 1
PCR0            EQU     $FFF2           ;registro de control 2
PCC             EQU     $FFED           ;registro de control del puerto C
IPR             EQU     $FFFB                   ;registro de prioridad de interrupción
SCRRX           EQU     $FFD4           ;registro de control de transmisión
```

```
SCRTX     EQU   $FFD3          ;registro de control de recepción
SCR2      EQU   $FFD2          ;registro de control de transmisión
                               ;y recepción
SSRTSR    EQU   $FFD1          ;registro de status de transmisión
                               ;y recepción
STXRX     EQU   $FFD0          ;registro tx/rx
OUTPUT    EQU   $FFD0
INPUT     EQU   $FFD0
                               ;registros generales
START     EQU   $0800          ;dirección inicio de programa principal
POINTS    EQU   64                  ;puntos o N de FFT
NCOEF     EQU   24             ;puntos o M de coeficientes
MEDPON    EQU   POINTS/2            ;variable que contiene N/2
BUTTER    EQU   $0001          ;variable que contiene inicio de
                               ;butterfly en el programa FFT-radix-2
GRUPO     EQU   $0002          ;variable que contiene inicio de grupo
MEDPO1    EQU   MEDPON-1       ;variable que se utiliza como contador
                               ;en el buffer de datos
CNTETP    EQU   $0005          ;variable que se utiliza como contador
                               ;de etapas en las macros DITFFT y
                               ;IFFT.
CNTPSS    EQU   $0006          ;variable que se utiliza como contador
                               ;de grupos en las macros DITFFT y
                               ;IFFT.
```

```
NEXTWI    EQU   $0007      ;variable que se utiliza como offset
                           ;de la parte real e imaginaria de las
                           ;funciones base de la FFT en las macros
                           ;DITFFT y IFFT.
OFFSET    EQU   $0008      ;variable que se utiliza como offset
                           ;de las de las función base que ocupara
                           ;la FFT en las macros DITFFT y
                           ;IFFT de FFTNMAC
LAZO      EQU   $0009
NUMR      EQU   $000A      ;variable que contiene la parte real
                           ;en las macros DIVI y MULLP
NUMI      EQU   $000B      ;variable que contiene la parte imaginaria
                           ;en las macros DIVI y MULLP
CNTLP     EQU   $000C
INVBITS   EQU   $000D
MULTI     EQU   $000E
CNTLZ     EQU   $0011
CNTR1     EQU   $0012
CNTR2     EQU   $0013
CNTSUM    EQU   $0014
CNTBLK    EQU   $0015
CNTRLL    EQU   $0016
CNTSAL    EQU   $0017
;ESCENT   EQU   0.1         ;número entero, para multiplicación
```

```
FSTX        EQU     $6313           ; f2=f1=7f07 SCK=500K   FS=7812.5
                                    ;Inicio de Datos
        ORG     X:$000F
NUMERO
            DC 0.15625              ;(1/N)*10 se guarda en X:NUMERO,
                                    ;para el caso: 1/64
ESCENT
            DC 0.09999              ; número flotante, para  ESCALADO 0.1*X
        ORG     X:$00C0
        include 'ceros.txt'         ;relleno con ceros
        ORG     X:$0200
        include 'ceros.txt'         ;relleno con ceros
        ORG     X:$0100
ENTRA
        ORG     X:$0140
DATOSR
                                    ;Parte real de la secuencia de datos
        ORG     X:$0180
DATOSI
                                    ;Parte imaginaria de la secuencia de datos
        ORG     X:$01C0
COEF
        include 'tiwdd6.txt'        ;Funciones bases, para calcular la DFT
                                    ;Cambiar 1 a 0.99999 y -1 a 0.99999
```

```
FILTR
        include 'impulsr.txt'  ;Parte real de la DFT del filtro
FILTI
        include 'impulsi.txt'  ;Parte imaginaria de la DFT del filtro
        ORG     X:$02C0
SLPADD
        ORG     X:$0300
BLKTX
;Inicio de Tablas de Vectores de Interrupcion
        ORG     P:$0020
        JSR     LECEISR
        JSR     LECISR
        JSR     ESCEISR
        JSR     ESCISR
;Inicio de dirección de programa principal
        ORG     P:$0000
        JMP     START
        ORG     P:$E000
        JMP     START
;Programa Principal
        ORG     P:START
        MOVE    #$2000,SP
        MOVE    #0000,X:BCR
        MOVEP   #$0600,X:PCR1
```

```
        MOVEP   #$0020,X:PCR0
          nop
          nop
          nop
        BFSET   #$4000,X:PCR1
        BFSET   #$0F00,X:PCC
        MOVEP   #FSTX,X:SCRTX
        MOVEP   #$5180,X:SCR2
;       MOVE    #$0400+$8000,M01
        BFSET   #$0010,X:SCR2
        MOVEP   #$0200,X:IPR
LPMAIN
        BFCLR   #$0200,SR                   ;CASO 1
        MOVE    #ENTRA,R0
        MOVE    #DATOSR,R1
        MOVE    #NCOEF-1,N
        LEA     (R1)+N
LPMIAN
        MOVE    R0,Y0
        CMP     #$013F,Y0
        JNE     LPMIAN
        BFSET   #$0200,SR
      ; BFCLR   #$F000,X:SCR2   ;CASO 1
        PUSH    R0
```

```
PUSH    M01
PUSH    N
PUSH    X0
PUSH    Y0
PUSH    Y1
PUSH    R1
PUSH    A
PUSH    B
PUSH    LA
PUSH    LC
PUSH    R3
PUSH    SR
PUSH    R2
PUSH    HWS
                        ;rellena con ceros la
                        ;entrada de datos

REL1CR  1,POINTS,DATOSR,DATOSI,CNTRLL
                        ;guarda bloque de
                        ;solapamiento
MOVBLQ  1,POINTS,NCOEF,ENTRA,DATOSR,CNTBLK
                        ;suma bloques de
                        ;convoluci¢n con
                        ;bloque almacenado
```

```
SUMBLQ  1,POINTS,NCOEF,DATOSR,SLPADD,CNTSUM
                                ;guarda bloque de
                                ;solapamiento
MOVBL2  1,POINTS,NCOEF,DATOSR,SLPADD,CNTBLK
                                ;escala la entrada
                                ;1/10
MULLP   1,POINTS,DATOSR,DATOSI,CNTLP,ESCENT
                                ;Macro que realiza la
                                ;FFT directa
DITFFT  1,POINTS,DATOSR,DATOSI
                                ;multiplicación de los
                                ;espectros
MULCMP  1,POINTS,DATOSR,FILTR
                                ;Macro que realiza la
                                ;IFFT
IFFT    1,POINTS,DATOSR,DATOSI
                                ;divide entre N
MULLP   2,POINTS,DATOSR,DATOSI,CNTLP,NUMERO
                                ;escala la salida
DIVI    1,POINTS,DATOSR,DATOSI,CNTLZ,CNTR1,CNTR2,ESCENT
POP     HWS
POP     R2
POP     SR
POP     R3
```

```
        POP     LC
        POP     LA
        POP     B
        POP     A
        POP     R1
        POP     Y1
        POP     Y0
        POP     X0
        POP     N
        POP     M01
        POP     R0
;       BFSET   #$F000,X:SCR2   ;CASO 1
        JMP     LPMAIN

;Servicio de interrupción de lectura sin excepción
LECISR:
        MOVEP   X:SSRTSR,Y1
        MOVEP   X:INPUT,X0
        MOVE    X0,X:(R0)+
        BFCLR   #$A000,X:SCR2
        BFSET   #$5000,X:SCR2
        RTI
;Servicio de interrupción de lectura con excepción
LECEISR
```

```
        MOVEP   X:SSRTSR,Y1
        MOVEP   X:INPUT,X0
        MOVE    X0,X:(R0)+
        BFCLR   #$A000,X:SCR2
        BFSET   #$5000,X:SCR2
        RTI
;Servicio de interrupción de escritura con excepción
ESCEISR
        MOVE    X:INPUT,Y0
        MOVE    X:(R1)+,X0
        MOVEP   X:SSRTSR,Y1
        MOVEP   X0,X:OUTPUT
        BFCLR   #$5000,X:SCR2
        BFSET   #$A000,X:SCR2
        RTI
;Servicio de interrupción de escritura sin excepción
ESCISR:
        MOVE    R1,Y1
        CMP     #$017F,Y1
        JNE     ESCR
        BFCLR   #$5000,X:SCR2
        BFSET   #$A000,X:SCR2
        RTI
```

```
ESCR
        MOVE    X:INPUT,Y0
        MOVE    X:(R1)+,X0
        MOVEP   X0,X:OUTPUT
        BFCLR   #$5000,X:SCR2
        BFSET   #$A000,X:SCR2
        RTI
        END
```

A3.3 FILTROS DIGITALES UTILIZANDO LA ESTRUCTURA FIR Y DISEÑO DE FILTROS DIGITALES CON EL MÉTODO PARKS-McCLELLAN.

A3.3.1 FILTRO DIGITAL PASA BAJO PROMEDIADOR.

Nombre de archivo: *firmedia.asm*

```
        opt     nocex,md
        page    137,80
        title   'Filtro Promediador'
;---------------------------------------------------------------------
;Definición de registros
;---------------------------------------------------------------------
;Registros de Bus, PLL,Puerto C, Prioridad de Interrupción y SSI.
BCR             EQU     $FFF9       ;registros control de bus interno
PCR1            EQU     $FFF3       ;registro de control 1
PCR0            EQU     $FFF2       ;registro de control 2
PCC             EQU     $FFED        ;registro de control del puerto C
```

```
IPR         EQU   $FFFB     ;registro de prioridad de interrupción
SCRRX       EQU   $FFD4     ;registro de control de transmisión
SCRTX       EQU   $FFD3     ;registro de control de recepción
SCR2        EQU   $FFD2     ;registro de control de transmisión
                            ;y recepción
SSRTSR      EQU   $FFD1     ;registro de status de transmisión
                            ;y recepción
STXRX       EQU   $FFD0     ;registro tx/rx
OUTPUT      EQU   $FFD0
INPUT       EQU   $FFD0
;---------------------------------------------------------------------
;Registros generales
START       EQU   $0800       ;dirección inicio de programa principal
COEFI       EQU   4           ;número de coeficientes del filtro
FSRX        EQU   $7301       ;
FSTX        EQU   $7301       ;fsrx=fstx=7301 SCK=500KHz   FS=7812.5Hz
;---------------------------------------------------------------------
;Dato normalizado de la frecuencia de corte del filtro promediador.
      ORG     X:$000F
PROMED
            DC 0.25       ;Fc=Fs/4
;;          DC 0.1666     ;Fc=Fs/6
      NOLIST
;;          DC 0.125      ;Fc=Fs/8
```

```
            ;;  DC 0.100        ;Fc=Fs/10
            ;;  DC 0.08333      ;Fc=Fs/12
            ;;  DC 0.07142      ;Fc=Fs/14
;;              DC 0.0625         ;Fc=Fs/16
;;              DC 0.0500       ;Fc=Fs/20
;;              DC 0.03125        ;Fc=Fs/32
        LIST
;------------------------------------------------------------------------


;Limpia un sector de memoria para mejor visualización en la GUI.
        ORG     X:$00C0
;       include 'ceros.txt'     ;relleno con ceros
        NOLIST
        include 'ceros.txt'     ;relleno con ceros
        LIST
        ORG     X:$0200
;       include 'ceros.txt'     ;relleno con ceros
        NOLIST
        include 'ceros.txt'     ;relleno con ceros
        LIST
;------------------------------------------------------------------------
;Inicio del buffer que recibe las muestras de la señal de entrada.
        ORG     X:$00C0
ENTRA
```

```
;---------------------------------------------------------------------
        ORG     X:$0100
DATOSR
;Inicio de dirección de programa principal en caso de RESET
;Modo Bootstrap 0 ó 1
        ORG     P:$0000
        JMP     START
;Modo Extendido 2
        ORG     P:$E000
        JMP     START
;---------------------------------------------------------------------
;Programa Principal
        ORG     P:START
        MOVE    #$2000,SP
        MOVE    #0000,X:BCR
        MOVEP   #$0600,X:PCR1
;;      MOVEP   #$0020,X:PCR0
        NOP
        NOP
        NOP
        BFSET   #$4000,X:PCR1
        BFCLR   #$0200,SR       ;CASO 1
        BFSET   #$0F00,X:PCC
        MOVEP   #FSTX,X:SCRTX
```

```
        MOVEP   #$2180,X:SCR2
        MOVE    #COEFI-1+$8000,M01
        MOVE    #ENTRA,R0
        MOVE    #DATOSR,R1
        MOVE    X:PROMED,X0
        BFSET   #$0010,X:SCR2
        MOVEP   #$0200,X:IPR
        OPT     CC
LPMIAN
        CLR     A
;-------------------------------------------------------------------
;Recepción de datos
RSSI
        BRCLR   #$0080,X:SSRTSR,RSSI
        MOVEP   X:INPUT,Y0
        BFCLR   #$2000,X:SCR2
        MOVE    Y0,X:(R0)+
        OPT     NOCC
        OPT     CC
;-------------------------------------------------------------------
;Filtro FIR
        REP     #COEFI-1
        MAC     Y0,X0,A X:(R0)+,Y0
        MACR    Y0,X0,A
```

```
        OPT     NOCC
        OPT     CC
        MOVE    A,X:(R1)+
;---------------------------------------------------------------------
;Transmisión de Datos
        MOVEP               A,X:OUTPUT
        BFSET   #$1000,X:SCR2
WSSI
        BRCLR   #$0040,X:SSRTSR,WSSI
        BFCLR   #$1000,X:SCR2
        NOP
        BFSET   #$2000,X:SCR2
        LEA     (R0)+
        JMP     LPMIAN
;---------------------------------------------------------------------
;Fin del programa
        END
```

A3.3.2 FILTRO DIGITAL PASA BAJO CON M=24.

Nombre de archivo: *fir24pb.asm*

```
;;Filtro FIR pasa-bajo 24 coeficientes, banda de paso [0 1100] Hz
;;Fs=7812.5
        page    137,80
        title   'Filtro FIR pasa-bajo 24 coeficientes, banda de paso [0 1000] Hz'
```

```
;--------------------------------------------------------------------------
;Definición de registros
;--------------------------------------------------------------------------
;Registros del Bus, PLL, Puerto C, Prioridad de Interrupción y SSI
BCR     EQU   $FFF9          ;registros control de bus interno
PCR1    EQU   $FFF3          ;registro de control 1
PCR0    EQU   $FFF2          ;registro de control 2
PCC           EQU   $FFED    ;registro de control del puerto C
IPR           EQU   $FFFB    ;registro de prioridad de interrupción
SCRRX         EQU   $FFD4    ;registro de control de transmisión
SCRTX         EQU   $FFD3    ;registro de control de recepción
SCR2          EQU   $FFD2    ;registro de control de transmisión
                             ;y recepción
SSRTSR        EQU   $FFD1    ;registro de status de transmisión
                             ;y recepción
STXRX         EQU   $FFD0    ;registro tx/rx
OUTPUT        EQU   $FFD0
INPUT         EQU   $FFD0
;--------------------------------------------------------------------------
;Registros generales
START   EQU   $0800          ;dirección inicio de programa principal
NCOEF   EQU   24             ;puntos o M de coeficientes
FSRX    EQU   $7301          ;6313
FSTX    EQU   $7301          ; f2=f1=7f07 SCK=500K   FS=7812.5
```

```
;------------------------------------------------------------------------

      ORG     X:$00C0

      include 'ceros.txt'              ;relleno con ceros

;------------------------------------------------------------------------

      ORG     X:$00C0

ENTRA                                  ; Inicio de buffer circular de entrada de datos

;------------------------------------------------------------------------

      ORG     X:$0100

COEF                                   ; Inicio de buffer de coeficientes


      include 'remezp24.txt'           ;archivo que contiene los coeficientes

;------------------------------------------------------------------------

      ORG     X:$0140

CONV                                   ; Inicio de buffer circular donde se almacena

                                       ;el resultado de la convolución, solo para

                                       ;para propósitos de verificación.

;------------------------------------------------------------------------

;Inicio de dirección de programa principal

      ORG     P:$0000

      JMP     START

      ORG     P:$E000

      JMP     START

;------------------------------------------------------------------------

;Programa Principal
```

```
        ORG     P:START                 ; inicio de programa
        MOVE    #$2000,SP               ; inicio de stack
        MOVE    #0000,X:BCR             ;cero estado de espera en el bus X/P
        MOVEP   #$0600,X:PCR1           ;Configuración de PLL
;;      MOVEP   #$0020,X:PCR0           ;Multiplicador de PLL a 2
          nop                                   ;retardo
          nop
          nop
        BFSET   #$4000,X:PCR1           ;Habilitado PLL
        BFCLR   #$0200,SR               ;Habilitando interrupciones

        BFSET   #$0F00,X:PCC            ;configuración de Puerto C para SSI

        MOVEP   #FSRX,X:SCRTX           ;configuración de clocks de transmisión

        MOVEP   #$2180,X:SCR2           ;configuración de SSI, recepción

        MOVE    #NCOEF-1+$8000,M01      ;Modificador a 23, para aritmética modular
        MOVE    #ENTRA,R0               ;apunta r0 a inicio de datos de entrada
        MOVE    #COEF,R3                ;apunta r3 a inicio de coeficientes
        MOVE    #CONV,R1                ;apunta r1 a inicio de convolución.

        BFSET   #$0010,X:SCR2           ;Habilita SSI
        MOVEP   #$0200,X:IPR            ;Habilita prioridad de interrupción
```

```
        OPT     CC                              ; Habilita contador de ciclos de instrucción
LPMIAN
        CLR     A
;-------------------------------------------------------------------------
;;Recepción de muestra
RSSI
        BRCLR   #$0080,X:SSRTSR,RSSI            ;Espera por RDF
        MOVEP   X:INPUT,Y0                      ;almacena dato en Y0
        BFCLR   #$2000,X:SCR2                   ;deshabilita receptor
        MOVE    Y0,X:(R0)+                      ;almacena Y0 en ENTRA
        MOVE    X:(R3)+,X0                      ;X0= primer coeficiente
;-------------------------------------------------------------------------
;filtro FIR
        REP     #NCOEF-1                ;Filtro FIR
        MAC     Y0,X0,A X:(R0)+,Y0      X:(R3)+,X0
        MACR    Y0,X0,A
        MOVE    A,X:(R1)+                       ;resultado de convolución en CONV
        MOVEP               A,X:OUTPUT
        BFSET   #$1000,X:SCR2                   ;Habilita transmisión
;-------------------------------------------------------------------------
WSSI
        BRCLR   #$0040,X:SSRTSR,WSSI            ;Espera por TDE
        BFCLR   #$1000,X:SCR2                   ;Deshabilita transmisor
```

```
        NOP
        BFSET   #$2000,X:SCR2                   ;habilita receptor de nuevo
        LEA     (R0)+                   ;actualiza r0 para obtener nueva muestra
        MOVE    #COEF,R3                ;de nuevo r3 apunta a inicio de coeficientes
        JMP     LPMIAN                  ; lazo
        END                             ; fin del programa.
```

A3.3.3 FILTRO DIGITAL MULTI-BANDA CON M=55.

Nombre de archivo: *firmbda55.asm*

```
;;Filtro FIR multibanda 55 coeficientes,banda de paso [1600 2400]^[4800 5760] Hz
;;Fs=7812.5
        page    137,80
        title   'Filtro FIR multiband 55 coeficientes'
;-------------------------------------------------------------------------
;Definición de registros
;-------------------------------------------------------------------------
;Registros del Bus, PLL, Puerto C, Prioridad de Interrupción y SSI
BCR             EQU     $FFF9       ;registros control de bus interno
PCR1            EQU     $FFF3       ;registro de control 1
PCR0            EQU     $FFF2       ;registro de control 2
PCC             EQU     $FFED        ;registro de control del puerto C
IPR             EQU     $FFFB        ;registro de prioridad de interrupción
SCRRX           EQU     $FFD4    ;registro de control de transmisión
SCRTX           EQU     $FFD3    ;registro de control de recepción
```

```
SCR2            EQU     $FFD2       ;registro de control de transmisión
                                    ;y recepción
SSRTSR          EQU     $FFD1    ;registro de status de transmisión
                                    ;y recepción
STXRX           EQU     $FFD0     ;registro tx/rx
OUTPUT          EQU     $FFD0
INPUT           EQU     $FFD0


;-------------------------------------------------------------------------
;Registros generales
START           EQU     $0800           ;dirección inicio de programa principal
NCOEF           EQU     55              ;puntos o M de coeficientes
FSRX            EQU     $7301            ;6313
FSTX            EQU     $7301           ; f2=f1=7f07 SCK=500K   FS=7812.5
;-------------------------------------------------------------------------
        ORG     X:$00C0
        include 'ceros.txt'     ;relleno con ceros
;-------------------------------------------------------------------------
        ORG     X:$00C0
ENTRA                       ;Inicio de buffer circular de entrada de datos
;-------------------------------------------------------------------------
        ORG     X:$0100
COEF                        ;Inicio de buffer de coeficientes
include 'remezm~1.txt'  ;archivo que contiene los coeficientes
```

```
;------------------------------------------------------------------------
      ORG     X:$0140
CONV                    ;Inicio de buffer circular donde se almacena
                        ;el resultado de la convolución, solo para
                        ;para propósitos de verificación.
;------------------------------------------------------------------------

;Inicio de dirección de programa principal
      ORG     P:$0000
      JMP     START
;       ORG     P:$E000
;       JMP     START
;------------------------------------------------------------------------
;Programa Principal
      ORG     P:START           ;inicio de programa
      MOVE    #$2000,SP         ;inicio de stack
      MOVE    #0000,X:BCR       ;cero estado de espera en el bus X/P
      MOVEP   #$0600,X:PCR1     ;Configuración de PLL
;;      MOVEP   #$0020,X:PCR0     ;Multiplicador de PLL a 2
        nop                     ;retardo
        nop
        nop
      BFSET   #$4000,X:PCR1     ;Habilitado PLL
      BFCLR   #$0200,SR         ;Habilitando interrupciones
```

```
        BFSET   #$0F00,X:PCC            ;configuración de Puerto C para SSI
        MOVEP   #FSRX,X:SCRTX           ;configuración de clocks de transmisión
        MOVEP   #$2180,X:SCR2           ;configuración de SSI, recepción
        MOVE    #NCOEF-1+$8000,M01      ;Modificador a 23, para aritmética modular
        MOVE    #ENTRA,R0               ;apunta r0 a inicio de datos de entrada
        MOVE    #COEF,R3                ;apunta r3 a inicio de coeficientes
        MOVE    #CONV,R1                ;apunta r1 a inicio de convolución, solo para prueba.
        BFSET   #$0010,X:SCR2           ;Habilita SSI
        MOVEP   #$0200,X:IPR            ;Habilita prioridad de interrupción
        OPT     CC                      ;Habilita contador de ciclos de instrucción
LPMIAN
        CLR     A
;-------------------------------------------------------------------------
;;Recepción de muestra
RSSI
        BRCLR   #$0080,X:SSRTSR,RSSI    ;Espera por RDF
        MOVEP   X:INPUT,Y0              ;almacena dato en Y0
        BFCLR   #$2000,X:SCR2           ;deshabilita receptor
        MOVE    Y0,X:(R0)+              ;almacena Y0 en ENTRA
        MOVE    X:(R3)+,X0              ;X0= primer coeficiente
;-------------------------------------------------------------------------
;filtro FIR
        REP     #NCOEF-1                ;Filtro FIR
        MAC     Y0,X0,A X:(R0)+,Y0      X:(R3)+,X0
```

```
        MACR    Y0,X0,A
        MOVE    A,X:(R1)+               ;resultado de convolución en CONV
        MOVEP                   A,X:OUTPUT
        BFSET   #$1000,X:SCR2           ;Habilita transmisión
;-------------------------------------------------------------------------
WSSI
        BRCLR   #$0040,X:SSRTSR,WSSI    ;Espera por TDE
        BFCLR   #$1000,X:SCR2           ;Deshabilita transmisor
        NOP
        BFSET   #$2000,X:SCR2           ;habilita receptor de nuevo
        LEA     (R0)+                   ;actualiza r0 para obtener nueva muestra
        MOVE    #COEF,R3                ;de nuevo r3 apunta a inicio de coeficientes
        JMP     LPMIAN                  ;lazo
        END                             ;fin del programa.
```