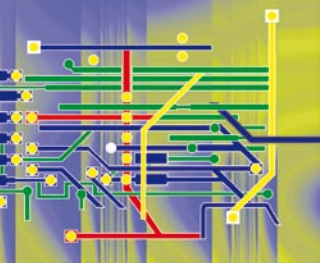


VISIÓN DE CASO

Clasificación del artículo:
REVISIÓN

TÉCNICAS PARA LA DISMINUCIÓN DEL TIEMPO DE EJECUCIÓN EN ALGORITMOS DE PROCESAMIENTO EN EL DOMINIO DE LA FRECUENCIA

Erwin John Saavedra Mercado*

Resumen

En el presente artículo se presentan un conjunto de técnicas y algoritmos que pueden ser utilizados para incorporar el procesamiento de señales en sistemas embebidos. Su propósito es informar sobre variantes o alternativas a los algoritmos usuales para lograr mejor desempeño de operación o para inclusive poder efectuar este tipo de cálculos en procesadores sencillos.

Palabras clave

STFT, sistemas embebidos, transformada de Fourier.

Summary

In the present article a set of techniques and algorithms are shown that can be used to incorporate the signal processing in embedded systems. Its intention is to inform on variants or alternatives to usual algorithms to obtain better operation performance or even to be able to carry out this type of calculations in simple processors.

Key words

STFT, Embedded Systems, Transformed of Fourier.

1. Introducción

El procesamiento digital de señales forma parte cada día más de nuestras aplicaciones electrónicas. El aumento en el desempeño tanto en resolución como en velocidad de cálculo en los procesadores ha permitido el advenimiento de un sin número de aplicaciones que hacen uso de técnicas matemáticas avanzadas, tales como: compresión de datos y análisis en el dominio de la frecuencia, entre otras. Hoy en día encontramos reproductores de MP3 del tamaño de un lapicero, con la capacidad de almacenar cientos de canciones con una calidad bastante satisfactoria Además, se encuentran teléfonos celulares con cámaras de video incorporadas, radio digital y, en resumen, toda una serie de artículos de uso diario y personal.

Sin embargo, en el momento de implementar este tipo de aplicaciones en sistemas electrónicos portátiles y que probablemente funcionarán a base de baterías, ciertas limita-

* Ingeniero en Control Electrónico e Instrumentación, Unidades Tecnológicas de Santander. Correo electrónico: esaavedra@ieee.org.

ciones deben ser consideradas. Entre estas limitaciones se deben señalar principalmente, la capacidad de almacenamiento y velocidad de cálculo de los sistemas. La primera se debe a la necesidad de disminuir costos e inclusive a la de disminuir tamaño; aunque se puede argumentar que en cantidades adecuadas, el costo puede disminuir considerablemente, el aspecto del tamaño y la "portabilidad" suele ser más importante. La segunda se debe principalmente al consumo de corriente, hecho que es aproximadamente proporcional a la velocidad de funcionamiento de los procesadores. La combinación de estos dos factores con la rotulación de una posible producción a nivel nacional, hecho que descarta la producción de altísimos volúmenes (con algunas excepciones), se resume en el hecho de que se utilizarán procesadores de baja velocidad y bajo costo.

2. Los sistemas embebidos

Un sistema embebido en un sistema electrónico conformado por una unidad (o unidades) de procesamiento, tanto CPU como ALU, memoria, periféricos y todo lo necesario para ser de este sistema un sistema *stand-alone*, es decir, que no necesite ningún componente externo para su funcionamiento. En lo que concierne a los procesadores, estos vienen como sólo procesadores, procesadores más memoria interna y procesadores embebidos o microcontroladores (también DSC o controladores digitales de señales, una mejora al tender un DSP y agregarle periféricos). En el momento se discutirá sobre los dos últimos, ya que el primero requiere de la construcción de un macrosistema, hecho que actualmente se asocia con costos altos y no clasifica en las limitaciones de tamaño y potencia que queremos afrontar.

Existen diferentes empresas fabricantes de procesadores embebidos, ya sean microcontroladores o DSP o, en general, procesadores embebidos. En todo momento están en competición por el mercado de diseño y fabricación de productos y, en realidad, se encuentran disminuciones de costos y aumento de especificaciones. Sin embargo, recuérdese que aunque las especificaciones puedan mejorar y los precios disminuir, el problema del consumo de corriente permanece al menos por el momento inmejorable. En fin, se tiene todo un portafolio de productos que van desde procesadores de 8 bits a 1MIPS[1] y con consumos de corriente debajo de los 5mA hasta procesadores digitales de señales de 32 bits a 150MIPS o a 16 bits a aproximadamente 8GMACS[2]. Un DSP como el último se puede hallar en el rango de US$150 en cantidades relativamente considerables. Los microcontroladores de bajo desempeño suelen hallarse en el rango de US$1 a US$5. En general, existen excepciones para la mayoría de parámetros, ya que los fabricantes siempre tienen algún producto orientado a cubrir aplicaciones con exigencias extremas, pero obviamente este tipo de excepciones poseen costos prohibitivos.

En lo que corresponde a la memoria, estos dispositivos poseen desde 64 bytes hasta 4kWords de memoria SRAM disponible para datos. Esta memoria es utilizada para almacenar las señales sobre las cuales se realizará el filtraje, buffers temporales mientras se comprime o almacenarán los resultados de algoritmos tales como: la transformada de Fourier. En lo que concierne a los microcontroladores de bajo costo y bajo consumo de corriente, estos no suelen poseer más de escasos 128 bytes de RAM, hecho que limita mucho su aplicabilidad. Otros aspectos que afectan el desempeño de los algoritmos son:

1 MIPS: Millones de Instrucciones Por Segundo.

2 GMAC: Billones de instrucciones de Multiplicación y Acumulación por segundo. Se puede pensar de las GMAC o MMAC como una unidad más adecuada de los MIPS si se observa desde el punto de vista del desempeño de cálculo del procesador.

1. La resolución no suele ser mayor a 16 bits.
2. Los procesadores suelen ser de punto fijo (los de punto flotante son escasos y de precios considerablemente mayores).
3. La comunicación a periféricos externos se hace a través de enlaces seriales que al final son de baja velocidad.
4. Los periféricos embebidos poseen especificaciones un poco inferiores a los periféricos externos, p.e.: conversores AD.
5. La lectura de memoria Flash tiene limitantes de velocidad y, en general, no puede ser leída sin estados de espera a velocidades superiores de 60MHz en sistemas con pipelines.

Estas y otras limitaciones afectan activamente el desempeño de un sistema embebido. Por ejemplo, el hecho de que los procesadores son de punto fijo implica que cualquier operación que requiera de cálculos hechos con valores decimales deberá ser emulada y una operación sencilla –como una multiplicación– pasa de tener una sola instrucción de máquina (en punto fijo), a tener una cantidad mayor de 30 (en punto flotante, y dependiendo del procesador y el compilador).

Las limitaciones en la resolución y el hecho de trabajar en punto fijo conllevan a otro problema, el de desbordamiento. Aunque éste no es tan crítico, como el de las operaciones de punto flotante, suele representar inconvenientes. La programación en lenguaje máquina permite afrontarlo de forma sencilla, pero la programación en lenguaje máquina implica la no portabilidad del código y demora mucho el desarrollo. Cuando se trabaje en lenguajes de alto nivel se debe tener cuidado con el desboradamiento, ya que el compilador no suele hacerlo.

3. Implementación de valores decimales en procesadores de punto fijo

Uno de los principales problemas de implementar algoritmos de procesamiento de señales es la alta cantidad de cálculos necesarios. Cuando esto se suma a la utilización de sistemas embebidos de punto fijo, el desempeño de éste puede verse disminuido notablemente. En general, se debe procurar trabajar con el sistema nativo de cálculo de cada procesador. Si éste es uno de punto fijo, se debe procurar evitar realizar cálculos en punto flotante, ya que estos deben ser emulados y consumirán mucho tiempo y código.

Una representación útil de los valores decimales en formato de punto fijo es el formato Q que, en realidad, es una versión escalada de un número entero.

Tabla 1. Ejemplos de valores en formato Q

Valor decimal	Equivalente en binario (8 bits)	Equivalente en decimal (entero)
0.25	01000000	64
0.5	10000000	128
0.75	11000000	192
0.00390625	00000001	1

Por ejemplo, la operación 128*0.5 = 64 puede ser escrita en formato entero (punto fijo) como 128*128 = 100000000000000b = 16384. Finalmente, la respuesta adecuada es el resultado dividido por 2 elevado a la cantidad de bits utilizados, en este caso 8. De aquí tenemos que 16384/256 = 64. Nótese que matemáticamente lo que se está realizando es:

$$\frac{128 * \lfloor (256) * 0.5 \rfloor}{256}$$

para el caso de 8 bits ó 256. El valor decimal de 0.5 es expresado como un número entero y, finalmente, la respuesta es reescalada. La utilización de este formato en la definición de los coeficientes de los filtros, el valor equivalente de una función –como el seno o el coseno–cualquier cálculo puede evitar una disminución del tiempo de procesamiento al incorporar notación flotante en un procesador de punto fijo.

4. Algoritmos para la transformación al dominio de la frecuencia

En lo que concierne al análisis en el dominio de la frecuencia, la transformada de Fourier (TF) representa el método más utilizado. La transformada discreta de Fourier (DFT) se define en (1).

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \qquad (1)$$

de donde $W_N = e^{-j\left(\frac{2\pi}{N}\right)}$.

Nótese que la DFT se diferencia de la TF en el sentido de que la primera tiene una salida discreta, mientras que la segunda es continua. La teoría de la DFT puede ser consultada en cualquier texto de tratamiento de señales en tiempo discreto [1]. Algunos aspectos para recordar al utilizar la DFT son:

1. El teorema de Nyquist indica que la velocidad de muestreo de un sistema f_s debe ser por lo menos dos veces mayor a la máxima frecuencia que se desea analizar.

2. Es **necesario** utilizar un filtro *anti-aliasing* siempre que se utilice un conversor analógico digital, sin importar que la variable medida se crea esté debajo de la mitad de la frecuencia de muestreo

Las dos variantes más utilizadas en sistemas embebidos, gracias a su sencilla implementación, son el algoritmo de Goertzel y la FFT. La primera es la reestructuración de la DFT en forma de un filtro IIR de segundo orden, facilitando su cálculo debido al uso de valores reales solamente (la DFT trabaja con valores complejos). Además –esto es muy importante en el caso de los sistemas embebidos– ésta utiliza muy pocas posiciones de memoria para cada coeficiente de frecuencia que se desea calcular. El algoritmo es resumido en dos ecuaciones (2) y (3) y genera el mismo resultado que la DFT, después de *N* muestras. El único inconveniente es la no linealidad de su fase (debido a que es un IIR), que para ciertas aplicaciones es crítico.

$$s_k[n] = x[n] + 2\cos\left(\frac{2\pi k}{N}\right) s_k[n-1] - s_k[n-2] \qquad (2)$$

$$y_k[n] = s_k[n] - W_n^k s_k[n-1] \qquad (3)$$

de donde $X(k) = y_k[n]_{n=N}$ y los valores iniciales son 0.

La otra variante de la DFT es el algoritmo de la transformada rápida de Fourier (FFT). Éste es un algoritmo popular que calcula exactamente el valor de la DFT siempre y cuando se tenga en cuenta que la duración *N* de la señal sea una potencia de 2. Esto no suele representar mayor problema en los sistemas embebidos, ya que coincidencialmente las capacidades de memoria de estos suelen ser potencias de 2, p.e.: 128 bytes, 256, 512, 1024, etc.

Variantes de la misma FFT, muchas veces ignoradas, son las versiones de radix múlti-

Uno de los principales problemas de implementar algoritmos de procesamiento de señales es la alta cantidad de cálculos necesarios. Cuando esto se suma a la utilización de sistemas embebidos de punto fijo, el desempeño de éste puede verse disminuido notablemente.

ples. Éstas poseen un incremento en la velocidad con respecto a la FFT tradicional (radix-2). La única restricción necesaria para utilizar estas variantes es que el valor de *N* ahora no es una potencia de 2, sino de 4 (radix-4), o del radix elegido. Finalmente, a diferencia de Goertzel y de la DFT, la FFT entrega el espectro completo y no puede calcular valores individuales de frecuencia.

A la hora de comparar estos métodos se debe tener en cuenta dos cosas: primero, la cantidad de muestras que se desean obtener; si se requiere del espectro completo, la FFT, utilizando $N \log_2 N$ suele ser el algoritmo preferido, al ser precisamente el más rápido. Segundo, si se requieren pocas muestras o un menor uso de memoria el algoritmo de Goertzel suele ser la mejor opción, a menos que se requiera de fase lineal, lo que para una cantidad de dos o tres o dependiendo de la cantidad de puntos, es mejor utilizar el mismo algoritmo de la DFT pero para un solo valor de frecuencia. La duración de la DFT es de N^2.

Finalmente, es necesario recordar que no sólo existe la transformada de Fourier para realizar análisis en dominios transformados (incluyendo el de frecuencia). La transformada discreta del coseno (DCT) y dos de sus cuatro variantes son utilizadas en el campo del procesamiento de imágenes. También existe la transformada seno, las *wavelets* y variantes de la DFT para no entregar el espectro sino el *cepstrum* que es una versión logarítmica del primero. Este último método suele ser utilizado por ciertos algoritmos de reconocimiento de voz basados en el hecho de que el oído humano percibe las diferencias entre frecuencias de una forma no-lineal.

5. Consideraciones en el uso de la DFT y la FFT

La implementación de la DFT en un sistema embebido requiere de ciertos cuidados. El principal de todos radica en establecer qué resolución de frecuencia es requerida por el problema. Entre más fina sea ésta, el sistema podrá discriminar entre dos frecuencias cercanas. Para la DFT, la resolución Δf viene dada por (4).

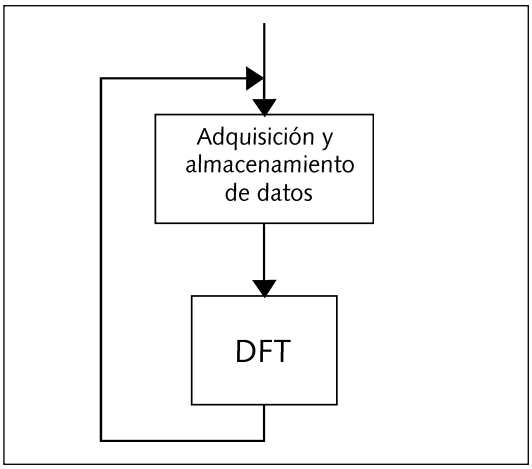
$$\Delta f = \frac{f_s}{N} \qquad (4)$$

Suponiendo que no se puede disponer de la frecuencia de muestreo, el único camino es aumentar *N*. Se tiene de (1) que la DFT (y la FFT también) entrega números complejos. Además, entrega la misma cantidad de muestras de salida que las utilizadas a la entrada. En otras palabras, debemos garantizar *2N* posiciones de memoria disponibles para almacenar tanto la parte real como la parte compleja. Asimismo, para $N > 16$ un procesador de 8 bits de resolución probablemente tenderá a desbordarse, además de tener errores de cálculo por redondeo o truncamiento muy altos, lo que implica que deberá trabajar por lo menos con 16 bits. Esto último implica la necesidad de tener *4N* posiciones de memoria disponibles para almacenar los resultados; en microcontroladores, con muy poca capacidad de memoria, lo anterior ha ser negativamente concluyente.

En lo que concierne al tiempo de cálculo, algo que suele ser olvidado es que para el caso de la FFT, el tiempo de cálculo de la mariposa (*butterfly*) es considerable. Una alternativa de esto es almacenar en un vector los índices vectoriales y realizar la mariposa (o diezmado en el tiempo) en el momento de la adquisición de los datos. En

La implementación de la DFT en un sistema embebido requiere de ciertos cuidados. El principal de todos radica en establecer qué resolución de frecuencia es requerida por el problema. Entre más fina sea ésta, el sistema podrá discriminar entre dos frecuencias cercanas.

Diagrama. 1 Flujo del proceso del cálculo del espectro

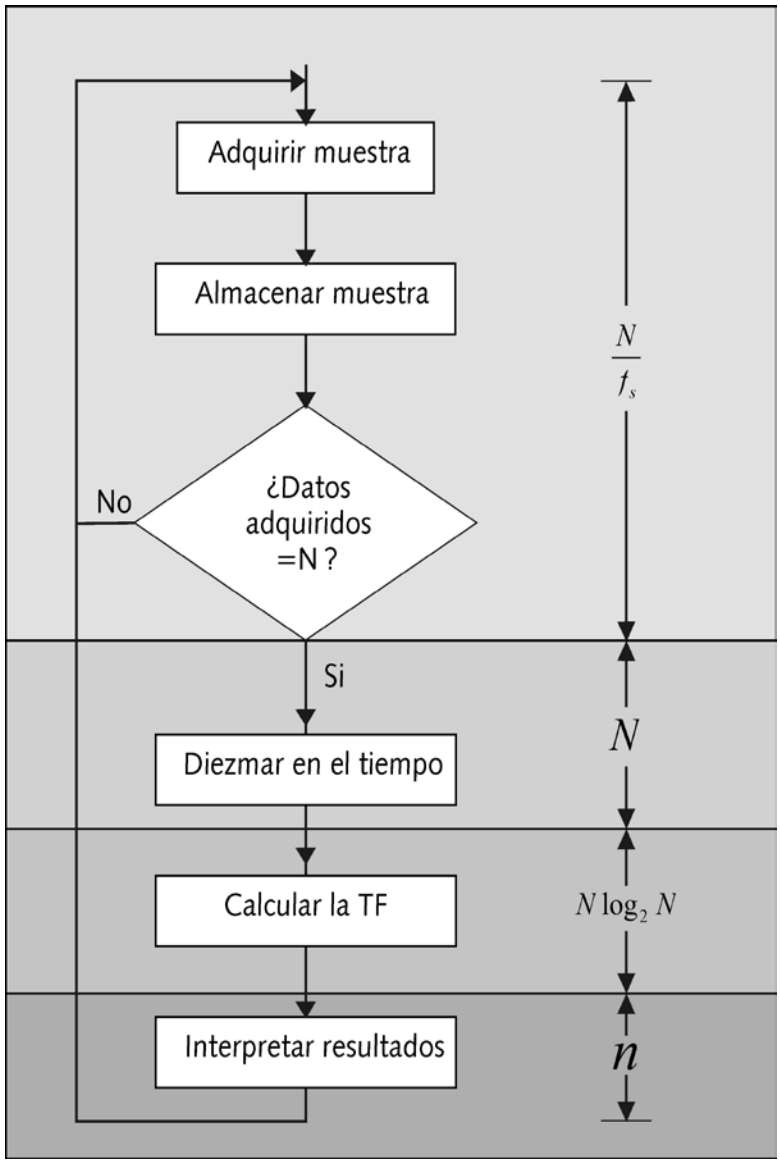


general, el cálculo del espectro se realiza en dos procesos. Ver el diagrama 1.

En el primer proceso se realiza la adquisición y almacenamiento de los datos. Una vez se obtienen *N* datos se procede a realizar la DFT. Finalizada, esta etapa se comienza nuevamente a adquirir los datos. En el siguiente diagrama se observa un diagrama de flujo un poco más detallado de los procesos necesarios. La optimización radica en que el proceso de adquisición de datos siempre tendrá un periodo fijo y la idea es **trasladar la mayor cantidad de procesos posibles a este bloque**.

En cuanto al proceso de diezmando asociado a la FFT, éste puede ser realizado en el tiempo o en frecuencia. Si se realiza en el tiempo puede ser incluido en el bloque de adquisición. Sin embargo, se puede acotar que si los resultados de la FFT son analizados en forma de vectores o posiciones de los vectores, el proceso de diezmado de cualquier tipo puede ser obviado. Por ejemplo, si se calcula la FFT y se analizan las muestras 4 y 5, equivalentes a $4\Delta f$ y $5\Delta f$, es posible analizar las muestras equivalentes sin el diezmado, de la misma forma, las muestras 17 y 23. En general, siempre que se necesiten detectar componentes fijos de frecuencia, como en aplicaciones de decodificación de tonos, el proceso de diezmado puede obviarse.

Diagrama. 2 Flujo del proceso del cálculo del espectro



6. La transformada de Fourier de corto-tiempo

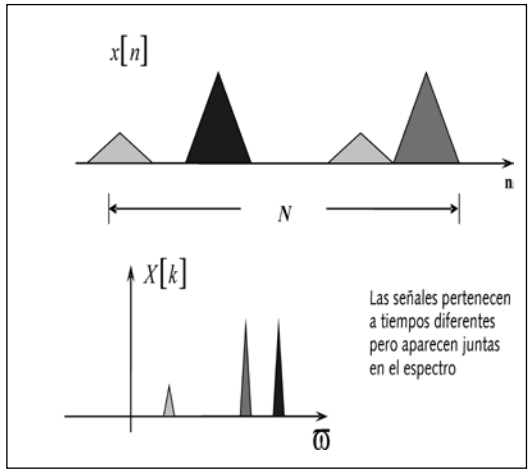
La transformada de Fourier de corto-tiempo (STFT) afronta ciertos problemas cuando utiliza la DFT o sus variantes para obtener el cálculo. En resumen, estos son:

1. Necesidad de una alta cantidad de muestras para obtener una resolución final.
2. Las señales muestreadas cambian muy rápidamente.

El primer problema proviene de (4). En esta ecuación se observa que Δf , la resolución espectral de la DFT depende de la velocidad de muestreo y de la cantidad de muestras. El problema de disminuir f_s radica en que se limita el rango de frecuencias que pueden ser detectadas por el sistema, ya que f_s debe ser por lo menos dos veces mayor a la máxima frecuencia deseada. La segunda opción es el incremento de *N*, con la implicación de que se necesita mayor espacio para el almacenamiento, mayor tiempo para el cálculo y mayor cantidad de muestras de la señal a examinar.

El segundo problema se puede observar en la gráfica 1. Aquí se tiene señales que cambian muy rápidamente en el tiempo. La TF está concebida para trabajar con señales periódicas y, aunque se puede tomar la DFT de un tiempo pequeño para concordar con la duración de las señales rápidas, la resolución de la señal puede verse afectada.

Gráfica 1. Señales combinadas en el espectro

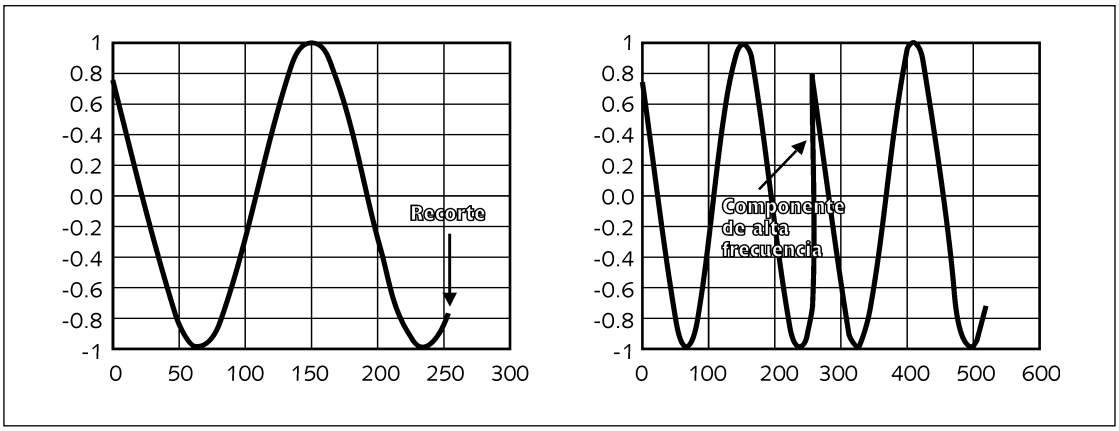


La solución sencilla a este problema consiste en utilizar la STFT. Este algoritmo calcula la TF de una señal de corto tiempo, pero conserva parte de la resolución de la TF tradicional (calcula la DFT durante el tiempo de existencia de la señal de interés). El truco consiste en realizar un *padding* o llenado con 0s. En resumen, se procede a adquirir un número de muestras adecuado dependiendo de la duración de las señales que se pretenden analizar y, posteriormente, se le agregan 0s, de modo tal que la formula (4) cumpla un requerimiento mínimo de resolución. Es importante ajustar el sistema para realizar un análisis tres veces en adelante a una misma señal, con el fin de evitar problemas debido al solapamiento de señales continúas en el tiempo y poder tener certeza de la presencia de la señal buscada.

Además del *padding*, el algoritmo de la STFT también realiza un proceso de enventanado para evitar que el recorte de los bordes se refleje como componentes de alta frecuencia ficticios en la señal transformada (ver gráfica 2). El propósito de la ventana es suavizar el recorte en los bordes para evitar el efecto antes mencionado. Es pertinente que se note que el proceso de enventanado se realiza en el tiempo y puede ser efectuado en el bloque de adquisición de datos, no afectando el tiempo total de procesamiento.

Ejemplos de ventanas utilizadas son *hamming*, *gaussiana* o

Gráfica 2. a) Recorte de la señal
b) Reconstrucción de la señal según Fourier



Gráfica 3. Versión enventanada utilizando hamming

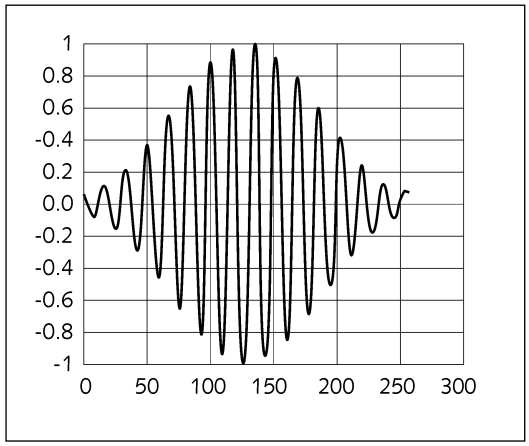
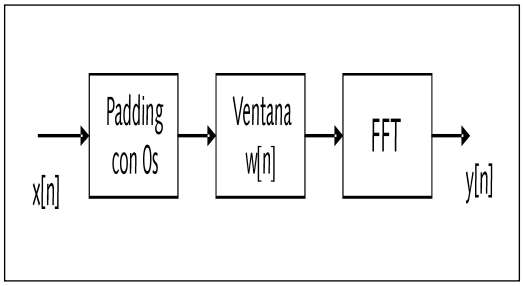


Diagrama 3. Diagrama de bloques de la STFT



filtros pasa bajos, entre otras. Finalmente, el algoritmo de la STFT se ve en la diagrama 3. Aquí se ha utilizado la FFT para la implementación del cálculo. Al final cualquier método o variante puede ser incorporado, teniendo en cuenta las restricciones propias de cada uno.

7. El teorema de Parseval

El teorema de Parseval relaciona la energía de la señal en el tiempo con la energía de su transformada en frecuencia. Aunque este no se utiliza directamente para transformar la señal del tiempo a la frecuencia, sí se puede utilizar para analizarla.

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X(k)|^2 \qquad (5)$$

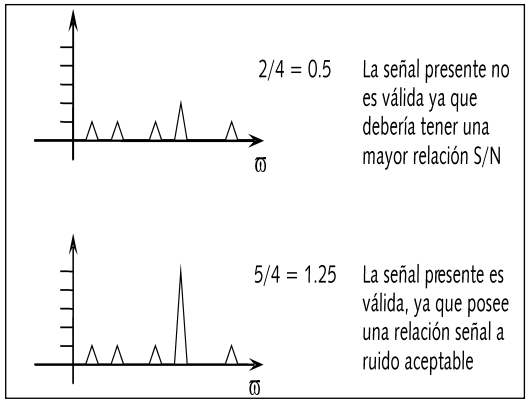
La utilidad del teorema de Parseval reside en que se puede calcular la energía total de la señal inclusive antes de realizar su transformada de Fourier. Se observa que si se calcula el cuadrado de las muestras instantáneas en el tiempo y se acumulan, al final se tendrá una versión escalada de la energía que tendrá toda la transformada

de Fourier. Este proceso puede ser realizado durante el proceso de adquisición de datos, análogo a lo que se indica en diagrama 2. Esto implica que el cálculo se puede realizar durante el periodo de adquisición de datos, no influyendo en el tiempo de procesamiento total.

Al final se podrá aproximar la relación señal/ruido de cada componente de frecuencia, ya que se conoce la energía de las componentes calculadas y la energía total de la señal (ver gráfica 1). Aunque la determinación de la energía se puede realizar después de calculada la FFT, la ventaja de utilizar el teorema de Parseval es que una vez se termine el cálculo de la FFT se puede proceder a estimar directamente la S/N. La alternativa sería implementar un bucle de *N* posiciones para sumar la energía de la señal transformada, pero esto obviamente afectaría el tiempo de procesamiento total.

Una aplicación de este procedimiento es la decodificación de tonos, ya sean DTMF o los utilizados por los fax (1200Hz y 2200Hz). Si se combina este método con la STFT u otra variante de análisis de Fourier basada en bancos de filtros, se puede implementar un decodificador de tonos de bajo costo. Como observación final se debe anotar que si se utiliza la STFT, la energía calculada en el tiempo deberá ser la de $x[n]w[n-m]$ en la que $w[n]$ es la función de ventana.

Gráfica 4. Relaciones de S/N



8. Conclusiones

Se discutieron varias técnicas y consideraciones para implementar algoritmos de procesamiento basado en el dominio transformado. La incorporación de la notación Q y el uso de la FFT o la STFT con las consideraciones sugeridas pueden disminuir el desempeño de muchos algoritmos hasta el punto de que no sea necesario la utilización de un procesador digital de señales, sino el uso de un microcontrolador trabajando entre 4MIPS y 8MIPS.

Referencias bibliográficas

[1] Oppenheim. A.V. (2000). *Tratamiento de señales en tiempo discreto*. S.d.: 2da Edición, Ed. Prentice Hall. Bogotá.

[2] Mariño. (1999). *Tratamiento digital de la señal*. S.d.: 2da Edición, .Ed. Alfa y Omega. Bogotá.

[3] Proakis, Manolakis. (1998). *Tratamiento digital de señales: principios algoritmos y aplicaciones*. S.d.: Ed. Prentive Hall. Madrid.

[4] Cooley, J. W. (1967) et al. *Historical Notes on the FFT*. IEEE Trans. Audi an electroacustics, vol Au-15.

[5] Winograd, S. (1978). *On Computing, the discrete Fourier Transform*, Math. Comp., Vol. 32.

[6] Bergland, G.D. (1969). *AGuided Tour of the FFT, IEEE Spectrum*, Vol. 6.