Cambio dinámico de frecuencia en señales de voz utilizando la tarjeta TMS320C67 DSK

Juan Fernando Navas Mantilla, ju-navas@uniandes.edu.co

Resumen—Se presentan dos posibles implementaciones de cambio en frecuencia de una señal de voz; en la primera de ellas el usuario determina dinámicamente la compresión de la Transformada de Fourier (FFT) de la señal de voz de entrada deseada. En la segunda la compresión o expansión de la Transformada de Fourier es determinada por una señal sinusoidal la cual puede ser modificada en amplitud y periodo por el usuario. Estos programas son montados en la tarjeta de desarrollo TMS320C67 de Texas Instruments.

Índice de Términos—DSP, FFT, Compresión, Expansión.

I. INTRODUCCIÓN

Los DSP's (Digital Signal Processors) son microprocesadores que son diseñados específicamente para el desarrollo de tareas en tiempo real; es por esta razón que su arquitectura es bastante disímil a la presentada por los microprocesadores mas comúnmente conocidos, aquellos que son utilizados en los computadores y que están diseñados hacia la realización de gran cantidad de tareas simultáneamente, es decir, al manejo de un sistema operativo.

Los DSP's nacen a partir de la necesidad de desarrollar aplicaciones en las que el tiempo de procesamiento era un factor determinante en el funcionamiento de la misma: aplicaciones de audio, comunicaciones alámbricas e inalámbricas, radares, instrumentación, etc. Pero al mismo tiempo que la velocidad de procesamiento (numero de instrucciones por segundo, numero de ciclos de reloj por instrucción) aumentaba considerablemente, para asegurar el éxito de los DSP's era necesario generar código con un rendimiento óptimo, es decir, que con el menor número de instrucciones lograra su objetivo.

Un impulso significativo al desarrollo de estos microprocesadores, y lo que en realidad generó esta gran revolución de la que hoy somos actores y observadores, fue la aparición del algoritmo de la FFT, Fast Fourier Transform, o Transformada Rápida de Fourier, que permitió tener acceso en tiempo real al espectro en frecuencia de cualquier señal, una habilidad que permitió gran cantidad de desarrollos posteriores (reconocimiento de patrones, análisis espectral, análisis de correlaciones, filtrado lineal).

El uso masivo de los DSP's ha acercado estas herramientas al público en general y en particular a los estudiantes de pregrado de Electrónica alrededor del mundo: ya existe un gran surtido de tarjetas de desarrollo de "bajo costo" que permiten el desarrollo rápido de proyectos sobre estos microprocesadores, y muchas de ellas permiten la programación de los mismos en lenguajes de programación de mas alto nivel que el Assembler original en que se programaban los DSP's, como C y C++.

Durante el semestre 2003-II en la clase de DSP's en la Universidad de Los Andes se estuvieron trabajando gran cantidad de aspectos teóricos y prácticos de esta área del conocimiento de la Electrónica, y se planteó la realización de un proyecto final de dificultad moderada, en el que se montara sobre una tarjeta TMS320C67 DSK de Texas Instruments una aplicación de tema libre.

Este proyecto se enfoca hacia dos aplicaciones de efectos de audio, y mas específicamente a efectos de voz. En realidad la primera de las aplicaciones (en la que el usuario de la misma puede cambiar la frecuencia de la voz en tiempo real) fue generada como un primer paso hacia la implementación de la segunda, que en realidad es el proyecto en si: en ella el espectro de frecuencia de la señal de voz de entrada se va comprimiendo o expandiendo de acuerdo al comportamiento de una señal sinusoidal la cual puede ser modificada en frecuencia y/o amplitud por el usuario.

Para el desarrollo de estos proyectos se siguió una metodología cuidadosamente diseñada: primero se simuló el comportamiento del efecto en un programa de fácil programación pero que no ofrecía las posibilidades de tiempo real, como MATLAB. Luego se cambió el código al lenguaje C, que es el idioma en que se comunica el programa CodeComposer con la tarjeta de desarrollo, y se probó el funcionamiento del mismo. Algunos cambios fueron hechos sobre la marcha, debido a problemas inesperados que surgieron.

Se presentará primero una descripción del algoritmo creado y luego una revisión de los resultados obtenidos, así como posibles cambios al algoritmo.

II. ALGORITMO

A. *MacroAlgoritmo*

El macroalgoritmo puede ser dividido en dos: en el primero se describe lo que se trabaja con las 256 muestras mas recientes de la señal de entrada, el cual se muestra en la figura 1. A estas muestras se les aplica la FFT de 256 puntos (tanto el numero de muestras como la complejidad de la FFT podría ser aumentada a 512, pero se mantuvo así por simplicidad y por no aumentar el retraso de la señal de salida con respecto a la señal de entrada). Luego ese espectro en frecuencia es comprimido o expandido según lo indique la señal D(X), donde X representa el cambio que se esta realizando; mas adelante en este documento se explicará mejor lo que significa la compresión y expansión del espectro de la señal. Después de estos cambios en frecuencia se aplica la IFFT,

Transformada Inversa de Fourier, que lleva de nuevo la señal al dominio del tiempo.

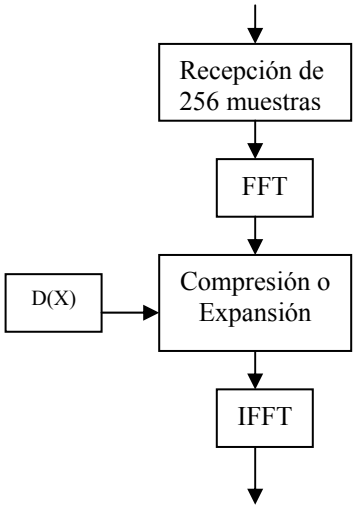

Fig. 1: Algoritmo 1

En el segundo macroalgoritmo se describe la entrada y salida de las muestras, y es mostrado en la Figura 2.

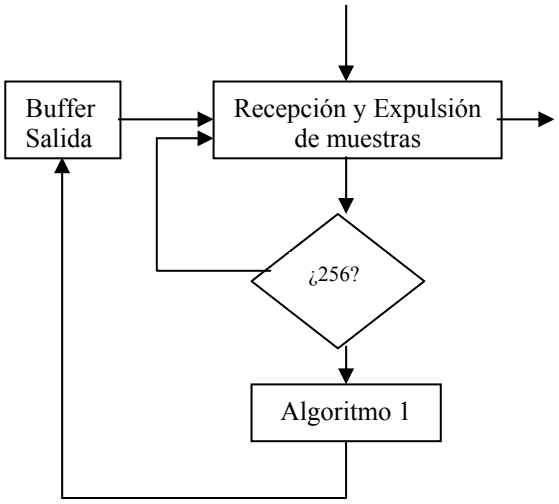

Fig 2: Algoritmo 2

A medida que las 256 muestras mas recientes son recolectadas, las últimas 256 que fueron procesadas por el Algoritmo 1 y que son guardadas en el Buffer de Salida son expulsadas. Eso indica un retraso global de 256 muestras, que a la velocidad de procesamiento de la tarjeta (8000 muestras por segundo) implican un retraso de 32ms, despreciable para efectos prácticos.

A continuación se irán detallando los aspectos más importantes de los algoritmos.

B. FFT e IFFT

Lo que hace el algoritmo de la FFT es hallar el comportamiento en frecuencia de las 256 muestras mas recientes, bajo el supuesto de que esta señal es periódica. Sin embargo, este espectro en frecuencia no es continuo, y el grado de precisión depende de la cantidad de puntos que tenga la FFT. En nuestro caso, se toman 256 puntos que representan cambios en frecuencia de 0 a 8kHz, lo que indica que la distancia entre dos puntos, esto es la *partición* de la FFT, será de 31.25Hz.

Ahora, para implementar la IFFT existen mecanismos que usan la propia FFT para hallar el resultado. En este proyecto se niega la parte imaginaria del espectro en frecuencia y el resultado se pone como entrada a ala FFT; esto produce la IFFT deseada.

C. Compresión y Expansión

La compresión y expansión son transformaciones que se hacen al espectro en frecuencia de la señal para cambiar de alguna forma sus características. Es importante señalar que aunque la FFT sea de 256 puntos, los 128 primeros son exactamente iguales a los 128 últimos transpuestos; esto porque se sabe que la máxima que se puede procesar a una tasa de muestreo de 8kHz e de 4kHz, es decir, la mitad exacta del espectro. Así pues, la compresión o expansión del espectro se hace con respecto al centro del espectro, que representa la frecuencia máxima permitida, como se muestra en las figuras 3a, 3b y 3c.

En la compresión, las componentes son llevadas hacia el centro, esto es, se mantiene la magnitud pero se aumenta la frecuencia, lo que hace que la señal de voz se oiga mas aguda. El grado de agudez es determinado por el número de fragmentos que se quiera comprimir el espectro. Las posiciones que quedan vacías (es decir, aquellas componentes de baja frecuencia) se rellenan con ceros.

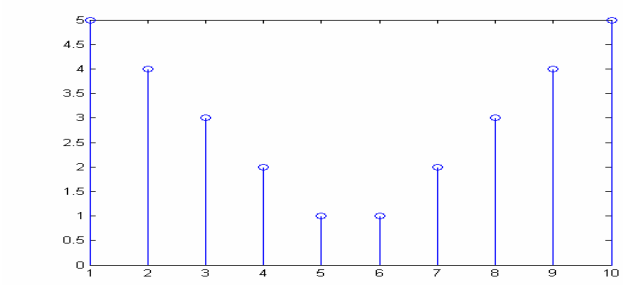

Fig 3a: el espectro original

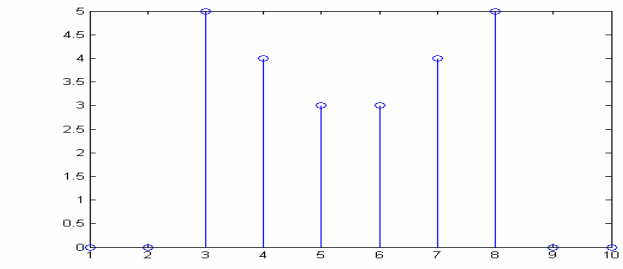

Fig 3b: el espectro comprimido en dos fragmentos

En la expansión la metodología es muy similar, salvo que ahora no se busca acercar el espectro al centro sino alejarlo de el, haciendo que la señal de voz se oiga mas grave. De nuevo,

las posiciones que quedan vacías (es decir, aquellas componentes de alta frecuencia) se rellenan con ceros.

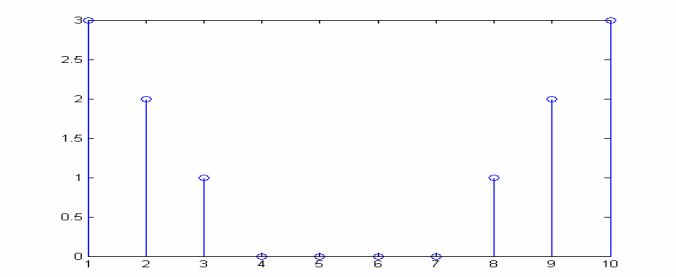

Fig 3c: el espectro expandido en dos fragmentos

D. Desviación en frecuencia

Cada 256 nuevas muestras se corre al algoritmo de la FFT, y al espectro resultante se le aplica compresión o expansión, después de lo cual se aplica la IFFT y se expulsan los datos resultantes. Pero si se quiere que la desviación en frecuencia siga un patrón en el tiempo, ¿Cómo debe ser programado el número de fragmentos en los que se va a comprimir o expandir el espectro?

En este proyecto se planteó un cambio en la frecuencia que siguiera el comportamiento de una señal sinusoidal. Para esto es necesario que el usuario determine:

- la amplitud de la señal, que determina el máximo grado de compresión o expansión de la señal de entrada, y
- el periodo de la señal, que determina cada cuanto se va a repetir el patrón de desviación.

Sin embargo, existen ciertas restricciones para el vector de desviaciones: debe ser de números enteros, pues representa el número de fragmentos en los que el espectro será comprimido o expandido. La única forma de lograr esto sin perdida de información sería exigirla al usuario desviaciones en frecuencia múltiplos enteros de 31.25Hz. Como este no es el caso, el vector de desviaciones debe ser "redondeado", es decir, llevado al entero mas cercano.

Por otro lado, la longitud del vector de desviaciones es igual al numero de veces que se comprime o expande el espectro, y ese número multiplicado por la tasa a la que salen 256 nuevas muestras (8000/256) debe ser igual al periodo de la señal sinusoidal suministrado por el usuario. Como se desea que la longitud del vector de desviaciones sea un entero positivo, también se hace un "redondeo" del periodo de la señal sinusoidal, llevándolo al valor permitido mas cercano.

En la figura se muestra una señal sinusoidal con periodo T = 2s y "amplitud" 200Hz, que es redondeada para que sus valores sean permitidos. Vemos que la longitud del vector que representa esta señal es 63, es decir, 2.016s; mientras que el valor máximo es de 6, que representa una desviación máxima de 187.5Hz.

E. Simulaciones

Se simuló el comportamiento del sistema en MATLAB, con resultados satisfactorios. En el código (que se debe encontrar en el mismo directorio de este documento) *efeefetes.m* se toma un archivo de formato WAV (*voice.wav*) y se transforma en uno llamado *out.wav*. Pueden apreciarse los cambios en frecuencia realizados a la señal de voz.

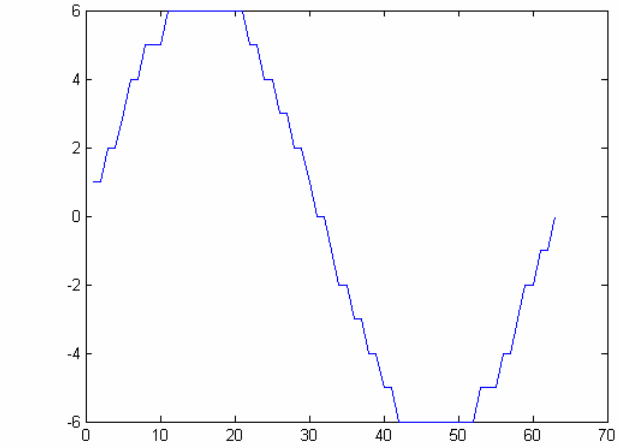

Fig 4: Señal sinusoidal de "amplitud" 200Hz y periodo 2s, que determina como van a ser las desviaciones cíclicas en frecuencia del espectro de la señal de voz

III. IMPLEMENTACIÓN EN LA TARJETA DE DESARROLLO TMS329C67 DSK

Convertir el código escrito en MATLAB a código C fue una tarea relativamente sencilla, pues se aprovecho el hecho de que el código en MATLAB fue escrito pensando en una futura implementación en tiempo real. Sin embargo, para mejorar el desempeño de la aplicaron fue necesario implementar algunos cambios:

- Se introdujo un filtro pasabajos son frecuencia de corte Fc = 1700Hz, para eliminar un ruido de alta frecuencia que se estaba generando; esto es equivalente a eliminar muchas de las componentes en alta frecuencia del espectro de la señal, lo cual no es critico dada la naturaleza de baja frecuencia presente en la voz humana. Los coeficientes del filtro fueron precalculados usando SPTOOLS de MATLAB, y puestos en el programa como un vector incluido en el encabezado del programa.
- Se eliminó la componente DC del espectro en frecuencia transformado, pues había un nivel DC indeseado a la salida del sistema.
- Se atenuó la señal de salida, para hacerla compatible con nuestros sistemas de medición y reproducción.

La salida de la tarjeta no tiene gran potencia ni volumen, por lo que fue llevada a la tarjeta de audio de un computador, la cual la amplificó sustancialmente, aunque le introdujo algún ruido indeseado.

A pesar de todos estos cambios hubo un aspecto que no se pudo solucionar: un ruido fuerte de baja frecuencia presenta en la salida del sistema. Dada la baja frecuencia del mismo (unos pocos Hz) no fue posible implementar un filtro pasa altos; se cree que la razón de la existencia de este ruido es el empaquetamiento de las muestras para ser transformadas y luego antitransformadas; es posible que con un enventanamiento o una transposición de las señales de salida se elimine el ruido.

La forma como el usuario tenia la posibilidad de cambiar la desviación en frecuencia (en el primer programa) y la máxima desviación en frecuencia y el periodo de la misma (en el segundo programa) fue implementada usando archivos .GEL, disponibles en el CodeComposer, que permiten manipular variables creadas en el entorno de programación de C. Cuando estos archivos *gel* son llamados, crean figuras que son *sliders* que al ser manipulados cambian el valor de estas variables en tiempo real.

En general, los resultados obtenidos en tiempo real fueron satisfactorios y muy similares a los obtenidos en las simulaciones. Se produjo un cambio bastante interesante y curioso en el *pitch* (frecuencia) de la señal de entrada.

IV. CONCLUSIONES

Se mostró que es posible implementar en un tiempo corto aplicaciones de efectos de voz en tiempo real usando la tarjeta de desarrollo de Texas Instruments, la cual demostró su versatilidad de manejo y su facilidad en programación, así como su comportamiento voluble. Se mostró también como un cambio mínimo en el espectro de una señal de entrada produce cambios bastante apreciables en la señal de salida, lo cual abre las puertas para nuevos experimentos

REFERENCIAS

[1] J.G Proakis, D.G Manolakis. “Tratamiento Digital de Señales: Principios, algoritmos y aplicaciones”. Ed Prentice Hall. 3ª Edition. 1999.

[2] R. Chassaing. “DSP Applications using C and the TMS320C6x DSK”. Ed John Wiley & Sons, Inc. 2002.