Codificador de Voz Multipulso en Tiempo Real según Recomendación GSM 6.10

Carles Antón Haro y José Adrián Rodríguez Fonollosa

Dpto. de Teoría de la Señal y Comunicaciones. Universitat Politècnica de Catalunya.
ETSE Telecomunicació, Apdo. 30002, Barcelona 08071
Tel: 93-4016439, Fax: 93-4016447, e-mail: adrian@tsc.upc.es

Abstract.- In this paper we explain a real-time implementation process for the source coder described in recommendation GSM 6.10 applicable to the full-rate voice channels of the pan-European Digital Mobile Radio system. We chose for the implementation the TI's TMS320C30 which provides floating-point operations and 16.7 MIPS. Major techniques to reduce computational load as well as different implementation stages are fully described.

1. INTRODUCCIÓN

En la recomendación GSM 6.10 se describe el procedimiento de transcodificación aplicable a los canales de voz full-rate del Sistema Paneuropeo de Comunicaciones Móviles Celulares. La entrada del sistema la constituye una señal digital de voz con codificación PCM lineal (13 bits/muestra), realizándose el procesado de ésta en tramas de 160 muestras. A la salida del sistema, cada trama es representada con un bloque de 260 bits que, a una frecuencia de muestreo de 8 kHz, suponen un bit-rate de 13 kb/s [1,2,3]. En referencia al sistema de codificación convencional (log-PCM, 64 kb/s), dicho codificador permite un ahorro del 80% del flujo de bits.

En las siguientes líneas, y una vez expuesto con brevedad el algoritmo empleado, se explicarán las técnicas utilizadas para reducir la carga computacional de algunas de las rutinas, implementadas en un primer momento en C. La optimización en cuanto a tiempo de ejecución se realizó tanto en lenguaje de alto nivel como en ensamblador, siendo la mayoría de las técnicas aplicables como tales, a un gran número de máquinas y de algoritmos de procesado de la señal.

2. DESCRIPCIÓN DEL CODIFICADOR

En la Fig. 1 podemos observar el diagrama de bloques simplificado del codificador multipulso RPE-LTP (Regular Pulse Excitation, Long Term Prediction) [1]. La trama de señal de voz es procesada en primer lugar con el objeto de eliminar cualquier posible offset (filtro paso-alto de primer orden). A continuación se la somete a un filtrado FIR de preénfasis mediante el cual se enfatizaran las componentes de alta frecuencia y baja energía. Con la señal obtenida a la salida, se realiza un análisis LPC (Linear Predictive Coding), que conlleva el cálculo de los 8 primeros valores de la autocorrelación de la señal. Este análisis nos permitirá determinar los coeficientes del filtro variante short-term (filtro FIR, de orden 8 y estructura lattice) con el que será procesada la señal a continuación. Los parámetros de este filtro -los coeficientes de reflexión-, se transforman en LAR (Log Area Ratios) antes de proceder a su transmisión.

Para las siguientes etapas es necesario dividir las tramas de señal residual de la etapa short-term en cuatro subtramas de cuarenta muestras. En consecuencia las operaciones descritas a continuación se llevaran a cabo cuatro veces por trama, hecho a tener en cuenta en el momento de calcular la carga computacional de cada rutina.

Así pues, en el bloque de análisis LTP se obtienen los parámetros que se utilizaran en el siguiente filtrado: el período de pitch y la ganancia LTP. Dichos parámetros se calculan a partir de las cuarenta muestras de la subtrama actual y de una secuencia, almacenada en memoria, consistente en una versión reconstruida de las 120 muestras precedentes. A continuación se realiza la diferencia entre la señal residual del filtro short-term y una predicción de ésta obtenida a la salida del filtro de análisis long-term (o de predicción a largo plazo). Los dos parámetros mencionados con anterioridad se calculan de tal manera que se minimice el error cuadrático medio de predicción.

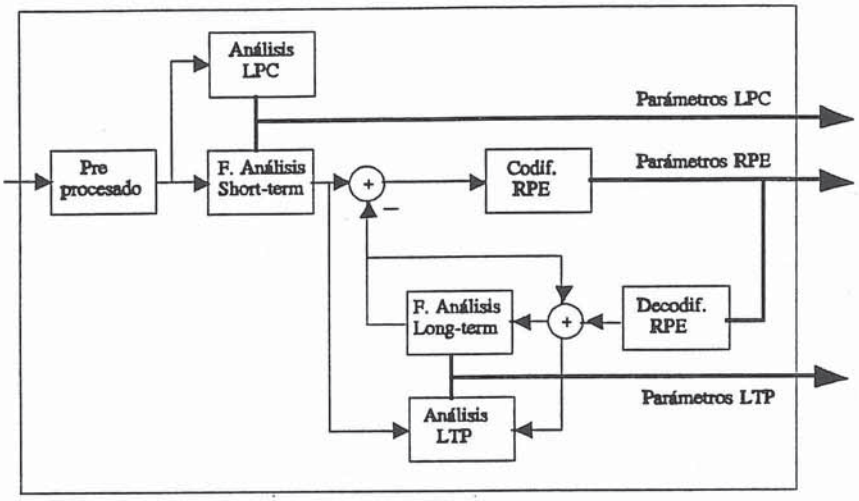

Fig. 1: Diagrama de bloques del codificador.

Por último, en el bloque de análisis RPE se lleva a cabo la función básica de compresión del algoritmo: cada subtrama de 40 muestras es filtrada paso bajo (Filtro de Ponderación FIR, 11 coeficientes), y diezmada adaptativamente por un factor de 3, quedándonos con aquella subsecuencia de 13 pulsos equiespaciados que presente una mayor energía. Finalmente, los 13 pulsos seleccionados son codificados mediante codificación APCM.

Todos los parámetros generados en la etapa RPE constituyen la entrada de un decodificador local, a cuya salida se obtendrá una versión cuantificada de la señal residual de la etapa long-term. Mediante la adición de la subtrama de estimaciones de la señal residual short-term se obtiene la versión reconstruida de ésta, cerrándose de esta manera el lazo de realimentación.

Por lo que hace referencia al decodificador (Fig. 2), diremos que se corresponde casi totalmente con el lazo de realimentación del codificador, con la salvedad que los filtros empleados son los inversos a los que se emplearon allí.

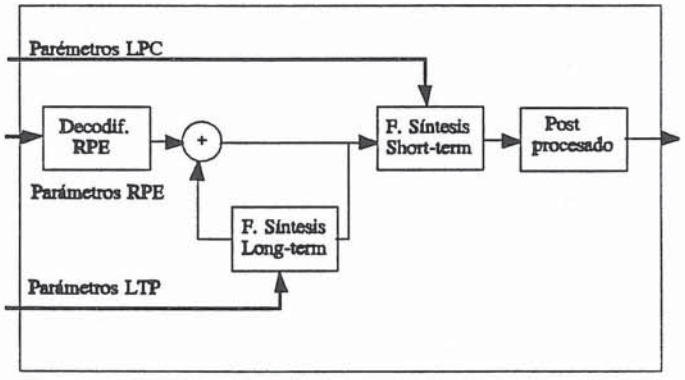

Fig. 2: Diagrama de bloques del decodificador.

3. REQUISITOS DE LA IMPLEMENTACIÓN

A parte de conseguir que la señal de voz sintetizada tenga una calidad aceptable y próxima a la calidad telefónica (hecho que viene garantizado por el hecho de trabajar sobre norma), la implementación del codificador debe cumplir varios requisitos.

Por una parte debe ser posible que el funcionamiento de éste se dé en tiempo real; en consecuencia, y debido a que el procesado de la señal se hace en base a tramas de 160 muestras (20 ms), es necesario que cada trama sea procesada en un tiempo igual o inferior a éste.

Por otra parte se observa que el modo de funcionamiento del sistema es claramente full-duplex. En la medida que es deseable emplear un solo microprocesador por extremo, éste deberá realizar en un tiempo inferior a la duración de una trama ambas operaciones: codificación de la trama generada y decodificación de la recibida [7].

4. ETAPAS DE IMPLEMENTACIÓN

En un primer momento se llevó a cabo un simulación previa del codificador -en lenguaje C- sobre un PC, en el cual se intentó generar un código lo más rápido posible (operaciones en coma flotante, uso de variables register, etc). En esta primera etapa se comprobó el buen funcionamiento del algoritmo mediante medidas de la SNR de la señal sintetizada y mediante audiciones de los ficheros de voz. Asimismo se obtuvo un conjunto de parámetros intermedios que facilitarían más tarde la depuración de las sucesivas versiones del código [7].

A continuación, y habiendo resultado plenamente satisfactorias las medidas de calidad efectuadas sobre la simulación, se trasladó el código a su plataforma definitiva: el DSP TMS320C30 de Texas. Sobre dicho procesador se realizaron medidas del tiempo de ejecución de las rutinas que contenían productos escalares y filtrados por ser éstas operaciones especialmente críticas en cuanto a tiempo de cálculo. A la luz de los resultados obtenidos sobre la versión del código que realizaba el máximo número de operaciones intermedias en coma flotante (Tabla I), se decidió emprender las siguientes acciones: las rutinas más farragosas (cálculo del pitch, filtros short-term, cálculo de autocorrelaciones y filtro de ponderación) serían substituidas por otras en ensamblador con el objeto de aprovechar todas las facilidades que proporciona el conjunto de instrucciones; las rutinas de carga computacional intermedia (rutinas de pre y postprocesado) serían mejoradas directamente en lenguaje de alto nivel. Se descartó la posibilidad de optimizar otras rutinas por considerarse poco rentable en cuanto a tiempo de proceso. Una vez conseguido el funcionamiento en tiempo real, se procedería a redireccionar la E/S del sistema sustituyendo los ficheros de voz por sendos conversores A/D y D/A.

5. TÉCNICAS DE REDUCCIÓN DEL TIEMPO DE EJECUCIÓN

En las rutinas optimizadas en lenguaje de alto nivel, las técnicas empleadas fueron las siguientes:

- Reducción del número de llamadas a rutinas (sobretodo dentro de los bucles) y uso exhaustivo de variables globales. De esta manera se evita trabajar en exceso con la pila y pasar un número elevado de parámetros a las funciones.
- Evitar en la medida de lo posible el uso de bucles, ya que ello supone gestionar variables de contador, realizar cierto número de saltos, etc. Es recomendable sustituir los bucles por código en línea generado, por ejemplo, con la directiva '#define'.
- Almacenar las variables de uso frecuente (variables de contador, punteros a arrays,

acumuladores, etc) en registros internos de la CPU cuyo tiempo de acceso es inferior al de la memoria RAM. En C esto se consigue mediante el uso del modificador 'register' [6].

Tabla I: Tiempos de ejecución de las rutinas más significativas antes de optimización. El porcentaje es respecto al tiempo disponible (duración de una trama).

Rutina	Nº ejec/trama	t_{total} [ms]	Porcentaje
Filtros short-term	4 cod+4 decod	8.30	41.50
Cálculo Pitch	4 cod	5.76	28.80
Filtro Ponderación	4 cod	2.10	10.50
Autocorrelaciones	1 cod	1.25	6.25
Selección del grid	4 cod+4 decod	0.80	4.00
Filtros preprocesado	1 cod	0.50	2.50
Filtro postprocesado	1 decod	0.50	2.50
Resto de rutinas		4.28	21.40
TOTAL		23.49	117.45

- Utilización de punteros a memoria para acceder a los arrays. El código generado de esta manera es más ligero que el generado con el método clásico (con indexación, s[n]) [6].
- Realizar el máximo número de cálculos intermedios (cálculo de correlaciones, filtrados, productos escalares, etc) en coma flotante, ya que en nuestro caso estas operaciones son más rápidas que las equivalentes en coma fija [4].
- Sustitución de los filtros conectados en cascada por el correspondiente filtro equivalente.

Obviamente las técnicas empleadas en las rutinas en ensamblador surgen del conocimiento profundo de la arquitectura del DSP. Sin embargo la mayoría de ellas son fácilmente transportables a otras máquinas. En concreto las técnicas más apropiadas en caso de utilizar el TMS320C30 son las siguientes:

- Uso de instrucciones que permiten realizar productos y acumulaciones en paralelo (MPYF||ADDF) para aligerar el cálculo de productos escalares y filtrados.
- Utilización de saltos retardados (BD Branch Delayed, DBD Decrement and Branch Delayed, BcondD Branch conditionally Delayed) en la ejecución de los bucles. De esta manera se evita romper el pipeline [4] y aprovechar algunos ciclos más de instrucción en cada iteración.
- Utilización de instrucciones que autogestionan el direccionamiento circular necesario en los filtrados [4], hecho que evita dedicar otras instrucciones a tal efecto.
- Uso de pre y postdecrementos de los punteros a memoria, consiguiéndose así, además de leer el valor apuntado, preparar el puntero para la próxima lectura.
- Recurrir a instrucciones del tipo RPTS (RePeaT Single instruction) y RPTB (RePeaT Block of instructions) para la implementación de bucles. Dichas instrucciones evitan realizar repetidos fetchs a las mismas instrucciones así como la ruptura del pipeline a cada iteración.

Ejemplos de utilización de estas técnicas aparecen en las referencias [4,5,7].

6. CONCLUSIONES

Los resultados obtenidos tras aplicar las técnicas anteriormente mencionadas aparecen en la Tabla II. En ella observamos que se ha conseguido reducir el tiempo de ejecución a un 54.75% del disponible, lo que nos permitiría destinar el tiempo restante a otras tareas o bien utilizar un DSP de prestaciones más modestas. Son especialmente significativas las reducciones conseguidas en las rutinas de cálculo del pitch y de filtrado short-term [7]. Por lo que hace referencia a la SNR, ésta permanece dentro de niveles aceptables para codificadores de estas características [7].

En el supuesto que fuese necesario reducir aún más el tiempo de ejecución, sería necesario ir implementando las rutinas restantes directamente en ensamblador.

Tabla II: Tiempos de ejecución de las rutinas más significativas tras la optimización. El porcentaje es respecto al tiempo disponible (duración de una trama).

Rutina	Nº ejec/trama	t_{total} [ms]	Porcentaje
Filtros short-term	4 cod+4 decod	3.04	15.20
Cálculo Pitch	4 cod	1.85	9.25
Filtro Ponderación	4 cod	0.70	3.50
Autocorrelaciones	1 cod	0.11	0.55
Selección del grid	4 cod+4 decod	0.80	4.00
Filtros preprocesado	1 cod	0.10	0.50
Filtro postprocesado	1 decod	0.07	0.35
Resto de rutinas		4.28	21.40
TOTAL		23.49	54.75

Referencias

[1] "GSM Full Rate Speech Transcoding. Recommendation GSM 6.10", GSM, Version 3.2.0 (July 1989).
[2] Raymond Steel, "Mobile Communication", London Pentech Corp,pags. 677-690 (1992).
[3] P. Kroon, R.J. Sluyter and E.F. Deprettere, "A Low Complexity Regular Pulse Coding Scheme with a Reduced Transmission Delay", Proc. ICASSP, vol.1, pags. 3083-3086 (1986).
[4] "TMS320C3x User's Guide", Texas Instruments, Digital Signal Processing Products (1991).
[5] "Digital Signal Processing Applications with the TMS320C30 Evaluation Module", Texas Instruments (1991).
[6] Herbert Schildt, "Programación en Turbo-C", Borland/Osborne/Mc Graw-Hill (1988).
[7] Carles Antón Haro, "Codificador de voz multipulso en tiempo real según recomendación GSM 6.10", Proyecto Final de Carrera, E.T.S.I. Telecomunicación de Barcelona, UPC, 1994.