Implementación sobre TMS320C31 del Codificador de Voz "Half Rate" descrito en la Recomendación GSM 6.20[†]

CARLES ANTÓN HARO, JAIME GONZÁLEZ GUIJARRO Y JOSÉ A. RODRÍGUEZ FONOLLOSA

Dpt. de Teoria del Senyal i Comunicacions - Universitat Politècnica de Catalunya
c/ Gran Capità s/n. Campus Nord UPC - 08034 Barcelona (SPAIN)
Correo electrónico: carles@gps.tsc.upc.es

Abstract:

In this paper, an implementation process for the GSM 6.20 "Half Rate" (HR) speech coder is reported. Such a coder, to be used in GSM's Phase 2, allows to allocate, within the same bandwidth, double the number of users than GSM's Phase 1 "Full-Rate" (FR) coder. The implementation was performed on and optimized (in terms of execution time) for the TI's TMS320C31 (50 Mflops, 25 MIPS). Both 'C' and assembler optimization techniques are presented and discussed. After optimization, execution time is reduced in a factor of 25.

1. Introducción

El codificador de voz "Half Rate" (HR) [1] fue seleccionado en su día por la ETSI (European Telecommunication Standards Institute) para ser utilizado en la segunda fase (Phase 2) de implantación del sistema de comunicaciones móviles GSM [2]. Dicho codificador genera un bit-rate neto de 5.6 kb/s frente a los 13 kb/s del codificador FR [3]. De esta manera, es posible multiplexar en el mismo ancho de banda el doble de usuarios que en la primera fase con el aumento de eficiencia espectral que ello supone. Esta reducción de la tasa de transmisión se consigue gracias a la utilización de un esquema de codificación VSELP (Vector-Sum Excited Linear Prediction) [4] el cual se enmarca dentro de la familia de codificadores CELP (Code Excited Linear Prediction) [5,6]. La calidad de la voz sintetizada permanece prácticamente inalterada pero la carga computacional es notablemente superior debido, básicamente, al uso de codebooks.

En esta comunicación, se describe el proceso de implementación del codificador de voz HR sobre un DSP TMS320C31 de Texas Instruments [7]. Dicho DSP dispone de aritmética de coma flotante y opera a una frecuencia de reloj de 50 MHz pudiendo ejecutar 25 MIPS y 50 MFLOPS. Dispone de 256 kpalabras de memoria RAM y 2 kpalabras de memoria dual-port.

En el siguiente apartado se lleva a cabo una breve descripción del funcionamiento del algoritmo de codificación.. A continuación, se describe el proceso de implementación sobre TMS así como las técnicas empleadas para reducir el tiempo de ejecución tanto en el ámbito de uso de lenguaje de alto nivel ('C') como de bajo nivel (Ensamblador). Por último, en el apartado 4 se exponen los resultados obtenidos.

2. Descripción del codificador

La entrada del sistema la constituye una señal digital de voz con codificación PCM lineal a 13 bits/muestra. A la salida del sistema, cada trama de 160 muestras (20 ms) viene representada por un bloque de 112 bits que, a una frecuencia de muestreo de 8 kHz suponen un bit-rate de 5.6 kb/s.

En la Fig.1 se puede observar el diagrama de bloques del codificador HR así como los parámetros obtenidos en cada una de las etapas. Es importante advertir que este diagrama es el correspondiente al algoritmo empleado para codificar tramas de voz sonoras. Caso de ser sordas, el esquema es ligeramente distinto y se detallará más adelante.

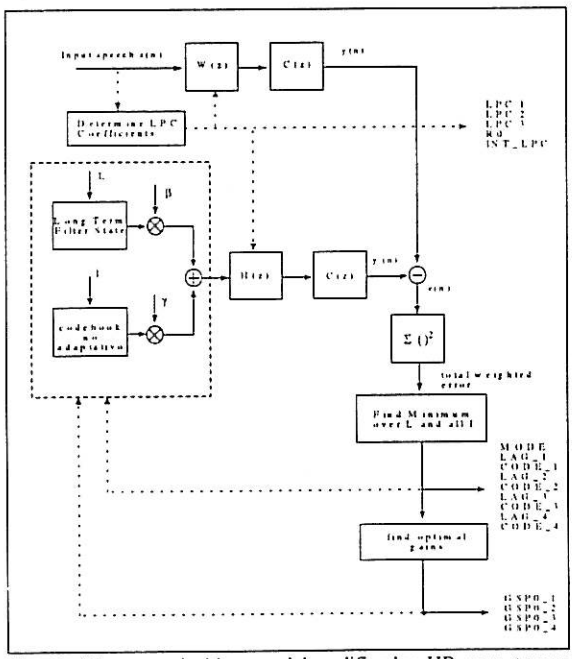

Fig. 1: Diagrama de bloques del codificador HR para tramas sonoras. Para tramas sordas, ambos codebooks son no adaptativos y desaparece el filtro de ponderación de ruido en los armónicos.

[†] Este trabajo ha sido financiado por la Comisión Interministerial de Ciencia y Tecnología (TIC95-1022-C05-03) y por el Comissionat per a les Universitats i Recerca (Generalitat de Catalunya).

2.1 Funcionamiento del codificador

En primer lugar, se realiza un filtrado paso alto de la trama de entrada con un filtro IIR de cuarto orden que elimina las componentes frecuenciales por debajo de los 120 Hz. A continuación, se realiza un análisis LPC (Linear Predictive Coding) de orden 10. A partir de dichos parámetros se obtiene, a su vez, el filtro W(z) cuya función es aprovechar el efecto enmascarador de los formantes permitiendo un error de síntesis mayor cerca de ellos. El filtro C(z), que tan sólo se emplea para trama sonoras, permite controlar la energía de e(n) en los armónicos de la frecuencia de pitch. Nótese que, ambos filtros ponderan espectralmente la potencia de la señal error, e(n), pero que, por simplicidad, han sido desplazados dentro de cada una de las ramas que confluyen en el restador. Asimismo, el filtro W(z) está incorporado en el filtro de síntesis short-term H(z). Los parámetros LPC (bajo la forma de coeficientes de reflexión) son cuantificados vectorialmente en tres grupos obteniéndose los índices LPC1..LPC3. El flag INT_LPC indica si para obtener una señal sintetizada de mayor calidad (mayor ganancia de predicción) es necesario o no interpolar los parámetros LPC de tramas consecutivas. La energía de la trama vendrá dada por el parámetro R0 que es cuantificado con 5 bits.

Para las siguientes etapas, se divide la trama en subtramas de 40 muestras. En consecuencia, el análisis descrito a continuación se realiza cuatro veces por trama hecho a tener en cuenta al calcular la carga computacional de cada rutina.

Mediante cálculo de la autocorrelación de la señal, se obtiene la ganancia de predicción LTP. Si ésta no sobrepasa un cierto umbral la trama es sorda (MODE=0) y no se prosigue con el cálculo del pitch. En caso contrario, el codificador HR utiliza una combinación de técnicas en lazo abierto y cerrado para calcular el periodo de pitch (posiblemente fraccionario) de la señal. La secuencia de valores de periodo de pitch para las cuatro subtramas ('trayectoria') se elige de tal manera que para los tres últimos se pueda aplicar una codificación delta con la que ahorrar algunos bits a la hora de cuantificar dichos valores. A partir de ellos se obtienen valores mas detallados de la ganancia de predicción que permitirán medir el mayor o menor grado de sonoridad de la trama (MODE=3..1).

El siguiente paso consiste en hallar los parámetros que caracterizan a la excitación obtenida como suma de dos excitaciones (ver Fig. 1). Caso de tratarse de una trama sonora (MODE=1..3), una de ellas es la salida del *Long Term Filter State* y la otra es la salida de un codebook VSELP no adaptativo de 512 elementos. En caso contrario, ambas son las salidas de sendos codebooks no adaptativos de 128 elementos cada uno. El filtro de predicción long-term cuyos *elementos* son las muestras precedentes de la excitación puede interpretarse como un codebook adaptativo para el que el periodo de pitch, previamente calculado, representaría el índice a dicho codebook. Así pues, en el caso de tramas sonoras, tan solo resta hallar el elemento del segundo codebook junto con los valores óptimos de las ganancias para cada codebook (β y γ) que minimicen el error (ponderado espectralmente) a la salida. En el caso de tramas sordas, se repetirá este proceso para cada uno de los dos codebooks VSELP.

Por último, β y γ son cuantificados vectorialmente y de manera distinta para cada uno de los diferentes modos de funcionamiento (MODE=0..3) dando lugar al conjunto de parámetros equivalentes GSP0_x (x indica el número de subtrama).

Nótese que el codificador HR es de *análisis por síntesis*. En efecto, para obtener los parámetros que representan cada trama de señal de voz, el codificador construye una réplica de la señal a sintetizar por el decodificador, y´(n), y la compara con la señal de entrada y(n). En consecuencia, el decodificador forma parte del codificador cuyo diagrama de bloques se presenta en la Fig. 2.

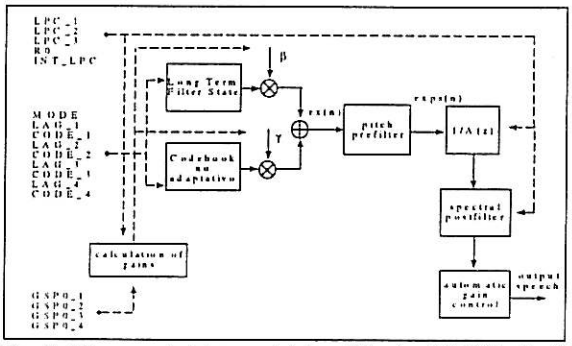

Fig. 2. Diagrama de bloques del decodificador "Half Rate". El elemento pitch prefilter únicamente está presente en el caso de sonidos sonoros.

2.2 Codebook VSELP

Los codebooks VSELP se caracterizan porque sus codevectors se obtienen por combinación lineal de unos pocos vectores básicos:

$$u_i(n) = \sum_{m=1}^{M} \theta_{im} * v_m(n)$$

donde $u_i(n)$ es el codevector i-ésimo del codebook (de longitud 40 muestras), v_m un vector de la base, y $\theta_i=[\theta_{i1}.., \theta_{iM}]$ con $\theta_{im} \in \{1,-1\}$ es la palabra código asociada al codevector. Tres son los ventajas de los codebooks VSELP. La primera es que para palabras código que sólo se diferencian en un bit, la señal error asociada al codevector de la segunda puede calcularse de manera recursiva a partir de la primera con el consiguiente ahorro de operaciones. Además, estos codebooks son robustos frente a errores de

signo en θ_{im} ya que la diferencia entre esos dos codevectors es relativamente pequeña. Por último, en la medida que la excitación es suma de dos codevectors cuyos índices óptimos se calculan de manera independiente, para cada subtrama sonora (p.e.) sería necesario ortogonalizar todo el segundo codebook respecto al codevector elegido del primero. En cambio, para un codebook VSELP basta con ortogonalizar los elementos de la base (M_1= M_2=7 para tramas sordas y M=9 para sonoras) siendo necesarias un número menor de operaciones.

Parámetros	Número de bits
Tipo de señal (MODE)	2
Nivel de energía (R0)	5
Coefs. LPC (LPC y INT_LPC)	29
Ganancias de excitación (γ y β)	20
Palabra código (CODE)	
Trama sonora/sorda	36 / 56
Lags (LAG)	
Trama sonora/sorda	20 / -
Total.....	**112**

Tabla I: Parámetros derivados del análisis de una trama.

3. Optimización del código

En un principio se disponía del código en 'C' suministrado por la ETSI. Previamente al inicio del proceso de optimización, se obtuvo un conjunto de parámetros intermedios que facilitarían más tarde a depuración de las sucesiva versiones del código. La reducción del tiempo de ejecución del código se llevó a cabo en dos etapas. En un primer momento se aplicaron técnicas de optimización en lenguaje de alto nivel 'C'. Una vez hecho esto, se procedió a optimizar las rutinas de mayor carga computacional directamente en lenguaje ensamblador del DSP.

3.1 Optimización en lenguaje de alto nivel

En esta primera etapa, se llevó a cabo una optimización general del código independientemente de la carga computacional de cada rutina en cuestión.

En tanto en cuanto el DSP empleado dispone de aritmética en coma flotante, el primer paso consistió en sustituir todas las operaciones en coma fija por las correspondientes en coma flotante. Así se eliminaba la necesidad de realizar continuas comprobaciones de saturación y escalados.

A continuación se procedió a sustituir las llamadas repetidas a una función concreta (p.e. en el interior de bucles) por el correspondiente código en línea. De esta modo se evita un uso excesivo de la pila para transferir parámetros. Por otra parte, se utilizó el modificador 'register' para almacenar variables de contador de los bucles y variables de uso frecuente en registros internos del DSP cuyo tiempo de acceso es inferior.

Por último, se sustituyeron, en la medida de lo posible, las divisiones por productos. Esto es así porque en el DSP utilizado es necesario llamar a una rutina especifica para realizar divisiones mientras que existe un multiplicador hardware para realizar productos en un ciclo de reloj.

3.1 Optimización en lenguaje de bajo nivel

Al contrario que la mayoría de las técnicas anteriores, las técnicas empleadas en las rutinas en ensamblador surgen del conocimiento profundo de la arquitectura del DSP. En nuestro caso, las más apropiadas son:

1. Uso de instrucciones que permiten realizar productos y acumulaciones en paralelo (MPYF||ADDF) para aligerar el cálculo de productos escalares y filtrados.
2. Utilización de saltos retardados (BD Branch Delayed, DBD Decrement and Brach Delayed, BcondD Branch conditionally Delayed) en la ejecución de bucles. De esta manera se evita romper el pipeline [7] y aprovechar algunos ciclos más de instrucción.
3. Utilización de instrucciones que autogestionan el direccionamiento circular necesario en los filtrados [7], hecho que evita dedicar otras instrucciones a tal efecto.
4. 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.
5. Recurrir a instrucciones del tipo RPTS (RePeaT Single instruction) y RPTB (RePeaT Block of instructions) para la implementación de bucles. Dichas instrucciones evitan la realización de 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 [8,9,7].

4. Resultados

En las Tablas II (codificador) y III (decodificador), vienen reflejados los tiempos de ejecución de los distintos bloques funcionales a lo largo de todo el proceso de optimización. La mayor parte (en valor absoluto) de la reducción del tiempo de ejecución proviene del paso de aritmética de coma fija a coma flotante. Sin embargo, las variaciones porcentuales derivadas de la optimización sí son significativas especialmente en el caso de los filtrados tarea para la cual están mejor dotados los DSPs. La única excepción es la rutina de filtrado paso alto debido a que, en este caso el código generado por el compilador ya era óptimo. Indicar, también, que la mayor parte de la reducción del tiempo de ejecución

Etapa del codificador	Coma fija [ms]	Coma flotante sin optimizar [ms]	Coma flotante optimizada [ms]
Filtrado paso alto	19.4	0.2	0.2
Análisis LPC	219	19.4	14.9
Cálculo coefs. filtro pond. espectral	20.3	1.2	0.5
Filtrado de la señal de entrada	27.9	2	0.6
Cálculo del pitch en lazo abierto	180.4 (143)	8.2 (3)	5.8 (2.8)
Búsqueda codevectors	123×4 (107.6×4)	7×4 (5.9×4)	3.9×4 $3.10\ (2.6 \times 4)$
TOTAL	**959 (860)**	**59 (49.4)**	**37.6 (29.4)**

Tabla II: Tiempo de ejecución de las rutinas del codificador para tramas sonoras y sordas (este último entre paréntesis caso de ser distinto). La entrada 'Búsqueda de codevectors' incluye el cálculo del pitch en lazo cerrado.

Etapa del decodificador	Coma fija [ms]	Coma flotante sin optimizar [ms]	Coma flotante optimizada [ms]
Reconstrucción de la excitación	6.2×4 (5.1×4)	0.4×4 (0.3×4)	0.3×4 (0.2×4)
Reconstrucción coeficientes LPC	23	0.7	0.4
Filtrado de síntesis LPC	3.6×4	0.2×4	0.05×4
Filtrado de realce de pitch	4.2×4 (-)	0.1×4 (-)	0.1×4 (-)
Postfiltrado espectral	9.2×4	0.5×4	0.2×4
Otros	18	1.8	0.6
TOTAL	**133.8 (112.6)**	**7.3 (6.5)**	**3.6 (2.8)**

Tabla III: Tiempo de ejecución de las rutinas del decodificador para tramas sonoras y sordas (este último entre paréntesis caso de ser distinto).

derivada de la optimización (21.4 ms para tramas sonoras) proviene de las rutinas de análisis LPC y de Búsqueda de codevectors (4.5 y 12.4 ms respectivamente).

Por último, nótese que las diferencias de tiempo de ejecución entre tramas sonoras y sordas en el codificador son mayores que en el decodificador. Esto es debido a que el *análisis* de dichas tramas es más complicado ya que los codebooks son mayores y es necesario llevar a cabo el cálculo de la frecuencia de pitch.

Por lo que hace referencia a la calidad subjetiva de la voz sintetizada, decir que es notablemente superior a la que se obtiene con el codificador FR [9]. Las razones son el mayor grado de sofisticación del algoritmo de búsqueda del periodo de pitch, y la posibilidad de usar codebooks relativamente grandes gracias a la estructura VSELP de éstos.

Como conclusión, decir que se ha logrado una reducir el tiempo de ejecución en un factor de 25 (aprox). Aún y no ser posible la ejecución en tiempo real ($t_{proceso} \leq$ duración de trama=20 ms) sobre este DSP, otros DSPs disponibles en el mercado estarían perfectamente capacitados para ello.

Referencias

[1] "European digital telecommunications system; Half rate speech. Part 2: Half rate speech transcoding (GSM 06.20)",*ETSI* version 4.0.0. (Marzo 1995).

[2] Siegmund M. Redl, Matthias K- Weber, Malcolm W. Oliphant "An Introduction to GSM", *Artech House* (1995).

[3] "GSM Full Rate Transcoding. Recommendation GSM 6.10" *ETSI* Versión 3.2.0 (Julio 1989).

[4] Ira A. Gerson and Mark A. Jasiuk "Vector Sum Excited Linear Prediction (VSELP) Speech Coding at 8 kb/s" *ICASSP*, vol. 1, 461-464 (1990).

[5] Allen Gersho, "Advances in Speech and Audio Compression" *Proceedings of the IEEE*, vol. 82, no. 6, 900-918 (1994).

[6] Manfred R. Schroeder & Bishnu S. Atal. "Code Excited Linear Prediction (CELP). High Quality Speech at Very Low Bit Rates", *ICASSP*, 937-940 (1985).

[7] "TMS320C3x Users Guide", Texas Instruments, Digital Signal Processing Products (1991).

[8] Jaime González Guijarro "Implementación del Codificador de Voz *Half-Rate* según Recomendación GSM 6.20 sobre DSP TMS320C31" *Proyecto Final de Carrera,* ETSET Barcelona, UPC (1997).

[9] Carles Antón Haro "Codificador de Voz Multipulso en Tiempo Real según la Recomendación GSM 6.10" *Proyecto Final de Carrera,* ETSET Barcelona, UPC (1994).