VOCODER LPC A 2400 BPS SOBRE TMS320C30 Y DSP32C*

Pere Obrador Espinosa
Francesc Vallverdú Bayés.
Departament de Teoria del Senyal i Comunicacions.
Universitat Politècnica de Catalunya.
ETSETB apt. 30002, 08080 Barcelona.

ABSTRACT

This script describes a vocoder LPC-10 system real time 2400 bps. implementation. The working condition of the system are described as well as the algorithm, which includes a silence detector to reduce the information flow. The implementation on a TMS320C30 DSP process is explained with a detailed time-consumtion table. The algorithm is also implemented on a DSP32C in order to compare its performances with the previous ones obtained with the TMS320C30.

INTRODUCCION

En este artículo se presenta la implementación, en tiempo real, de un sistema de codificación-decodificación de señal de voz basado en análisis LPC. El algoritmo propuesto se ha implementado sobre dos procesadores de señal (TI TMS320C30 y AT&T DSP32C) con el fin de comparar las prestaciones de los mismos. Los procesadores elegidos son de características similares a priori, presentando cada uno de ellos ventajas e inconvenientes según sea el aspecto a considerar.

El sistema de codificación responde a la estrategia de codificación de fuente. Se realiza una parametrización LPC de diez coeficientes para cada trama de voz analizada. La duración de cada una de las tramas es de 22,5 ms. La información a transmitir está compuesta por los parámetros LPC obtenidos, una estimación de la energía de la trama y una estimación del pitch asociado a la misma. Este último parámetro incluye información sobre el carácter sonoro, sordo o transición asociado a la trama analizada. El conjunto de estos 12 valores es codificado según la norma STANAG/4198, obteniéndose los 54 bits que serán transmitidos por trama, con lo que se obtiene una cadencia de 2400 bits por segundo al ser la frecuencia de muestreo utilizada de 8 KHz.

El sistema completo se ha desarrollado para ser utilizado coma banco de pruebas de los dos procesadores de señal mencionados (TI TMS 320C30 y AT&T DSP32C) . Los resultados que se presentan han sido obtenidos en condiciones de prueba equivalentes, es decir, utilizando el mismo tipo de conversores A/D y D/A, filtros analógicos antialiasing y de reconstrucción etc.

A continuación se describe el algoritmo de codificación utilizado y se presentan resultados obtenidos con el sistema trabajando en tiempo real.

---

* Los trabajos contenidos en este artículo corresponden en parte al proyecto nº 132.168 del PIE con la empresa ENHER S. A.

1. SISTEMA DE CODIFICACION DE VOZ.

Se utiliza un algoritmo de codificación paramétrica de voz VOCODER LPC10, en el que se utiliza como modelo de síntesis un filtro A.R. de 10 polos. Con él se consigue transmitir señal de voz a 2.4Kbps [1,2,3]. En el algoritmo existen tres partes claramente diferenciadas el transmisor, el receptor y el detector de silencios.

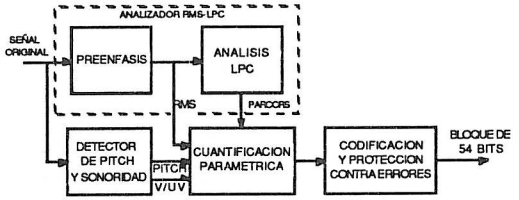

Figura 1. Diagrama de bloques del transmisor.

En la figura 1 se esquematiza el diagrama de bloques que define al transmisor. Una vez digitalizada la señal a una frecuencia de muestreo de 8 KH_z. se enventana constituyendo una trama de análisis LPC cada 22,5 ms.. Para cada trama debe obtenerse el conjunto de los 10 coeficientes correspondientes al filtro de síntesis, la energía de la trama y el valor del periodo de pitch. El conjunto de estos 12 parámetros es codificado según las normas STANAG/4198 obteniéndose bloques de 54 bits por trama.

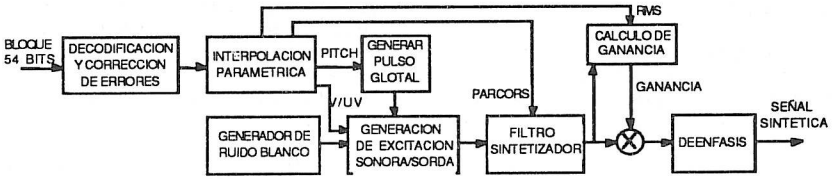

Figura 2. Diagrama de bloques del receptor.

En la figura 2 se presenta el diagrama de bloques del sistema receptor. Este debe decodificar la señal recibida y recuperar los coeficientes LPC, valor de energía y valor de pitch, que permiten realizar la síntesis de voz correspondiente a cada trama. Si la trama es sonora se realiza una interpolación síncrona con el pitch, de los parámetros entre dos tramas consecutivas. Se genera la excitación correspondiente al tipo de trama recibida (sonora, sorda o transición) y se filtra con el filtro de síntesis correspondiente. Finalmente se realiza un ajuste de ganancia y se deenfatiza la señal sintetizada.

La transmisión digital de señal se realiza a través de una red de conmutación de paquetes. Ello implica que los bloques a transmitir deben ser adecuadamente empaquetados. Las repercusiones del sistema de transmisión sobre el sistema de codificación de voz son de dos tipos. La primera es que la red introduce un retardo considerable (entre 100 i 300 ms.), la segunda es que el proceso de empaquetamiento consume aproximadamente 3 ms. por trama.

1.1. Análisis.

El algoritmo de análisis se divide en dos partes. Por un lado se preenfatiza y enventana la señal con una ventana de Hamming. Se calcula el valor RMS de la energía de la trama, y se obtienen los parcors correspondientes con el algoritmo de Durbin-Levinson [1]. Por otro lado debe obtenerse el tipo de trama de señal de voz (sonora, sorda o transición). Si se está procesando una trama sonora debe obtenerse el valor del periodo de pitch, que se limita a un máximo de 156 muestras y a un mínimo de 20. Si la trama es sorda se indica dando un valor de periodo de pitch igual a cero. Por último, si la trama corresponde a una transición sonoro-sordo o sordo-sonoro se asigna un uno al periodo de pitch [2].

El detector de pitch es crítico en entornos ruidosos si se pretende obtener una calidad de señal decodificada aceptable. Por ello se utiliza el algoritmo descrito por Moreno et Al [2] que constituye una mejora considerable respecto al método de autocorrelación.

1.2. Síntesis

El sintetizador responde al diagrama de bloques mostrado en la figura 2. En primer lugar se realiza una interpolación de los parámetros recibidos por trama. Según se reciba una trama sorda o sonora se genera la excitación correspondiente del filtro de síntesis. Para tramas sordas se utiliza un generador de ruido blanco, mientras que para tramas sonoras se genera un tren de pulsos glotales. Si la trama recibida es una transición, se genera media trama con los parámetros de la trama anteriormente recibida, i la otra media con los de la trama futura. Evidentemente el sistema precisa introducir un retardo mínimo de una trama tanto para interpolar como para procesar las transiciones. Para evitar errores de detección de pitch suele utilizarse un filtro de mediana. En este caso el mínimo retardo es de dos tramas. En el sistema implementado no se permite tal retardo debido al tipo de red utilizado. Por ello, puede darse la circunstancia en recepción de recibir dos o tres tramas seguidas correspondientes a transición, que deben ser tratadas adecuadamente.

1.3. Dobles y triples transiciones

El tener que sintetizar una trama de transición seguida de otra trama de transición (pitch=1), no está contemplado en el algoritmo original ya que el filtro de mediana eliminaría tal posibilidad. Debido a la configuración del autómata finito del detector de pitch, sólo se produce una doble transición después de una trama sonora, por lo que se puede considerar un error del detector de pitch, pues a priori después de la segunda transición se volverá a sonoro. Por esto la solución que se propone es sintetizar la primera trama de transición como sonora con pitch constante e igual al de la trama anterior y con interpolación del resto de sus parámetros con los de la trama siguiente. Se pasa a sintetizar la siguiente trama, que también es de transición, y se reciben los parámetros de una tercera trama siguiente a ésta. Existen dos posibilidades:

a) la siguiente trama es sonora. Se propone sintetizar la segunda trama de transición con un pitch que se interpolará entre el de la primera trama de transición y el de la tercera trama , y con el resto de parámetros interpolados entre los suyos y los de la tercera trama.

b)la siguiente trama es sorda. En este caso, poco probable, se tiene una triple transición. La solución que se adopta es sintetizar la segunda trama de transición como una trama de transición normal, o sea media trama con los parámetros de la anterior y media con los de la posterior.

1.4. Detector de silencios.

El detector de silencios [8] tiene como misión avisar al empaquetador, en caso de silencio y que éste no envíe el paquete correspondiente a la trama de silencio, de manera que se relaja el flujo de información por la red.

3. IMPLEMENTACION DEL ALGORITMO.

3.1. TMS320C30

El algoritmo se implementa en tiempo real en un sistema de desarrollo que cuenta con un DSP TMS320C30 de TI, dos canales de conversión A/D D/A y un banco de memoria externa de 1MW (1W = 32 bits).

El algoritmo se implementa en primer lugar en un PC (386 20MH$_Z$) en lenguaje C, sin hacer ningún tipo de optimización desde el punto de vista algorítmico, ni programación en ensamblador. Esta simulación es particularmente lenta (50s/trama), pero el código C que se implementa en el PC es el que servirá de base para la implementación en el DSP.

El primer paso de la implementación en el DSP es retocar el código C para que el compilador [4,5] (versión 3.0) pueda optimizar el código máquina generado. Este compilador optimiza bastante

bien el código C escrito basándose en apuntadores a vectores, y utilizando la post y pre-modificación de los mismos. En este punto también se realizan una serie de mejoras algorítmicas del programa global para reducir tanto el tiempo de ejecución como la memoria requerida. Cabe decir que el compilador no aprovecha muchas de las prestaciones del DSP como puede ser la multiplicación y acumulación en un sólo ciclo (MAC). Esta es la versión 1 de implementación en la tabla 1.

La utilización intensiva de variables en registros también genera menos código y mayor rapidez de ejecución. Una limitación es que el compilador sólo permita 2 registros para enteros, 2 para reales y 4 para apuntadores, y no deja cambiar sus contenidos en la función en que se declaran. Una vez se modifica el código con uso intensivo de registros se obtiene la versión 2 con una reducción drástica en el tiempo de ejecución, pero lejos todavía del tiempo máximo disponible.

Se realizan a continuación una serie de mejoras en cuanto a tiempo de ejecución, que no en cuanto a espacio de memoria ocupado. Se trata de eliminar el cálculo de dos funciones muy costosas en tiempo, como son la ventana de Hamming y la generación de ruido gausiano blanco. En su lugar se leen unas tablas residentes en memoria estática. Sin embargo, la optimización más importante es la función Producto_Escalar escrita en ensamblador y aprovechando al máximo las posibilidades del DSP (MAC, cache, etc.). El Producto_Escalar se utiliza en cálculo de energías, y sobre todo en la implementación de las autocorrelaciones. Los tiempos medidos en esta versión 3 aparecen también en la tabla 1, y como se observa, el algoritmo ya funcionaria en tiempo real si tuviera que ser implementado sin tener en cuenta la red de transmisión.

La versión 4 es la definitiva, y para rebajar el tiempo de ejecución se realizan una serie de mejoras algorítmicas en el detector de silencios, se elimina el cálculo de la correlación de pitch en los puntos que van de n=1 a n=19, ya que no se necesitan, y se implementa el filtro de síntesis ("Lattice") en ensamblador optimizado. Con todo esto se consigue rebajar la barrera de los 19.5 ms impuesta por el gestor de paquetes. Destacar finalmente que los tiempos dados son tiempos máximos de proceso. Solamente a modo de ejemplo citar que para tramas sordas, el tiempo medio de ejecución es de unos 12 ms, mientras que para tramas sonoras es de unos 17 ms.

FUNCION	VERSION 4	VERSION 3	VERSION 2	VERSION 1
FILTRO PASO BAJO	3.28	-	-	7.20
ENVENTANADO PITCH	0.32	-	-	0.39
DETECTOR DE PITCH	4.76	5.55	24.00	79.10
FILTRO PREENFASIS	0.08	-	-	0.34
VENT. LPC (HAMMING)	0.35	-	1.74	2.02
CALCULO DE ENERGIA	0.05	-	0.14	0.40
ALGORITMO DE DURBIN	0.99	1.21	3.87	12.58
CODIFICACION	0.27	-	-	0.47
DECODIFICACION	0.23	-	-	0.28
SINTESIS SONORA	4/6.8	-	4.7/7.9	14/27
" SORDA	2.30	3.32	8.24	15.53
FILTRO DEENFASIS	0.15	-	-	0.39
DETECTOR SILENCIOS:				
VOZ	0.82	1.17	2.73	3.72
' SILENCIO	1.31	2.17	3.68	4.97
INTERRUPCION A/D,D/A	0.40	-	-	-
CONTROL ALGORITMO	0.35	-	-	0.50
TOTAL MAXIMO	19.34	21.21	46.01	135.64

Tabla 1. Tiempo en milisegundos de cada función implementada.

3.2. DSP32C. Banco de pruebas comparativo.

El algoritmo que se utiliza como banco de pruebas para comparar el comportamiento y prestaciones de los dos DSPs es una versión del algoritmo totalmente optimizada en C, pero sin utilizar funciones en ensamblador, por tanto no es en tiempo real.

Se implementa el código en el DSP32C haciendo las modificaciones oportunas, debido a que el compilador de C de AT&T [6,7] (versión 1.3.1) no es ANSI -C, y debido a que la gestión de memoria del DSP32C permite diferenciar palabras de 32, 24, 16 y 8 bits mientras el TMS320C30 sólo puede referenciar palabras de 32 bits. Esto último incide sobre todo en las rutinas de comunicación PC-DSP.

Una vez se tiene una versión equivalente ejecutándose en los dos DSPs se pasa a ver las prestaciones de cada uno en cuanto a calidad, tiempo de ejecución y memoria ocupada.

La calidad de las señales sintetizadas es la misma en los dos casos. Incluso no se aprecia diferencia entre la sintetizada con el DSP32C al ser analizada con TMS320C30.

Los tiempos de ejecución son significativamente diferentes, ya que el ciclo de instrucción de los dos DPSs ya lo es. El TMS320C30 tiene un ciclo de instrucción de 60 ns, y el DSP32C lo tiene de 80 ns. La prueba realizada de velocidad da como resultado que el DSP32C es 1.38 veces más lento que el TMS320C30. Teniendo en cuenta que el TMS320C30 tiene un ciclo de instrucción más corto, y que no se han hecho optimizaciones en ensamblador ajustadas a cada DSP, el resultado de la prueba parece lógico.

La ocupación de memoria es menor en el TMS320C30, que ocupa unos 63 Kbytes, mientras el DSP32C ocupa 72Kbytes. Esta diferencia tan notable se debe principalmente al código de las funciones de las librerías del DSP32C que ocupan 9.2 Kbytes, en frente a los 2.6 Kbytes que ocupan las del TMS320C30.

4. CONCLUSIONES

Cabe resaltar la importancia de las herramientas de trabajo como los debuggers y monitores, a la hora de realizar una implementación del tipo que aquí se ha explicado. En este sentido ha sido particularmente útil el sistema debugger de TI sobre el TMS320C30, que permite analizar tanto código ensamblador, como código C. En cambio los compiladores utilizados no generan código máquina suficientemente bueno como para despreocuparse de hacer las optimizaciones a mano. Este es un punto en que los fabricantes están continuamente trabajando, para poder tener en un futuro próximo compiladores que optimicen directamente el código C al máximo, según las características de cada DSP, así como otras herramientas de desarrollo.

5. AGRADECIMIENTOS

Los autores desean agradecer la inestimable ayuda prestada por los componentes del grupo de tratamiento de voz del departamento de Teoria del Senyal i Comunicacions de la U.P.C., que ha permitido obtener los resultados aquí presentados, al haber podido aprovechar toda su experiencia acumulada en temas de codificación LPC. También deseamos mencionar la gran ayuda recibida del grupo de M. Bertran, a nivel informático, del departamento de Matemática Aplicada y Telemática de la U. P. C.

6. REFERENCIAS

[1] L. R. Rabiner, R. W. Schafer. "Digital Processing of Speech Signals". Prentice-Hall 1978.

[2] A. Moreno, J. Aracil. "Pitch detector in speech signals corrupted by noise". EUSIPCO90, I European Signal Processing Conference. pp 1163-1169. Barcelona, España Sept. 1990.

[3] C. Nadeu, J. B. Mariño, A. Oliveras. "SIMBAD: una herramienta para análisis y síntesis de voz", informe interno. Grupo Processat del Senyal, Dep. Teoria del Senyal i Comunicacions, Jun. 1990

[4] "TMS320C30 C Compiler Reference Guide". TI 1990.

[5] "Third Generation TMS320C3X User's Guide". TI 1989.

[6] "WE DSP32C Digital Signal Processor information Manual". The AT&T Documentation Management Organization. 1990.

[7] "WE DSP32 & DSP32C C Language Compiler User Manual". The AT&T Documentation Management Organization. 1988.

[8] E. Lleida, F. Sitjar. "Detector de Silencios", informe interno. Grupo Processat del Senyal, Dep. Teoria del Senyal i Comunicacions, Sept. 1990.