ESTUDIO DE LA IMPLEMENTACIÓN DE ALGORITMOS DE TRATAMIENTO DE VOZ EN FPGAs

Daniel Rolando Llamocca Obregón
(llamocca.dr@pucp.edu.pe)

Grupo de Procesamiento Digital de Señales e Imágenes-Pontificia Universidad Católica del Perú
Av. Universitaria S/N Cdra. 18-Lima 32, Perú
Telf.: +511-4602870 Ext. 304 Fax: +511-2618861

Abstract

The present research shows a digital speech coding system implemented in a FPGA. The main objective is showing that is possible the implementation of speech processing algorithms in FPGAs.

The present digital system may serve as a basis for the development of more complex real-time speech processing algorithms, such as coding, synthesis, voice recognition, and dialog systems. The presented architecture is fully reconfigurable, so that in the future it would be possible to improve the performance and/or make the algorithm more complex. The system is described by the VHDL hardware description language, and the results achieved were obtained using MAX+PlusII 10.2. The design takes 60% of a FLEX10K70RC240-4, and works at a maximum operating frequency of 26.13 MHz.

RESUMEN

El presente trabajo presenta un sistema digital de codificación de voz implementado en un FPGA, a fin de poder demostrar la factibilidad de implementar algoritmos de tratamiento de voz en FPGAs.

Este sistema digital puede servir de base para desarrollar algoritmos más complejos de procesado de voz en tiempo real, como codificación, síntesis, reconocimiento de voz y sistemas de diálogo. La arquitectura lograda es totalmente reconfigurable a fin de que en el futuro sea posible mejorar las prestaciones y /o hacer más complejo el algoritmo. El sistema es descrito en su totalidad en lenguaje de descripción de hardware VHDL y los resultados alcanzados se obtuvieron en MAX+Plus II 10.2. El diseño ocupa el 60% de un FLEX10K70RC240-4, y funciona a una frecuencia máxima de 26.13 MHz.

ESTUDIO DE LA IMPLEMENTACIÓN DE ALGORITMOS DE TRATAMIENTO DE VOZ EN FPGAs

Daniel Rolando Llamocca Obregón
(llamocca.dr@pucp.edu.pe)

Grupo de Procesamiento Digital de Señales e Imágenes-Pontificia Universidad Católica del Perú
Av. Universitaria S/N Cdra. 18-Lima 32, Perú
Telf.: +511-4602870 Ext. 304 Fax: +511-2618861

RESUMEN

El presente trabajo presenta un sistema digital de codificación de voz implementado en un FPGA, a fin de poder demostrar la factibilidad de implementar algoritmos de tratamiento de voz en FPGAs.

Este sistema digital puede servir de base para desarrollar algoritmos más complejos de procesado de voz en tiempo real, como codificación, síntesis, reconocimiento de voz y sistemas de diálogo. La arquitectura lograda es totalmente reconfigurable a fin de que en el futuro sea posible mejorar las prestaciones y /o hacer más complejo el algoritmo. El sistema es descrito en su totalidad en lenguaje de descripción de hardware VHDL y los resultados alcanzados se obtuvieron en MAX+Plus II 10.2. El diseño ocupa el 60% de un FLEX10K70RC240-4, y funciona a una frecuencia máxima de 26.13 MHz.

1. INTRODUCCIÓN

La mayoría de los sistemas de tratamiento de voz se realizan utilizando procesadores digitales de señales (DSPs), debido a que se logran altas tasas de rendimiento en la implementación en hardware comparándolas con las de software. Existen aproximaciones por hardware basadas en ASICs, que pueden lograr prestaciones muy elevadas; sin embargo, el proceso de diseño lento, costoso y engorroso, la imposibilidad de reconfigurar el diseño una vez construido el circuito [1], hacen que la mayoría de diseñadores eviten estas aproximaciones. Recientemente en los FPGAs se pueden implementar complejos algoritmos de tratamiento digital de señales, que es el caso de los algoritmos de tratamiento de voz. Sin embargo no existen mucha información sobre implementaciones de algoritmos de tratamiento de voz para FPGAs.

En el presente trabajo se describe un codificador de voz basado en la técnica de Predicción Lineal (LP). El objetivo de esta investigación es demostrar la factibilidad de implementar un algoritmo de tratamiento de voz basado en el algoritmo LPC-10 en un FPGA. La implementación usando dispositivos lógicos programables, posee la ventaja de permitir el diseño de arquitecturas propias, así como la explotación del paralelismo y segmentación de operaciones [1].

Este Trabajo está organizado de la siguiente manera: la Sección 2 presenta el modelo de síntesis de voz a fin de facilitar la comprensión del funcionamiento del codificador; la Sección 3 explica el codificador implementado, el cual se basa en el algoritmo LPC-10, la Sección 4 muestra la arquitectura del sistema, y se explica cada bloque en forma genérica y con más detalle las partes que requieren atención especial; la Sección 5 muestra los resultados parciales de cada bloque así como los resultados totales. Finalmente, se dan las conclusiones.

2. DESCRIPCIÓN DEL MODELO DE PREDICCIÓN LINEAL

2.1 Algoritmo LPC-10

Debido a que el codificador diseñado implementa casi en su totalidad este algoritmo, se explica a continuación el algoritmo LPC-10, el cual se basa en la técnica LP

El algoritmo LPC-10 describe un codificador de voz, el cual representa a la voz mediante un conjunto de parámetros, y ahorra espacio físico o ancho de banda, pues no se trabaja con las muestras sino con los parámetros.

En general, para cualquier señal debe ser mucho más eficiente transmitir sus parámetros, si es posible determinarlos, que sus muestras. La reconstrucción de la señal a partir de los parámetros es llevada a cabo por el receptor de comunicaciones o por el sistema de recuperación de datos.

Para determinar los parámetros de la voz, se usa la técnica LP, que es un modelado de la señal (los cálculos son simples en comparación con otros modelos). En este caso, se pierde fidelidad al parametrizar la voz, pero ya

que el interés se centra en la información, la técnica de predicción lineal es aceptable para modelarla [2].

2.2 Eficiencia del modelo

Si se toman muestras de voz a 8 KHz, y cada muestra se cuantifica usando 8 bits, en un segundo de voz se tiene:

$$8000\frac{muestras}{segundo}\times 8\frac{bits}{muestra}=64000\frac{bits}{segundo}$$

Se requiere 64kbits para almacenar un segundo de voz. Y, usando el algoritmo LPC-10, se requieren 2.4kbits para un segundo de voz, lo que implica una gran eficiencia. La Figura 1 muestra el esquema de producción de voz.

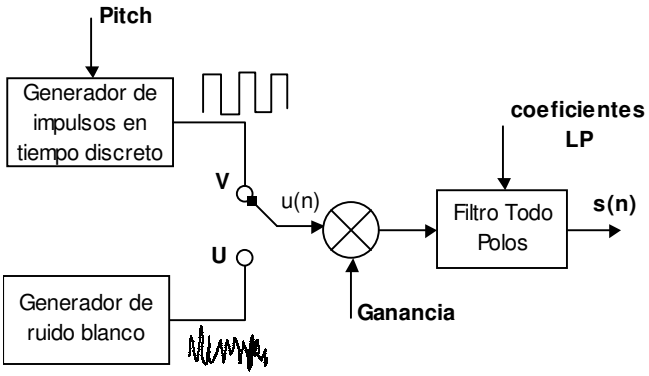

Figura 1. Modelo LP de producción de voz [2]

A partir de este esquema, es claro que los parámetros que se debe obtener para poder sintetizar la voz más adelante son los siguientes:

- Coeficientes LPC (coeficientes del filtro todo polos).
- Decisión secuencia vocálica / no vocálica.
- Frecuencia Fundamental del generador de impulsos en tiempo discreto (o pitch), para el caso de voz vocálica.
- Ganancia (G)

Para la reproducción de voz, es necesario extraer estos parámetros a partir de una secuencia de voz real. A este proceso se le conoce como análisis LP. El algoritmo LPC–10 es un estándar que define cómo se hace el análisis LP: qué algoritmos se van a usar para determinar los parámetros, así como la forma de cuantificarlos.

3. CODIFICADOR BASADO EN EL ALGORITMO LPC-10

3.1 Generalidades

La voz se divide en cuadros de 180 muestras cada uno. La velocidad de muestreo es de 8 kHz. Las muestras de voz se cuantifican mediante la representación en complemento a 2 usando 8 bits de precisión[2].

Cada cuadro dura 22.5 ms, y dado que la voz permanece estacionaria durante intervalos de 10 a 30 ms, todos los parámetros deben recalcularse cada 22.5 ms.

3.2 Obtención de parámetros

La frecuencia fundamental del generador de impulsos en tiempo discreto se estima mediante la función AMDF [1] (average magnitude difference function).

La decisión de secuencia vocálica / no vocálica se estima usando las medidas de cruce por cero y las medidas de energía [3].

Los coeficientes LP se calculan mediante el método de la covarianza, el cual consiste en:

- Obtención de la matriz de covarianza.
- Resolución de la matriz de covarianza:
  - Descomposición de la matriz de covarianza: Método de descomposición de Cholesky
  - Forward elimination
  - Back substitution

Así se obtienen los coeficientes del filtro todo polos que modela el tracto vocal

El algoritmo LPC-10 adicionalmente convierte los parámetros LPC, que corresponden a los coeficientes de un filtro IIR todo polos, a parámetros de un filtro Lattice, y luego efectúa una transformación no lineal (log area ratio) a estos parámetros, a fin de tener una alta estabilidad frente a la cuantización de coeficientes [2].

4. ARQUITECTURA DESARROLLADA

4.1 Diagrama de Bloques

La Figura 2 muestra el diagrama de bloques del sistema desarrollado.

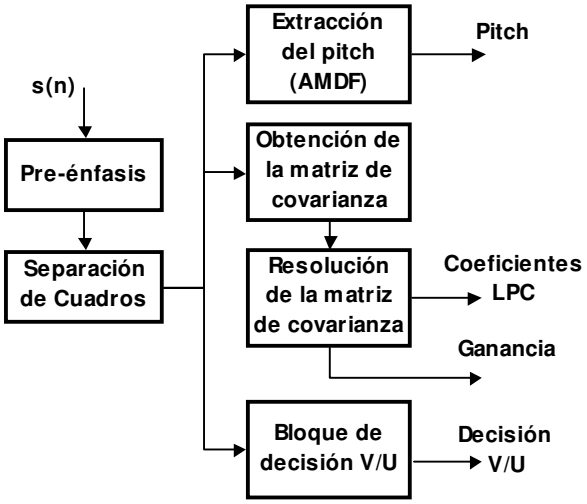

Figura 2. Diagrama de Bloques

4.2 Subsistemas:

4.2.1. Filtro de Pre-énfasis

La señal pasa primero por un filtro de pre-énfasis, a fin de incrementar la energía del espectro de alta frecuencia [2]. Típicamente, se usa un filtro IIR con la siguiente respuesta impulsiva:

$$P(z) = 1 - z^{-1}$$

Debido a que se conocen los coeficientes, este filtro es muy amigable para implementarlo en un FPGA, usando el método de aritmética distribuida [4].

4.2.2. Separación de Cuadros

Luego, la señal pasa por el bloque separador de cuadros. Este bloque se encarga de proveer los cuadros necesarios para el análisis LP, cada cuadro contiene 180 muestras de voz actuales más 90 muestras anteriores [2]. Una memoria de RAM de doble puerto (que permite la lectura y escritura en paralelo), junto con un bloque de control los que se utilizan para implementar este bloque. Así, los datos pueden leerse mientras siguen ingresando a la memoria muestras de voz.

4.2.3. Obtención de la matriz de covarianza

Una vez que el cuadro está en la RAM, el bloque de obtención de la matriz de covarianza se encarga de crear esta matriz, la cual es necesaria para hacer el análisis LP (ver Figura 3). Esta matriz consta en un bloque multiplicador y acumulador, un generador de direcciones de lectura en RAM (la que guarda el cuadro de voz) como de escritura en otra RAM (la que recibe la matriz de covarianza). Cada uno de los elementos de esta matriz está codificado con 22 bits, para trabajar así con la máxima precisión posible.

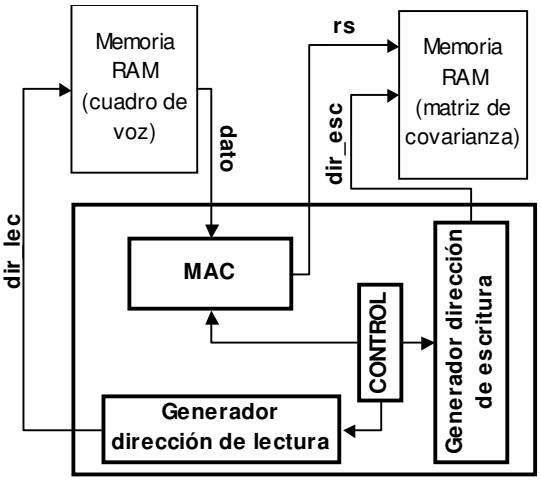

Figura 3. Bloque obtención matriz de covarianza

4.2.4. Resolución de la matriz de covarianza

Seguidamente, el bloque de resolución de la matriz de covarianza se encarga de obtener los coeficientes LPC, así como la ganancia (ver Figura 4).

Debe anotarse aquí, que este bloque es el más complejo de todos, y el que demandó el mayor tiempo de diseño. Esto se debe a que la descomposición de Cholesky implica la obtención de raíz cuadrada y división, y en donde para obtener una alta precisión la longitud de los datos debe ser alta (hasta 32 bits en algunos casos). Los coeficientes LPC obtenidos representan los coeficientes de un filtro IIR todo polos, y para asegurar la estabilidad del sistema se cuantifican con 16 bits. La ganancia (G) se cuantifica con 5 bits [2].

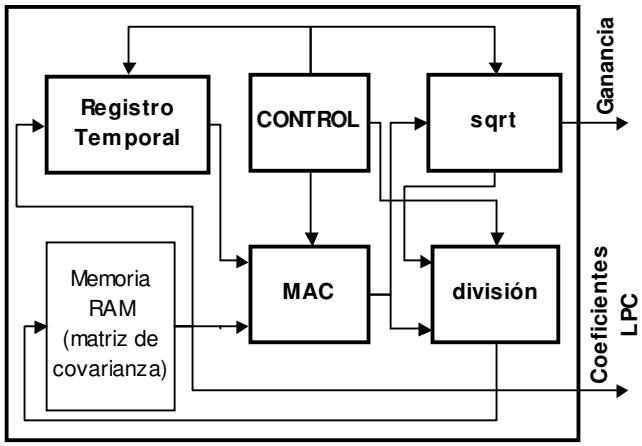

Figura 4. Bloque resolución ecuación de covarianza

4.2.5. Extracción del pitch

En forma paralela, se realiza la obtención de la frecuencia fundamental de la voz (pitch) mediante la función AMDF (average magnitude difference function). Este parámetro se cuantifica con 7 bits [2].

4.2.6. Decisión V/U

El bloque Decisión V/U (ver Figura 2) se encarga de discernir si una secuencia de voz es vocálica o no vocálica. El modelo de producción de voz de la Figura 1 requiere este parámetro a fin de usar la secuencia de excitación adecuada. La decisión V/U se estima usando las medidas de cruce por cero y energía [3]-

5. RESULTADOS

La descripción VHDL de cada módulo fue optimizada para obtener frecuencias de 26.13 MHz (reloj global del Design Laboratory Package UP2) y ocupar el menor

número de celdas lógicas, para implementarlo en FPGAs de propósito general como el EPF10K70RC240-4. La Tabla 1 muestra el número de celdas, la cantidad de bits de memoria usadas por cada módulo, y por todo el módulo.

	LCs	Memory bits
Pre-énfasis	142 (4%)	0
Separador de cuadros	0	4096 (22%)
Extracción del pitch	274 (7%)	0
Obtención matriz de covarianza	254 (6%)	2816 (15%)
Resolución matriz de covarianza	1495 (40%)	256 (1%)
Decisión V/U	130 (3%)	0
TOTAL	**2281 (60%)**	**7168 (39%)**

Tabla 1. Resultados

El Bloque de Pre-énfasis ocupa una cantidad mínima de celdas lógicas, debido a que el filtro IIR se implementa usando el método de aritmética distribuida, el cual mejora el rendimiento y reduce el consumo de celdas lógicas [4].

El Bloque Separador de cuadros es una memoria que almacena el cuadro de voz con la que se va a trabajar (270 muestras o 270 bytes). Para implementar esta memoria en el FLEX se utilizó la Megafunción LPM 'lpm_ram_dp', y dado que 270 bytes se direccionan con 9 bits, la memoria tiene hasta 2^9 = 512 bytes (aunque sólo se usen 270 bytes). La cantidad final de bits usados es: 512 x 8 = 4096 bits.

El Bloque Extracción del Pitch ocupa un moderado número de celdas lógicas, pues la función AMDF se basa en operaciones sencillas como sumas y comparaciones, aunque con una longitud de palabra grande (16 bits).

El Bloque Obtención de Matriz de Covarianza ocupa una moderada cantidad de celdas lógicas, debido a que consta de una unidad multiplicadora - acumuladora (MAC), y de lógica de generación de dirección de lectura y escritura (ver Figura 3). Cada elemento se cuantifica con 22 bits, y se requieren 100 posiciones de memoria para almacenar la matriz. Dado que 100 palabras se direccional con 7 bits, la memoria tiene hasta 2^7 = 128 palabras de 22 bits. Entonces, esta memoria usa 128 x 22 = 2816 bits de memoria para almacenar la matriz de covarianza generada.

El Bloque Resolución Matriz de Covarianza, es el de mayor consumo de celdas lógicas, debido a que involucra los algoritmos más complejos. Este bloque contiene un registro temporal (ver Figura 4), que utiliza 256 bits de memoria y almacena datos que se usan reiteradas veces, a fin de mejorar el rendimiento del sistema.

El Bloque Decisión V/U ocupa una mínima cantidad de celdas lógicas, debido a que la energía y las medidas de cruce por cero se hallan mediante sumas.

Se utilizó la síntesis FAST, la cual además de aumentar la frecuencia del sistema, provocó una reducción en el consumo de celdas lógicas, esto se debe a que la mayoría de las funciones descritas se basan en sumadores, multiplicadores, comparadores, los cuales hacen uso extensivo de los carry chains y mejoran el rendimiento del sistema [5]. Debido a la gran cantidad de procesamiento de datos (longitud de datos de hasta 32 bits) la frecuencia del sistema es de 26.212 MHz. Sin embargo, esta elevada longitud de datos asegura la fiabilidad del sistema.

6. TRABAJOS FUTUROS

Los coeficientes LPC deben convertirse primero a parámetros de celosía para que tengan mayor estabilidad, y luego deben ser nuevamente convertidos a log area ratios y obtener una mayor estabilidad. Así estaría descrito en su totalidad el algoritmo LPC-10.

El algoritmo LPC-10 describe un codificador básico, se desea en el futuro implementar codificadores más complejos. También se espera poder realizar el diseño de un sintetizador de voz y de esta manera tener un sistema codificador-sintetizado descrito totalmente en hardware y con un rendimiento óptimo.

7. CONCLUSIONES

Las características del sistema desarrollado son apropiadas para implementar en FPGAs. Sin embargo, debe notarse que el codificador descrito es un sistema básico. En general un algoritmo de tratamiento de voz es complejo, así que se requiere un FPGA de gran capacidad. Sin embargo, esto ya es posible debido a la aparición de FPGAs con abundantes celdas lógicas y funciones especializadas de procesado digital de señales. El presente trabajo demostró que es posible implementar este tipo de algoritmos en un FPGA de propósito general como el FLEX10K70RC240-4.

8. REFERENCIAS

[1] Digital Signal Processing with Field Programmable Gate Arrays / U. Meyer - Baese; Springer-Verlag Berlin Heidelberg, May 2001.

[2] Discrete-Time Processing of Speech Signals / John R. Deller , Jr, John G. Proakis, John H. L. Hansen; IEEE Press,2000

[3] An algorithm for Determining the Endpoints of Isolated Utterances / L.R. Rabiner and M.R. Sambur, June 1974.

[4] Implementing FIR Filters in FLEX Devices, / ALTERA Corporation App. Note 73, February 1998.

[5] FLEX10K Embedded Programmable Logic Data Sheet