Sistema empotrado de reconocimiento de voz sobre FPGA

Jesús Balosa, Francisco J. Crespo, Angel Barriga
IMSE-CNM-CSIC/Univ. Sevilla
Sevilla, España
barriga@imse-cnm.csic.es

***Abstract*—En esta comunicación se presenta un sistema empotrado sobre FPGA de reconocimiento de voz que aplica el algoritmo LPC (*Linear Predictive Coding*). El sistema está basado en el procesador MicroBlaze de Xilinx. Se describe el desarrollo del sistema desde la implementación del controlador del códec de audio (tanto el hardware como el desarrollo de los drivers) hasta la adaptación del algoritmo LPC a los requerimientos de la arquitectura hardware.**

I. INTRODUCCIÓN

Los sistemas automáticos de reconocimiento de voz (ASR, *Automatic Speech Recognition*) han experimentado un notable avance en las últimas décadas. Sin embargo la mayoría de las aplicaciones de los sistemas ASR están basadas en realizaciones software sobre ordenadores. El desarrollo de aplicaciones sobre sistemas empotrados hace necesaria la adaptación de los sistemas ASR. El objetivo de esta comunicación consiste en presentar la puesta a punto de un entorno de desarrollo para aplicaciones ASR empotradas. Esto supone la especificación de la plataforma de desarrollo y la adaptación de técnicas ASR para dicha plataforma. Por lo tanto en esta comunicación se va a presentar por un lado la adaptación de una plataforma hardware reconfigurable basada en FPGA que permita configurar diferentes arquitecturas de sistemas empotrados y, por otro lado, el desarrollo de un algoritmo de reconocimiento de voz sobre dicha plataforma hardware.

Se ha seleccionado como plataforma de desarrollo una placa basada en FPGA de Xilinx ML505 que, entre otros componentes, dispone de entradas y salidas de audio, códec de audio, dispositivos de comunicación, memorias (SRAM, DDR2, CompactFlash), etc. Se ha desarrollado un controlador de audio como periférico del procesador MicroBlaze.

El desarrollo del algoritmo de reconocimiento de voz ha requerido su adaptación a los requerimientos del sistema empotrado (recursos reducidos de memoria, procesador sin unidad de manejo de memoria, etc). Como resultado se dispone de una plataforma de desarrollo de aplicaciones ASR que facilita tanto la exploración de nuevas arquitecturas hardware como el desarrollo de otros algoritmos empotrados de reconocimiento de voz.

II. RECONOCIMIENTO DE VOZ

Una de las técnicas más empleadas en la codificación del modelo de voz corresponde a la técnica LPC (*Linear Predictive Coding*). Dicho algoritmo permite representar una señal de voz de 160 muestras en tan sólo 13 datos. Esto permite su aplicación a la compresión de voz, transmisión digital (voz sobre IP, PCS, GSM) y, en nuestro caso, reconocimiento.

Los elementos que mejor caracterizan la voz humana son las frecuencias de los formantes del tracto vocal y las propiedades de la señal de excitación generada por las cuerdas vocales. La técnica LPC permite representar la voz mediante un número reducido de parámetros en lugar de tener que almacenar la forma de onda [1]. Esto se debe a que es posible predecir una señal mediante la siguiente expresión:

$$y(n) = \sum_{i=1}^{N} a_i x(n-i) \tag{1}$$

donde *y(n)* es la muestra predicha en el instante *n* y a_i son los coeficientes LPC.

La diferencia entre la muestra predicha y la señal se denomina error de predicción:

$$e(n) = x(n) - y(n) = x(n) - \sum_{i=1}^{N} a_i x(n-i) \tag{2}$$

El objetivo del algoritmo LPC es minimizar ese error. Por lo tanto, con objeto de tener coeficientes LPC que minimicen ese error es necesario diferenciar la ecuación 1 respecto a cada coeficiente e igualar a cero. Esto da lugar a un sistema de *N* ecuaciones lineales [2]:

$$Ra = r \tag{3}$$

donde

$$R = \begin{bmatrix} R(0) & R(1) & \dots & R(N-1) \\ R(1) & R(0) & \dots & R(N-2) \\ \vdots & \vdots & \vdots & \vdots \\ R(N-1) & R(N-2) & \dots & R(0) \end{bmatrix}$$

$$r = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(N) \end{bmatrix}$$

El algoritmo de Levinson-Durbin es un método iterativo que permite obtener los coeficientes LPC [2].

III. PLATAFORMA DE DESARROLLO

A. *Codec de audio*

La placa de desarrollo ML505 dispone de una FPGA de Xilinx de la familia Virtex-5. Dicha placa contiene entradas y salidas de audio controladas mediante el códec AC97 de Analog Device AD1981B [3]. Entre las características de este códec destacamos las siguientes:

- Salida S/PDIF de 20 bits para el formato de dato con frecuencia de muestreo de 48 kHz y 44.1 kHz.
- Muestreo de frecuencia variable para salida y entrada de audio.
- Códec estéreo *full-duplex*.
- Tasa variable de las muestras de 7040 Hz a 48 kHz con resolución de 1Hz.
- Micrófono estéreo con función de preamplificado.

El códec dispone de módulos de conversión analógico-digital (ADC) y digital-analógico (DCA). Los convertidores ADC y DCA están basados en convertidores Σ-Δ. El módulo de conversión digital-analógico se utiliza para generar la salida de audio, es decir, para reproducir sonidos. Está compuesto por 4 convertidores Σ-Δ, dos de 16 bits y dos de 20 bits.

Por otro lado el módulo de conversión analógico-digital recibe los datos de entrada de audio y convierte la señal analógica en una señal digital. Este módulo está constituido por dos convertidores Σ-Δ de 16 bits.

Las entradas y salidas de audio están conectadas a 4 puertos tipo *jack*. Dispone de dos entradas de micrófono y conexión en línea (*Line In*) y dos conectores de salidas de audio (para auriculares y conexión en línea).

B. *Módulo IP del controlador AC97*

La versión del módulo IP del controlador AC97 de Xilinx empleado es la versión 1.00a. Dicho controlador ha sido diseñado para configurar el códec AC97, grabar (capturar) sonido usando el bus OPB y reproducirlo usando el bus FSL. Puesto que en las actuales versiones de Microblaze el bus OPB ya está obsoleto se ha adaptado el controlador mediante un puente PLB-OPB. También se han corregido errores en la descripción de dicho componentes que originaban que tanto la función de grabación de sonido como la lectura del registro de estado del AC97 no funcionaran.

Los problemas con la grabación y con la lectura del registro de estado del AC97 se encuentran relacionados entre sí. Ambos son causados por un error en el fichero de descripción del hardware del controlador. Dicho error se debe a una mala alineación de las lecturas de datos en serie, que son desplazados, lo que provoca que el controlador se encuentre sin el estado válido en el registro de entrada o en los datos de sonido.

El controlador recibe/transmite los datos desde/hacia el códec AC97 usando dos flujos de datos series unidireccionales, `SData_In` y `SData_Out`. El dato es separado en tramas simultáneas en las que cada trama está dividida en 12 ranuras de datos. Además un reloj `Bit_Clk`, es generado por el códec y transmitido al controlador. Por su parte el controlador envía una señal `Sync` para indicar que empieza una trama.

El flujo de datos de entrada se lee desplazando secuencialmente el flujo de datos de `SData_In` y escribiendo dicho dato en el registro apropiado al final de una ranura por medio de la señal `Slot_End` generada en el controlador. Esta señal se activa un ciclo de reloj antes del final de la actual ranura porque el flujo de salida debe ser cargado antes del comienzo de la siguiente ranura. El problema de la mala alineación de las lecturas de datos en serie se solucionó añadiendo un ciclo de reloj de retraso antes de leer el registro de desplazamientos de datos.

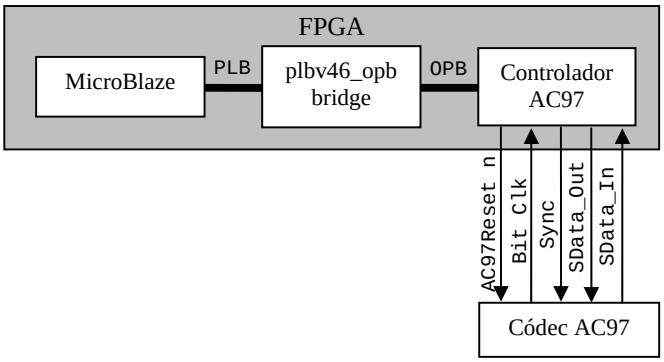

Figura 1. Arquitectura del sistema de captura de sonidos

C. *Desarrollo de los drivers*

Los *drivers* son aplicaciones software que permiten configurar y controlar el controlador AC97. Estas aplicaciones facilitan el desarrollo de aplicaciones de usuario. En concreto los *drivers* desarrollados contienen la declaración de las direcciones de memoria donde se encuentran los registros internos del controlador AC97, así como las macros para el manejo dichos registros y los valores constantes más significativos. La tabla 1 describe las principales funciones.

Estas funciones ofrecen las facilidades para inicializar e interaccionar con el códec AC97. El códec se inicializa con las funciones XAC97_SoftReset, XAC97_HardReset. Estas funciones inicializan los registros del códec a los valores adecuados para realizar la captura de sonidos. Las funciones WriteAC97Reg y ReadAC97Reg son las funciones principales para realizar la grabación y reproducción de sonido.

Junto a estas funciones existen una serie de constantes y macros que facilitan el uso del códec y su configuración.

Además de las funciones de bajo nivel mostradas en la tabla 1 se han desarrollado funciones de alto nivel que el usuario puede utilizar para manejar el códec desde la aplicación software. Dichas funciones son:

- **init_sound**: Inicializa el dispositivo. Para ello aplica un *reset* hardware, se limpia la FIFO y se hace un *reset* software. La tarea de inicialización requiere establecer una secuencia de operaciones en un determinado orden de ejecución. Tras la inicialización el códec informa al controlador que está operativo. A continuación se activan los convertidores ADC y DAC y se establece la frecuencia de muestreo. Finalmente se configuran los controles de volumen y del micrófono.
- **rec_sound**: Se capturan los datos de la entrada de audio y se almacena en memoria.
- **play_sound**: Se habilitan los controles de sonido para la reproducción por la salida correspondiente. Se envían al códec los datos de reproducción almacenados en memoria.

TABLA 1. *DRIVERS* DEL CONTROLADOR DEL CÓDEC AC97

Función	Descripción
WriteAC97Reg	Escribe un valor en la dirección indicada del códec AC97
ReadAC97Reg	Lee un dato de la dirección indicada del códec AC97
XAC97_ClearFifos	Limpia la memoria FIFO del controlador AC97
XAC97_SoftReset	Realiza un *reset* software del códec AC97
XAC97_HardReset	Realiza un *reset* hardware del códec AC97

IV. IMPLEMENTACIÓN DEL ALGORITMO LPC

Existen multitud de implementaciones software del algoritmo LPC tanto en aplicaciones específicas como formando parte de librerías para el tratamiento de señales de audio. Estas implementaciones son inadecuadas para un sistema empotrado debido a diversos aspectos relacionados con la implementación del algoritmo. Así, por ejemplo, dichas realizaciones suelen hacer uso de memoria dinámica tanto para la señal de entrada de audio como para los vectores intermedios y para la base de datos con la cual se compara el locutor a identificar. Otro aspecto a tener en cuenta corresponde a la limitación de tamaño de memoria del sistema empotrado.

La señal de audio que se procesa en el algoritmo tanto para el entrenamiento como para el reconocimiento es adquirida mediante un micrófono conectado a la placa de desarrollo basada en FPGA. El códec AC97 convierte dicha señal en una señal digital que es capturada por el controlador AC97 y almacenada en memoria. A continuación se realiza el procesado de los datos de acuerdo con el esquema mostrado en la figura 2.

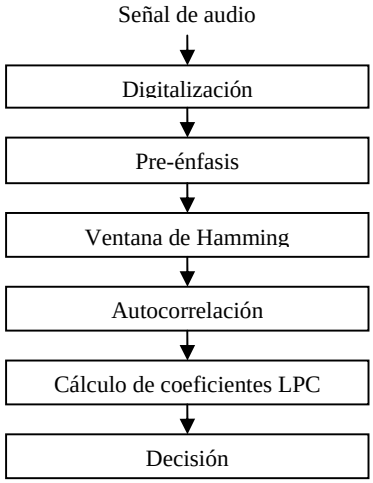

Figura 2. Flujo del proceso de identificación

La etapa de pre-énfasis realiza un filtrado para hacer más significativas las frecuencias altas de la señal de voz. Este filtro de preénfasis obedece a la expresión siguiente:

$$s(n) = v(n) - a \times s(n-1) \qquad (4)$$

donde $v(n)$ es la señal de voz de entrada y $s(n)$ la señal filtrada.

La siguiente etapa corresponde a una segmentación en ventanas de Hamming en intervalos de 20 ms. Durante ese tiempo la señal se considera cuasi estacionaria. A continuación se aplica el producto de la señal con la ventana de Hamming con objeto de suavizar los bordes de dicha ventana:

$$W_n = 0.54 - 0.46\left(\frac{2\pi n}{N}\right) \quad 0 \le n \le N \qquad (5)$$

siendo $W_n=0$ en cualquier otro caso.

A continuación se calcula la autocorrelación de la señal:

$$r_l(m) = \sum_{n=0}^{N-1-m} [x_l(n)\bar{x}_l(n+m)] \quad m = 0,1,...,p \qquad (6)$$

donde p es el orden del análisis LPC [4]. Valores típicos de p van entre 8 y 16. En el desarrollo que se describe en esta comunicación se ha empleado p=13.

El algoritmo de Levinson-Durbin [2] permite calcular en forma recursiva la solución de una ecuación que involucra una matriz Toeplitz (ecuación 3).

V. OPERACIÓN DEL SISTEMA

El sistema de identificación tiene dos fases de operación: entrenamiento e identificación.

La etapa de entrenamiento consiste en crear una base de datos de los coeficientes LPC de individuos que deberán ser identificados por el sistema. El entrenamiento requiere realizar un conjunto de iteraciones en las que los coeficientes se ajustan realizando la media aritmética con los coeficientes asociados.

La figura 3 muestra un ejemplo que ilustra como a medida que se incrementa el número de iteraciones en el entrenamiento se incrementa la precisión en la identificación.

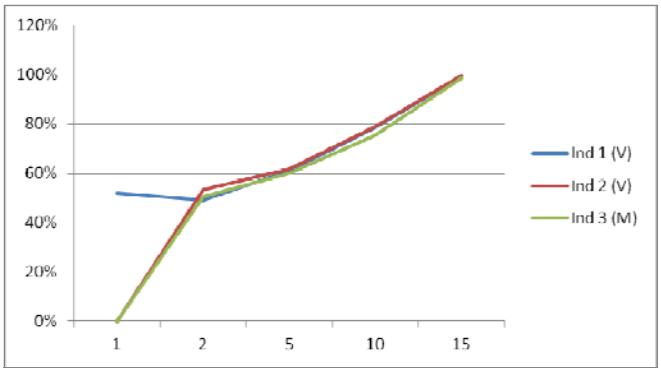

Figura 3. Tasa de reconocimiento en función de la iteraciones en el entrenamiento.

La fase de reconocimiento consiste en determinar la similitud entre los coeficientes LPC del locutor con los contenidos en la base de datos de los individuos entrenados. En este caso se aplica la distancia euclidiana y tomando como referencia dos umbrales, uno de acierto (T_a) y otro de similitud (T_s), se decidirá si se ha identificado al individuo dentro de la base de datos. Los umbrales seleccionados han sido los siguientes:

- T_a, umbral de acierto: $T_a < 0.02$
- T_s, acierto parcial (similitud): $0.02 < T_s < 0.03$
- en otro caso significa que no se ha encontrado un individuo registrado que corresponda al locutor que se quiere identificar

VI. CONCLUSIONES

Se ha descrito un sistema de identificación de voz basado en el procesador MicroBlaze de Xilinx. El sistema ha sido implementado sobre la placa de desarrollo ML505 que dispone de un FPGA Virtex5 así como entradas y salidas de audio controladas mediante el códec AC97 de Analog Device AD1981B. La puesta a punto del sistema ha requerido adaptar un controlador del códec de audio y desarrollar los *drivers* que permiten controlar la captura de audio desde la aplicación software. La aplicación software que implementa el algoritmo LPC se ha adaptado a la arquitectura del procesador. Por lo tanto se dispone de un sistema empotrado que permite desarrollar aplicaciones biométricas de reconocimiento de voz.

AGRADECIMIENTOS

Este trabajo ha sido soportado parcialmente por el proyecto financiado por la Unión Europea MOBY-DIC Project FP7-IST-248858, por el Ministerio de Ciencia y Tecnología bajo el proyecto TEC2008-04920 y TEC2011-24319 con cofinanciación FEDER y por la Junta de Andalucía bajo el proyecto P08-TIC-03674.

REFERENCIAS

[1] Priyabrata Sinha, “Speech Processing in Embedded Systems”, Springer, 2010

[2] L. Rabiner and B. H. Juang. Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cli_s, NJ, 1993.

[3] AD1981B: AC97 SoundMAX® Codec Datasheet, Analog Devices Inc, Rev. C, 2005.

[4] Milan G. Mehta: “Speech recognition system”, Master Thesis, Texas Tech University, 1996.