IMPLEMENTACIÓN DE LA TRANSFORMADA CEPSTRUM CON MÓDULO FFT RECONFIGURABLE.

J. Arévalo, C. Pedraza

Universidad Santo Tomás, Bogotá.

{Julianalejandra, cesar.pedraza}@gmail.com

RESUMEN

Este artículo muestra el proceso de diseño de una arquitectura reconfigurable para la transformada cepstrum, orientado para sistemas embebidos. Dicha transformada consiste en el análisis de una señal para obtener su componente en frecuencia fundamental (caso de una señal de voz) mediante la transformada de Fourier. Esta transformada se caracteriza por ocupar grandes cantidades de recursos de hardware, no siempre disponible en sistemas embebidos, lo que lleva al diseñador a hacer uso de tendencias como la RPD (Reconfiguración Parcial Dinámica). Se presenta el método de diseño del algoritmo en hardware, los resultados que se obtuvieron y finalmente se concluye sobre las ventajas y desventajas del uso de la RPD en este tipo de aplicaciones.

1. INTRODUCCIÓN.

En la actualidad una de las soluciones más comunes a los problemas de diseño son los sistemas embebidos. Gracias a las plataformas de hardware disponibles, un diseñador se puede plantear la posibilidad de incluir un sistema microprocesado con todos sus periféricos dentro de un mismo chip. Estos periféricos pueden ser *cores* que cumplen tareas de comunicaciones, procesamiento de información, entre otros, y que pueden ser abiertos o no abiertos.

A menudo en el diseño de sistemas de procesamiento de información tales como voz y vídeo, es necesario implementar algoritmos con restricciones temporales fuertes que requieren parte o todo su desarrollo en hardware [8].

Pero los recursos de hardware disponibles en un FPGA deben ser usados correctamente, y esto es, optimizar al máximo el espacio requerido para una tarea determinada, y que permita incluir más periféricos en un sistema, que éste sea mas veloz o que tenga un menor consumo de potencia. La reconfiguración parcial dinámica es una de las opciones disponibles que permiten optimizar el uso de hardware en FPGAs.

El objetivo del proyecto es validar biométricamente a una persona para un aplicativo de seguridad. La voz es uno de los sistemas biométricos mas importantes y de relativa facilidad de implementación. Determinar la frecuencia fundamental de la voz (pitch) es posible mediante la transformada Cepstrum, y por ello se implementó en un sistema embebido. La información entregada por la transformada permite encontrar el tono de voz de una persona, y la forma en que varía la frecuencia de la señal de voz y, de esta forma modelar el tracto vocal del hablante, como un sistema lineal invariante en el tiempo. En el presente documento se presenta un diseño realizado en una plataforma VirtexII Pro de Xilinx de la transformada real Cepstrum con una variante de su arquitectura para hacer posible la reconfiguración parcial dinámica. Se pretendía que un porcentaje del área ocupada por el diseño pudiera ser cambiada por un periférico que se requería en el sistema embebido, sacrificando velocidad de procesamiento en la transformada.

Se muestra la forma en que se implementó la transformada rápida de Fourier y el logaritmo, bloques constituyentes de la transformada Cepstrum, así como la arquitectura requerida para que sea reconfigurable. En la sección 2 se presentan las características principales de la transformada Cepstrum, en la sección 3 se muestra la arquitectura utilizada y finalmente se presentan los principales resultados de los diseños.

2. TRANSFORMADA CEPSTRUM.

La señal de voz se puede caracterizar como la señal de excitación proveniente de los pulmones, convolucionada con la respuesta al impulso del sistema que representa al tracto vocal [5]. El cálculo del Cepstrum inicia por obtener el espectro de la señal de voz. Las propiedades de la transformada de Fourier, convierten la convolución en el dominio del tiempo, en una multiplicación del dominio de la frecuencia. Las dos señales pueden ser separadas si se aplica el logaritmo.

$$s(t) = x(t) * h(t) \quad S(f) = X(f) \cdot H(f) \qquad (1)$$
$$\log[S(f)] = \log[X(f)] + \log[H(f)]$$

Donde *s(t)* es la señal de voz, *x(t)* es la señal de excitación y *h(t)* es la respuesta al impulso del tracto vocal. La propiedad aditiva se mantiene si se aplica nuevamente la transformada de Fourier, que permite determinar la frecuencia fundamental de la señal, ya que

las componentes periódicas en la transformación anterior (debidas a los armónicos) se identifican como un pico en la función cepstral, que representa el periodo fundamental de la señal.

$$C(q) = F[\log\{F[x(n)]\}] \quad (2)$$

Aunque las unidades de la función cepstral son de tiempo, para identificarlo como resultado del cepstrum se les da el nombre de *quefrency* y usualmente se analiza en una escala de 0 a 15 ms. para señales de voz [1], donde la parte de baja frecuencia de la señal representa la envolvente del espectro del tracto vocal y la parte de alta frecuencia representa el periodo de la señal de excitación.

3. ARQUITECTURA DEL DISEÑO.

3.1. Arquitectura general.

El sistema se implementó en un FPGA Virtex II Pro XC2VP30 de Xilinx, con una señal de reloj de 100 MHz. Este dispositivo ofrece la posibilidad de implementar sistemas embebidos con dos microprocesadores PowerPC 405, cuyos periféricos pueden ser implementados para un bus OPB o PLB. El objetivo del proyecto es generar un core para este FPGA y que acelere el cálculo de la transformada dentro del sistema embebido.

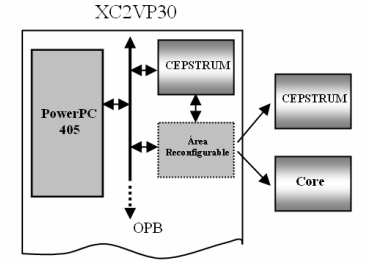

Fig 1. Diagrama de bloques del diseño.

El proceso del cálculo del Cepstrum real, requiere dos bloques de transformada rápida de Fourier (FFT) y un bloque para el cálculo del logaritmo, además de algunos componentes de control y funciones externas para acoplar y manejar el sistema, según se muestra en la figura 2. Los datos que se procesan provienen de una memoria que contiene la señal de voz. La memoria de datos está ubicada en el bloque *memoria* de la primera FFT.

El cálculo de la transformada de Fourier genera un resultado complejo, por lo que se usa un módulo CORDIC que calcule su magnitud antes de entregarlo al bloque del logaritmo y en la salida del sistema.

Fig. 2. Diagrama de bloques de la implementación de la transformada cepstrum.

3.2. Cepstrum.

El cálculo de la transformada Cepstrum se realiza con 4 procesos: la primera FFT, el cálculo de magnitud de los vectores complejos resultantes, la segunda FFT y la magnitud de esta. Cada proceso puede operar independientemente lo que hace posible segmentar el sistema aumentando su desempeño. A continuación se describen cada uno de los bloques funcionales del sistema.

3.2.1. Transformada Rápida de Fourier.

La trasformada de Fourier implementada utiliza el algoritmo Cooley – Turkey base 4 de 1024 puntos, diezmada en frecuencia. La base del algoritmo se ha preferido debido a que opera con más velocidad que la base 2, aunque requiere de más elementos de procesamiento [5]. El objetivo es reducir el tamaño de la DFT, para lo cual, se organizan las muestras de entrada y se calculan múltiples DFT de 4 puntos. Posteriormente, se recombinan los resultados una cantidad de veces determinada por el número de muestras que se quieren procesar. Las consideraciones matemáticas del algoritmo se presentan en [2]. La DFT calculada se describe con la ecuación 3.

$$\begin{aligned} X(4k) &= \sum_{n=0}^{(N/4)-1}\left[x(n)+x\left(n+N/4\right)+x\left(n+N/2\right)+x\left(n+3N/4\right)\right]\cdot W_N^0 W_{N/4}^{kn} \\ X(4k+1) &= \sum_{n=0}^{(N/4)-1}\left[x(n)-j\cdot x\left(n+N/4\right)-x\left(n+N/2\right)+j\cdot x\left(n+3N/4\right)\right]\cdot W_N^1 W_{N/4}^{kn} \\ &\vdots \end{aligned} \quad (3)$$

Donde N es el número de puntos, $x(n)$ la señal de entrada, $X(k)$ la DFT, y W^i_N el factor exponencial de la DFT, también llamado factores de giro.

La DFT correspondiente a la FFT de base 2 se denomina *mariposa*, por su diagrama de flujo. Como una analogía, llamaremos a la ecuación (3) *libélula* para diezmado en frecuencia. Las muestras son combinadas en $N/4$ libélulas y este proceso se realiza un número de veces igual al logaritmo en base 4 de la cantidad de puntos. Estos procesos son denominados *etapas* de la FFT. Para el caso presentado, las etapas son:

$$etapas = \log_4(N) \quad etapas = \log_4(1024) \quad etapas = 5 \quad (4)$$

Los datos son representados en punto fijo con palabras de 32 bits, divididas en 16 para parte real y 16 para parte imaginaria, cada una con el bit más significativo para el signo. La escalabilidad se controló alterando la posición del punto en la representación entre etapa y etapa de la FFT. La arquitectura general de la FFT consta de 4 bloques principales, como muestra la figura 3.

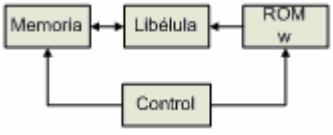

Fig. 3. Arquitectura de la FFT implementada.

Para la implementación de la libélula fue necesario utilizar multiplicadores complejos, que necesitan 4 multiplicadores reales y 2 sumadores reales. Deben tenerse en cuenta los que combinan las señales reales e imaginarias, según la ecuación 3. La libélula utiliza un total de 12 multiplicadores reales y 14 sumadores reales.

Se usa un bloque ROM que contiene las constantes de los factores exponenciales requeridos para el cálculo de la FFT, una memoria para almacenar los resultados parciales de la FFT entre etapas y un módulo de control.

3.2.2. Logaritmo.

La implementación del logaritmo se realizó por medio de una aproximación lineal a trozos, escogida por su simplicidad y buenos resultados. La función de aproximación se obtuvo por medio de segmentos de recta, en los que se toma el valor de entrada, se divide entre una potencia de dos y se le suma un valor constante que proviene de una memoria de solo lectura (ROM).

3.2.3. Algoritmo CORDIC.

Los algoritmos CORDIC son utilizados en diferentes aplicaciones en las que se necesita el cálculo de funciones trigonométricas y no se cuenta con una gran cantidad de recursos de procesamiento [3]. Existen básicamente 2 modos de cálculo CORDIC[4] modo rotacional y modo vectorizado, según lo que se desee calcular. El primero se utiliza en el cálculo de funciones y el segundo para el cálculo de magnitud y fase. En éste diseño se busca encontrar la magnitud del vector de salida de la FFT por lo que se utiliza el modo vectorizado [3].

$$x_{i+1} = x_i - d_i y_i 2^{-1} \quad y_{i+1} = y_i + d_i x_i 2^{-1} \quad z_{i+1} = z_i - d_i \tan^{-1} 2^{-1} \qquad (6)$$

Cada iteración CORDIC requiere dos corrimientos, una búsqueda en una tabla y 3 sumas reales. Así mismo, se requiere el uso de un acumulador que guarde el valor del ángulo de rotación y registros que almacenen los valores actuales de las coordenadas [3]. La magnitud de cada vector se obtiene con 8 iteraciones dado que se han usado registros de 8 bits para la rotación.

3.3. Reconfiguración parcial activa.

El siguiente paso consistió en implementar el diseño usando el flujo para reconfiguración parcial. En el algoritmo de la transformada cepstrum es claro el uso de dos bloques FFT y dos CORDIC, por lo que se ha optó por dos variantes del diseño:

- La primera opción consistió en implementar dos módulos FFT y dos CORDIC, a fin de que se pudiera segmentar el diseño y obtener más velocidad de procesamiento. Esta opción tiene como desventaja el uso de más recursos de hardware.
- La segunda opción consistió en implementar un solo módulo FFT, liberando espacio en el FPGA para ser ocupado por otro periférico.

El flujo de RPD empleado aquí se denomina "Early Access Partial Reconfiguration" EA PR [9], que permite reconfigurar áreas con formas que no cubren columnas completas en el FPGA y cruzar señales del área fija en las áreas reconfigurables sin necesidad del uso de bus macros. Estas cualidades de este tipo de reconfiguración obedecen a las características de la familia VirtexII Pro de Xilinx, tales como la ausencia de glitches al configurar el dispositivo y la configuración parcial por columnas. Se han establecido dos regiones reconfigurables, como se observa en la figura 4. El área reconfigurable A se estableció para cambiar el sistema de control, y la B para implementar el módulo FFT y CORDIC adicional o el periférico opcional para el procesador.

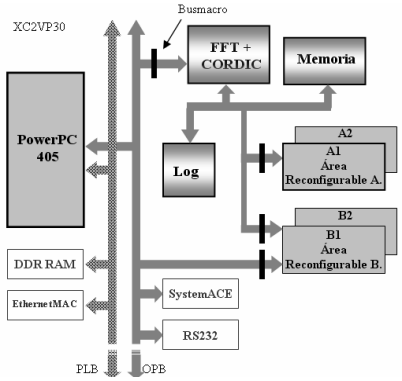

Fig 4. Diagrama de bloques del sistema implementado.

Para comunicar los módulos reconfigurables se usaron de busmacro de tipo slice, que permiten ser ubicados en posiciones pares [9].

4. RESULTADOS.

4.1. Implementación.

El flujo reconfigurable completado para la parte estática se muestra en la figura 5.

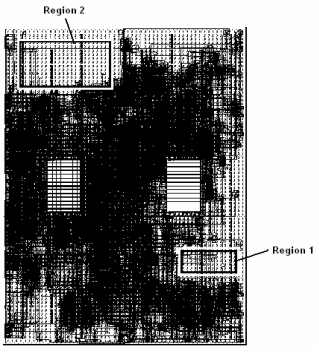

Fig 5. Implementación del sistema estático incluyendo controladores DDR RAM, PLB, OPB, RS232.

Para implementar el sistema completo fue necesario evitar incluir cores para CompactFlash o Ethernet, que ocupan gran cantidad de área del FPGA, y que la

herramienta de ruteamiento no es capaz de finalizar. En la figura 5 se alcanza a observar algunas de las señales de la región estática que cruzan hacia las reconfigurables, situación permitida en el flujo de diseño empleando EA PR. Pero un exceso de este tipo de señales puede complicar la implementación de los módulos reconfigurables, ya que no habría suficiente lugar para su ruteamiento. La tabla 1 muestra el resultado de la implementación de los cores y su velocidad.

Core	Velocidad	Utilización de Hardware	
		Slices	RAM
FFT	~ 52us	2960	32KB
Cepstrum segmentado.	~ 52us	6912	64KB
Ceptrum con 1 módulo FFT	~ 104us	3956	32KB

Tabla 1. Estadísticas de la implementación.

4.2. Desempeño.

Después de completar el diseño y simulación de cada uno de los bloques funcionales del sistema, se realizaron pruebas para determinar su comportamiento y el error presentado. A continuación se presentan los resultados obtenidos.

FFT: Para observar el comportamiento del bloque FFT, se realizaron pruebas con 3 señales de voz inicializadas en la memoria de datos.

El error relativo obtenido en las 3 simulaciones, tiene un promedio de 6,59%, presentando una diferencia importante para los valores pequeños (menores de 10), debido a la representación en punto fijo. Sin embargo, se observa una clara semejanza entre el espectro de la señal, generado en MATLAB con la función FFT, y los resultados del diseño. Esta característica de la forma de onda del espectro, es la que permite realizar el análisis espectral pues indica la presencia de componentes de frecuencia fuertes, aunque no tenga el valor exacto. El promedio del error absoluto es de 13,68 unidades; y es despreciable cuando los valores son suficientemente altos como para ser importantes en el análisis de la señal.

El bloque CORDIC mostró muy buenos resultados, pues permitió calcular magnitudes de los vectores con un error relativo del 0.405% a pesar de no trabajar con punto flotante. Como en el caso del bloque CORDIC, el bloque del logaritmo introduce un error muy pequeño, en este caso menor al 0.1 % por lo que para este tipo de aplicación es bastante tolerable.

Transformada Cepstrum: La implementación del sistema completo se realizó con la unión de todos los elementos anteriores siguiendo la implementación tradicional para la transformada cepstrum. Esto implica que el error total del sistema, será el acumulado del error de los elementos. La figura 6 muestra una comparación entre una señal generada por MATLAB y los resultados del sistema.

El tiempo requerido para el cálculo de cada transformada cepstrum, depende directamente del tiempo de las FFT, pues dependiendo de la etapa en la que se encuentre, se determina que operación debe realizarse con los datos procesados.

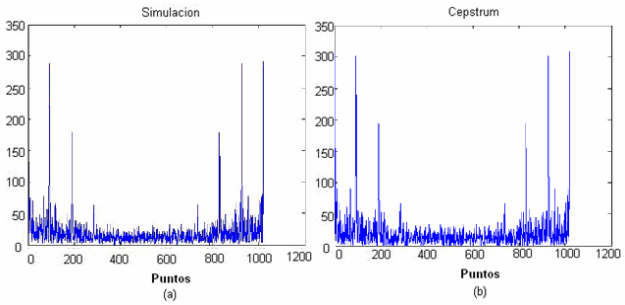

Fig. 6. Comparación entre las gráficas. (a) Resultados de la implementación. (b). Transformada Cepstrum embebida.

El sistema presenta ventajas de velocidad de procesamiento así como de costos de implementación respecto a otras plataformas de desarrollo como la TMS320 de Texas Instruments, cuya latencia de la FFT de 1024 puntos base 4 es de 3878 ciclos de máquina para el mejor caso [10]. Esto implica un mejor desempeño del core para el sistema embebido dado que además de la velocidad de procesamiento el sistema es segmentado además de las ventajas en costos, consumo y velocidad al implementar el sistema en chip (SoC) en un FPGA.

5. CONCLUSIONES.

El sistema presentado genera la posibilidad de realizar aplicaciones de procesamiento de voz en tiempo real al mantener una velocidad de 19.2 kHz haciendo uso de dos módulos FFT y dos CORDIC. Al emplear un solo módulo FFT, esta velocidad se reduce a aproximadamente la mitad, dado que hay que reutilizar la FFT existente para el cálculo de la segunda FFT.

La reconfiguración parcial dinámica a pesar de ser un método relativamente complejo de implementar diseños embebidos, presenta ventajas como el reutilizamiento del hardware y el consumo de potencia, aunque este último no detallado en este documento.

Entre las desventajas que encontramos en el proceso de diseño con RPD, se encuentran la dificultad de completar su tarea la herramienta de ruteo, PAR. Esto se debe a la gran cantidad de restricciones físicas impuestas, como la localización de los bus macro (bastantes para los buses OPB y de la FFT) y las restricciones de área para los módulos.

6. REFERENCIAS.

[1] D. Childers, D. Skinner, y R. Kemerait. “The cepstrum: a guide to processing”. Proceedings of the IEEE, VOL. 65, No 10. Oct. 1977.

[2] TEXAS INSTRUMENTS. “DSP Applications with TMS320”, Radix 4 Decimation-in-Frequency FFT Algorithm Description 1986

[3] PARHAMI, Behrooz. Algorithms and Hardware designs. Oxford: Oxford University Press. 2000

[4] ANDRAKA, Ray. A survey of CORDIC algorithms for FPGA based computers. Proceedings of the 1998 ACM/SIGDA sixth

international symposium on Field programmable gate arrays. [En línea]. 1998. Disponible en www.fpga-guru.com/files/crdcsrvy.pdf
[5] PROAKIS, J. Manolakis D. Tratamiento digital de señales. Madrid, Prentice Hall. 1997.
[6] Bobda C. et al. Designing Partial and Dynamically Reconfigurable Applications on Xilinx Virtex II FPGAs using Handle C. University of Erlagen Nuremberg.
[7] Mesquita D. et al. Remote and partial reconfiguration of FPGAs: tools and trends. IEEE 2003.
[8] XAPP 290. Xilinx Two flows for partial reconfiguration: Module based or difference based.
[9] UG 208. Xilinx Early Access Partial Reconfiguration User Guide. for ISE 8.1.01i. 2006.
[10] Texas Instruments TMS320 datasheet and TMS320 Benchmarks. www.ti.com.