Realización para un DSP de Aplicaciones en Tiempo Real

Ricardo Alzate Castaño*
Germán Castellanos Domínguez**

RESUMEN

Se plantea una metodología para el diseño de sistemas en tiempo real, orientados hacia la ejecución final en una plataforma de Hardware (DSP), con base en la descomposición jerárquica de un problema a manera de etapas constitutivas fundamentales, donde el último nivel (etapa base), se desarrolla aplicando técnicas de procesamiento digital de señales, que buscan implementar los algoritmos adecuados para facilitar la obtención del objetivo planteado. Se realiza como ejemplo ilustrativo, la estimación de la frecuencia fundamental (pitch), en una señal de voz, donde los resultados obtenidos en términos de tiempos de proceso y valores calculados, se

* Ingeniero Electrónico - Profesor Universidad Nacional de Colombia, Manizales, Colombia.
E-mail:ricardoalz@hotmail.com

** Ph.D. Profesor Universidad Nacional de Colombia, Manizales, Colombia
E-mail:gcastell@ieee.org

Fecha de recepción: Marzo 16 de 2004
Fecha de aprobación: Agosto 26 de 2004

consideran satisfactorios respecto a modelos de referencia. Aunque se acepta la posibilidad de complementar dicho procedimiento con rutinas de optimización, que maximicen el aprovechamiento para los recursos potenciales del sistema.

Palabras Clave: Algoritmos, descomposición jerárquica, DSP, metodología, optimización, tiempo real.

ABSTRACT

A methodology for real-time systems design is proposed, focusing on their execution on Hardware platforms (DSP), based on hierarchical understanding of problems using fundamental building stages, in which the lowest level (the foundation stage), is developed by applying digital signal processing techniques in order to implement the algorithms and to accomplish the objective. An illustrative example, consisting of getting the fundamental frequency (pitch) from voice signals, is shown in order to explain this method, proving good performance in terms of process rate and calculated values, compared to reference models. Finally, complementary procedures for optimization is accepted for maximize the potential resources of the system.

Key Words: Algorithms, DSP, Hierarchical, Methodology, Optimization, Real Time.

1. INTRODUCCIÓN

Los sistemas en tiempo real (STR), se asocian generalmente con aplicaciones donde el número de procesadores que interactúan, al igual que la cantidad de restricciones impuestas, están en continuo crecimiento [1]. Por tanto, para dichos casos, es importante generar un equilibrio entre tareas realizadas, tiempos de proceso y resultados obtenidos, de manera que se garantice una eficiente operación de conjunto. Adicionalmente, en las últimas décadas, los dispositivos digitales con amplia escala de integración, se vienen convirtiendo en soluciones óptimas y de costo reducido, para aplicaciones con cierta envergadura [2]. Razón por la cual, fabricantes como: Texas Instruments, Motorola y Analog Devices, entre otros, han desarrollado soluciones a nivel de procesadores digitales de señal (DSP), que permiten la ejecución para tareas de cálculo complejas, con alta relación entre velocidades de proceso y capacidades de memoria, ajustables a STR. Por tanto, independiente de la tarea a realizar, se hace importante enunciar un planteamiento suficientemente claro, que permita cubrir los pormenores implicados, para condiciones extremas de ejecución, de manera que se garantice un desempeño eficiente del sistema en tiempo real, como propósito principal de un procedimiento de diseño.

Con respecto a este último punto, se presenta la implementación de una metodología para diseño de sistemas en tiempo real, desarrollada sobre un dispositivo de hardware (DSP), utilizando técnicas de procesamiento digital de señales. Aunque dicha metodología es aplicable a procedimientos en comunicaciones, control, sistemas expertos, etc, se realizará a manera de ejemplo ilustrativo, la ejecución de un proceso de análisis en bioseñales, consistente en la detección de un parámetro (frecuencia fundamental o pitch) en señales de voz [3], tal y como se describe en las secciones posteriores.

2. ESTRUCTURAS EN TIEMPO REAL

Un sistema que interactúa con el ambiente en instantes precisos, y cuyo tiempo de procesamiento sea menor o igual a la duración de una ventana de análisis (apertura), se denomina sistema en tiempo real (STR) [4]. Dichos sistemas, se utilizan en procesos que requieren un estricto cumplimiento de tiempos, alta confiabilidad, manejo intenso de datos, al igual que alto grado de predictibilidad y adaptabilidad. Los STR se han integrado a una

amplia variedad de aplicaciones que incluyen: sistemas médicos, de control, de manufactura, de robótica y de multimedia, entre otros, adquiriendo distintas formas, que van desde simples dispositivos de monitoreo y control, hasta sistemas altamente complejos y de características críticas [4]. Debido a esto, los STR demandan una alta confiabilidad, con resultados correctos, predecibles y a tiempo, incluso en presencia de fallos [5]. Luego, puede verse a un STR, como una entidad de cómputo y control, compuesta por unidades planificables (tareas), correspondientes a procedimientos que abarcan: algoritmos en control de procesos [6], ejecución de transacciones en bases de datos [7] o transmisión de paquetes en redes de comunicación [8], que al ejecutarse de manera concurrente, y tras cumplir con requerimientos estrictos para tiempos de respuesta, garantizan su correcta operación.

3. DISEÑO DE UN SISTEMA EN TIEMPO REAL

En la metodología de diseño para STR, existen una serie de pasos que deben ser tenidos en cuenta [9]:

- ☑ La primera parte del diseño de un STR consiste en identificar las restricciones (principalmente de tiempo) que deben ser satisfechas por el sistema.
- ☑ Se realiza un diagrama de bloques, indicando las partes funcionales que integrarán el STR.
- ☑ Selección de los componentes individuales, tomando en cuenta la arquitectura propuesta y respetando siempre las restricciones de tiempo que se imponen.
- ☑ Realización de un diagrama de conexiones físicas necesarias entre dispositivos, al igual que un conjunto de pruebas de escritorio (simulaciones, cálculo de tiempos internos y de respuesta, entre otras). Si las pruebas realizadas acreditan un comportamiento adecuado, se procede a la realización del prototipo, junto con pruebas reales, para conocer el desempeño final del sistema.
- ☑ Las pruebas reales (mediciones de voltaje, frecuencia, tiempos, entre otras) aplicadas al sistema bajo condiciones normales de operación, determinan si el prototipo desarrollado cumple con las especificaciones requeridas.

Por tanto, se realiza como complemento a lo enunciado anteriormente, la división modular aplicada al procedimiento de "Estimación del pitch en señales de voz", con base en el desarrollo de una metodología para especificación de sistemas en tiempo real, a manera de etapas constitutivas fundamentales [1], que cubriendo los respectivos procesos de partición [2] y análisis (o verificación [10]), permita adecuar el modelo obtenido, a una estructura de hardware disponible [11].

3.1 Formulación del Problema:

El primer paso considerado dentro del proceso de diseño efectuado sobre sistemas en tiempo real, corresponde a la definición formal para tareas a desempeñar [10], de una manera suficientemente concisa, tal que permita evitar de inicio, posibles divergencias al momento de desarrollar implementaciones y asegure desempeños eficientes, reflejados en acierto de resultados.

Luego, con respecto al caso particular considerado, dicha tarea consiste en determinar parámetros de la señal analizada, a partir de la implementación de un algoritmo, aplicando técnicas sobre procesadores digitales de señal, de manera que los resultados obtenidos tras utilizar el método de cálculo correspondiente, sean consecuentes con aquellos generados por un sistema de referencia.

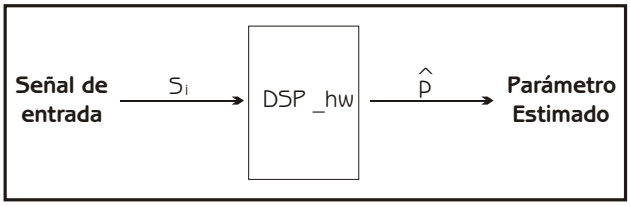

Figura 1. Nivel Superior

Lo anterior, corresponde al nivel superior, representado en la figura 1, y para el cual se genera la siguiente condición, formulada en términos de simbología matemática simple, basada en el modelo descrito por [1]:

$$<< ahora = 0 >> DSP\_hw << CDSP\_hw >>$$

$$CDSP\_hw \equiv \forall t1,t2 \in Z\{0,1...\}/t2>t1 \Rightarrow$$

$$\exists s_i\,[t1,t2], \hat{p}[t1+\Delta t,\ t2+\Delta t\ ]\ /abs(p - \hat{p})<e$$

Que enuncia lo siguiente:

- ☑ << Para el instante actual >>el bloque DSP_hw (Nivel superior) debe cumplir la condición $CDSP\_hw$>>
- ☑ Dicha condición ($CDSP\_hw$), asume que para todo tiempo $t1$ y $t2$, enteros positivos, y tal que $t1 > t2$, entonces:
- ☑ Debe existir una señal de entrada (S_i) durante el intervalo [$t1,t2$], que produzca un resultado para el parámetro en consideración ($\hat{p}$), en el intervalo [$t1+\Delta t,t2+\Delta t$] (donde Δt corresponde al tiempo de retardo generado por procesos de cálculo), y tal que dicho estimado no genere un error superior al máximo preestablecido e, respecto al valor de referencia p.

A partir de esta notación, se definirán etapas subsiguientes que corresponden con niveles inferiores, dentro de la estructura asignada para el árbol de partición (figura 5).

3.2 Primer Nivel de Particionamiento:
En éste, se considera la división inicial para el bloque de nivel superior DSP_hw , en:

- ☑ Una componente de hardware, equivalente al sistema de adquisición (DSP_codec [11]),
- ☑ Al igual que, una componente de Software, encargada de manipular la operación del anterior, a través de ejecución de instrucciones del sistema ($Sist\_soft$).

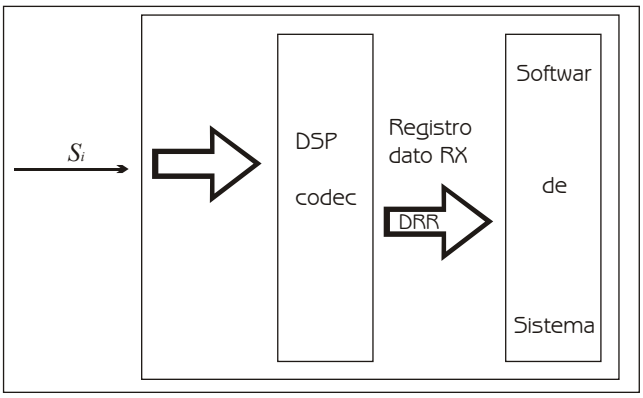

Figura 2. Primer Nivel de Particionamiento

Luego, el requerimiento a este nivel, se restringe al desarrollo del proceso de señal, con velocidad suficiente para la cantidad de muestras adquiridas.

Evitando así inconsistencias por pérdida de datos, consecuencia de muestreo por encima o por debajo del valor nominal (upsampling o downsampling, respectivamente [12]). Por tanto, se definen las siguientes condiciones sobre dichos sub-bloques:

$$<< ahora = 0 >> DSP\_codec << CDSP\_codec >>$$

$$CDSP\_codec \equiv \forall n<\infty;\tau,n_0 \in Z^+,\tau \neq 0 \Rightarrow$$

$$(DRR(n) \forall n \in [n_0,n_0+\tau] \rightarrow sample\ (s_i))$$

Inicialmente en el codec, para todo valor de tiempo discreto n, positivo y finito, debe cumplirse que la asignación transferida en cualquier instante, hacia el registro de recepción de datos (DRR), corresponda con un valor perteneciente al conjunto de muestras de señal en entrada S_i , durante un intervalo $[n_0, n_0 + \tau]$, donde n_0 equivale al instante inicial, al tiempo que τ se relaciona con el periodo de muestreo (diferente de cero).

$$<< ahora = 0 >> Sist\_soft << CSist\_soft >>$$

$$CSist_s oft \equiv \forall n \in [n_0,n_0+\tau] \Rightarrow$$

$$\exists DRR(n)/\in \{sample(s_i)\}$$

De la misma manera, el software de sistema, requiere que durante el intervalo $[n_0\ , n_0+\tau_0]$, el valor proveniente desde el registro de recepción de datos (*DRR*), pertenezca a una de las muestras de señal en entrada S_i.

Las premisas anteriores buscan establecer la correcta sincronización entre los diferentes eventos implicados en el procedimiento de adquisición [1].

3.3 Segundo Nivel de Particionamiento:

Para esta etapa, se asume una nueva subdivisión, esta vez aplicada sobre el bloque, software de sistema (*Sist_soft*), en:

- ☑ Rutina para servicio de interrupción, y
- ☑ Receptor de puerto serial multicanal (McBSP [15]).

Seguidamente, se realiza el acople entre recursos de hardware (codec, como dispositivo de E/S), y aplicaciones a nivel de software para proceso de datos, haciendo uso de una entidad de almacenamiento (*Buffer*), definida como ventana de longitud determinada por la apertura de proceso.

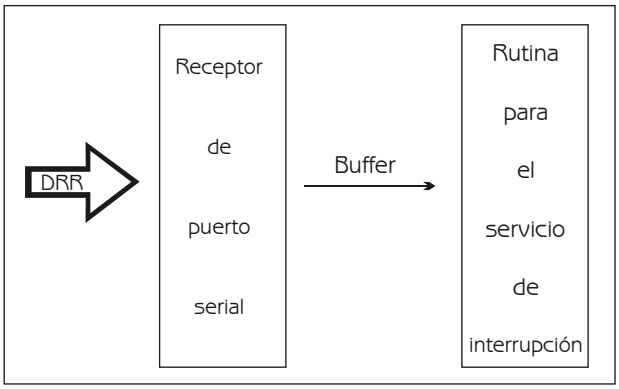

Figura 3. Segundo Nivel de Particionamiento

Por consiguiente, la condición a cumplir, se basa en lograr que la rutina se ejecute completamente, antes que otro conjunto de datos se encuentre disponible para realizar un nuevo proceso de señal.

$$<< \textbf{ahora} = 0 >> \textbf{Rxsp} <<\textbf{CRxsp}>>$$

$$CRxsp \equiv \forall n=0,1...,N-1; m=1,1...,M \Rightarrow$$

$$if(n=N) \Rightarrow (Buffer(m)=DRR(n\tau))$$

En primera instancia, es necesario activar el servicio de interrupción, cada vez que se complete una trama, es decir, al conformarse una palabra de N bits en el registro de recepción de datos (*DRR*). A partir de lo cual, deberá asignarse su contenido, en la posición respectiva para un buffer de longitud M.

$$<< \textbf{ahora} = 0 >> \textbf{Isr} <<\textbf{CIsr}>>$$

$$\textbf{CIsr} \equiv \textbf{ifIsr (Buffer)} \in [t_0, t_0+\Delta t] \Rightarrow$$

$$M\tau\Delta t$$

En la misma medida, debe garantizarse que la rutina a ejecutar (*Isr*), sobre el conjunto de datos en entrada (*Buffer*) durante el intervalo $[n_0, n_0+\Delta t]$, no emplee un tiempo de cálculo Δt, superior al máximo valor permitido ($M\tau$).

3.4 Tercer Nivel de Particionamiento:

Finalmente, se hará uso de información disponible acerca del estado del *Buffer* destinado a la entrada de datos. Es decir, se aplica una nueva partición sobre la rutina de servicio a interrupción.

Por tanto, el proceso de señal (para este caso correspondiente a determinación de la frecuencia fundamental), se llevará a cabo sólo cuando la condición sobre la bandera asignada, sea la requerida para proceder.

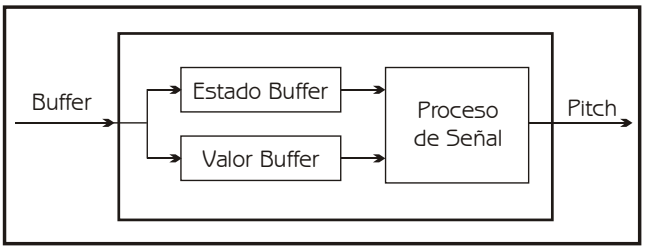

Figura 4. Tercer Nivel de Particionamiento

$<< ahora = 0 >> Signal\_proc <<CSignal\_proc>>$

$CSignal\_proc \equiv if\ (m = M) \Rightarrow$

$dato = [buffer(1), buffer(2),..., buffer(M)]$

$\hat{p} = Signal\_proc(dato)$

Luego, al detectarse un marco completo de entrada ([*buffer(1), buffer(2)...buffer(M)*]), se realiza sobre el mismo (*dato*), un correspondiente proceso de señal (*Signal_Proc*), que asigna como salida de sistema, el valor equivalente al parámetro en consideración ($\hat{p}$) para el segmento de señal analizado.

3.5 Árbol de Partición:

Con base en las condiciones previamente establecidas, es posible construir, para efectos de representación visual, respecto a los diferentes niveles considerados, un diagrama o *árbol de partición* [2], como el visualizado en la figura 5.

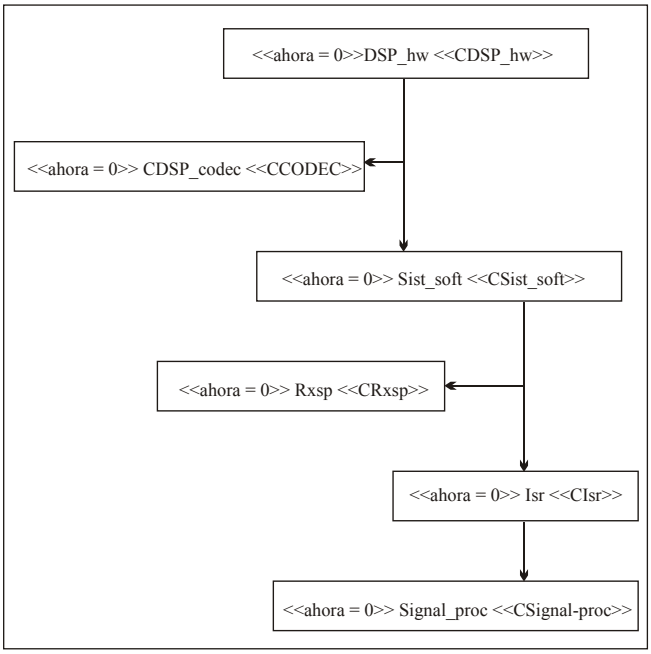

Figura 5. Diagrama para el árbol de Partición

3.6 Verificación:

Por último, se confrontará la validez para el procedimiento desarrollado, verificando el cumplimiento de las premisas generadas, tras aplicar restricciones en cuanto a recursos disponibles para llevar a cabo la implementación final [1].

Luego, a partir de especificaciones técnicas concernientes al sistema de procesamiento empleado (TMS320C6701 [11]), y a características de señal manipulada (paquetes muestreados a 10 kHz), dichas restricciones corresponden a:

- ☑ *dt*, referido a retardos por propagación o conmutación, y que se consideran despreciables.
- ☑ τ, periodo de muestreo, con un valor equivalente de 0.1 ms.
- ☑ Memoria de programa, que no debe superar los 512 KB.
- ☑ N, longitud para palabra adquirida en el codec del dispositivo de hardware, correspondiente a 32 bits para máxima resolución.
- ☑ e, máximo error porcentual permitido en valor estimado, asumido para el 5%.
- ☑ Finalmente, la apertura de proceso o número de muestras por ventana, que se toma como de 128 elementos.

Como resultado, el sistema propuesto a partir de esta metodología de especificación, al operar sobre la plataforma de hardware descrita [11], y bajo las condiciones de operación precisadas, permitirá obtener en su salida, un valor estimado ($\hat{p}$) correspondiente al parámetro en consideración (frecuencia fundamental - pitch), tras un retardo (Δt) equivalente al empleado en ejecutar el bloque para proceso de señal (*Signal_proc*), que no debe superar un periodo de 12.8 *ms*. , para señales con ancho de banda inferior a 5 *kHz*.

4. IMPLEMENTACIÓN DEL SISTEMA

Utilizando como referencia el algoritmo descrito en [9], se realizó la ejecución sobre un procesador digital de señales [11], para el procedimiento de estimación del pitch en señales de voz, haciendo uso de la herramienta de software "*Code*

Composer Studio" [15], creada para interactuar con tales dispositivos.

Dicha interfaz (figura 6), se desarrolla sobre una plataforma operativa *Windows*, al tiempo que utiliza como lenguaje de base, el conjunto de comandos *Ansi* C, permitiendo desarrollar aplicaciones en un entorno más amigable, que los convencionales, basados en lenguaje ensamblador [16].

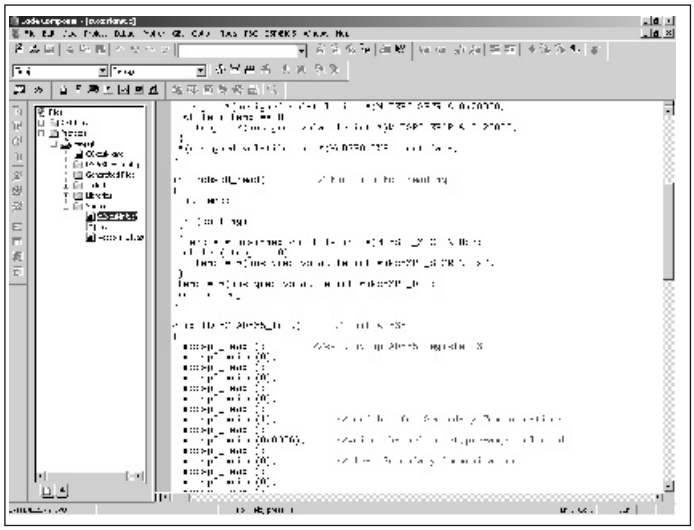

Figura 6. Ventana para ejecución de código

El algoritmo implementado contiene la siguiente secuencia de etapas (figura 7):

- ☑ En primera instancia, es necesario adecuar el sistema para su posterior ejecución, a partir de procedimientos de inicialización y configuración, aplicados sobre aquellos elementos implicados en desarrollo del proceso, correspondientes a los módulos: codificador/decodificador de audio (*Codec*), y puerto serial multicanal (*McBSP*). Para terminar en un ciclo infinito, en espera de ser anticipado por procesos de mayor prioridad (interrupciones). Todo lo anterior incluido en una rutina principal, denominada *Main*.
- ☑ Cada vez que el registro de recepción de datos (*DRR*), activa un servicio de interrupción desde el *McBSP*, el ciclo infinito en *Main* se interrumpe para dar paso a las instrucciones contenidas en la función *Hookint*, efectuando: inicialización en vector de interrupciones, mapeo entre la interrupción 15 del procesador utilizado y la petición del *DRR*, activación de interrupción y asignación para rutina de proceso (*McBSPcvISR*).

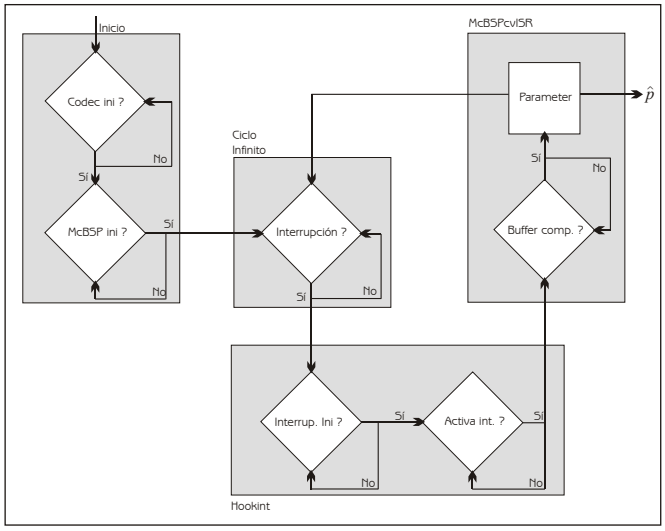

Figura 7. Algoritmo implementado

- ☑ Por último, cuando la petición de interrupción *Hookint*, asigna ejecución para el procedimiento *McBSPcvISR*, se realiza lectura del valor proveniente desde el puerto de entrada, equivalente a una muestra de señal actual, asignada posteriormente a la posición respectiva en el buffer de datos, que luego de alcanzar la cantidad de muestras considerada por ventana (*M=128*), permite ejecución para la rutina de cálculo, contenida en la función *parameter*.

Dicha función, realiza estimación para el parámetro requerido, tal y como se describe a continuación.

5. DESARROLLO DE ALGORITMOS

La ejecución para los algoritmos implementados, se obtuvo a partir de las siguientes fases (figura 8):

- ☑ Ejecución de códigos en un entorno de programación simulado (empleando Matlab), aplicado sobre señales de prueba, correspondientes a fonemas vocálicos segmentados pertenecientes a la base de datos MIRLA (propiedad de la Universidad Nacional de Colombia, sede Manizales).

- ☑ Realización de comparación, en términos de desempeño, sobre resultados obtenidos por un sistema de referencia: Algoritmo Childers [17], como método de validación respecto a contornos del pitch.

- ☑ Implementación de algoritmos, en código orientado al procesador digital de señales disponible (referencia TMS320C6701 [18], [15]).

- ☑ Adquisición de señales analógicas vía puertos, para crear un intercambio de datos entre las tarjetas de procesador (DSP) y de audio (PC).

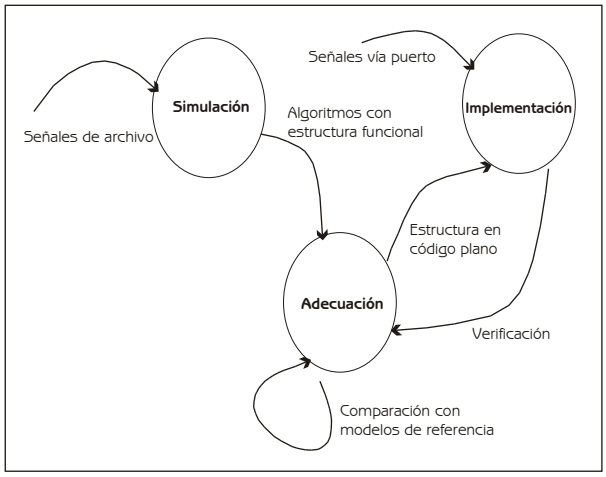

Figura 8. Diagrama de flujo de datos (DFG), para metodología

5.1 Algoritmo para Cálculo del Pitch:

Haciendo uso de un método basado en análisis de dominio frecuencial (HPS - Harmonic Product Spectrum [19]), se desarrolla estimación para la frecuencia fundamental en señales de voz [3], a partir de los siguientes pasos (figura 9):

- ☑ Ventaneo tipo Hanning, con traslape de 64 muestras, en el segmento de señal adquirido (cuya apertura es de 128 datos), para reducir sensibilidad a los cambios presentados en extremos de ventana.

- ☑ Posteriormente, se aplica transformada rápida de fourier (FFT), con resolución de 2048 puntos, sobre dicho segmento.

- ☑ Luego, se define un vector de posiciones para candidatos a frecuencias sub-armónicas, conformado por aquellos valores superiores al 30% del mayor armónico obtenido.

- ☑ Finalmente, se ejecuta un juicio de decisión, respecto a las distancias relativas entre posiciones para tales armónicos, de cuyo máximo común divisor, se obtiene el valor de periodo fundamental.

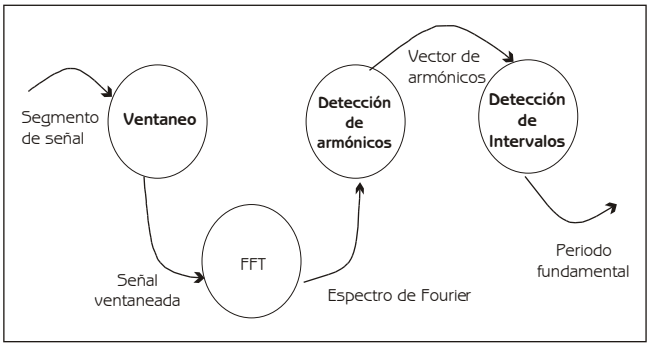

Figura 9. Algoritmo para cálculo de Picht

6. RESULTADOS

La implementación a nivel de hardware (DSP), para los procedimientos anteriores, permitió obtener lo siguiente:

6.1 Cálculo del Pitch:

El algoritmo para detección de la frecuencia fundamental en señales de voz, genera una envolvente con valor promedio muy cercano al patrón de referencia, como se observa en la figura 10, tomada para un segmento de voz equivalente al fonema '*a*', con valor promedio de 227 Hz y tasa de muestreo de 10kHz. Resultados adicionales, respecto a valores promedio obtenidos sobre otras señales de prueba, se visualizan en la tabla 1.

Análogamente, se cita información acerca de indicadores de desempeño para el sistema en tiempo real, con base en especificaciones técnicas agrupadas en la tabla 2.

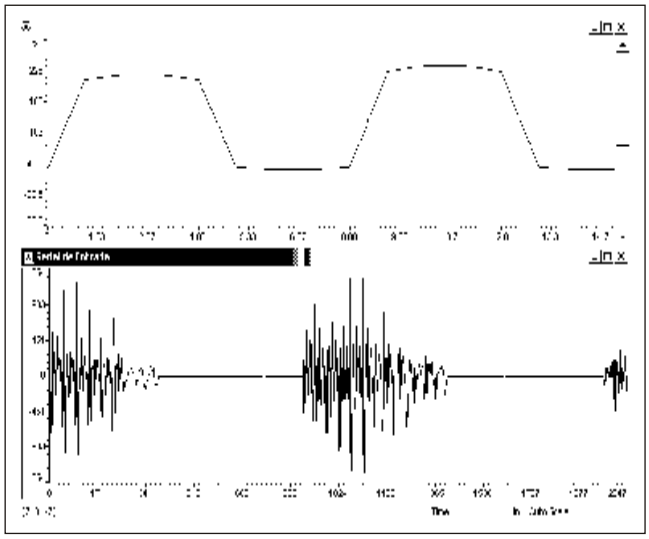

Figura 10. Señal de entrada y contorno respectivo obtenido en DSP

7. CONCLUSIONES

Con base en los resultados obtenidos, puede concluirse que:

- ☑ La metodología propuesta, permite ejecutar de manera directa y eficiente, la estimación en línea del parámetro requerido en la señal analizada, facilitando implementación de tareas en tiempo real, con base en el uso de herramientas tanto de hardware, como de software, especializadas para tal fin.
- ☑ En la misma medida, se obtuvo una respuesta satisfactoria en cuanto a valores calculados respecto a los modelos de referencia.
- ☑ Respecto a velocidades de proceso, se evidencia un promedio superior al calculado en el diseño para tiempo real, presentando un desempeño que basado en resultados con bajo porcentaje de error, se podría considerar apenas justo, aunque se admite un complemento necesario por parte de rutinas de optimización que permitan maximizar la capacidad de aprovechamiento sobre los recursos potenciales del sistema.
- ☑ Luego, puede considerarse a éste, como un primer paso dentro de la realización de sistemas más robustos, que incluyan conjuntos ampliados de características, enfocadas hacia la ejecución de tareas particulares desarrolladas en diferentes áreas tecnológicas que se aplican a diario en nuestro país.

Tabla 1. Error porcentual para diferentes señales de prueba

Tono de calibración (Hz)	Promedio medido (Hz)	Error porcentual (%)
100	103	3
125	131	4,8
250	253	1,2
300	304	1,3
500	501	0,2
1000	1003	0,3
2000	2001	0,05
Señal Referencia Praat (Hz)	**Promedio medido (Hz)**	**Error porcentual (%)**
227	220	-3,08
114	126	10,52
171	186	8,77
210	200	-4,76
233	245	5,15
199	196	-1,5
194	201	-3,6
144	159	10,41
140	131	-6,42

Tabla 2. Indicadores de desempeño para sistema implementado

Criterio	Valor
Tiempo de ciclo por instrucción	40 ns
Duración promedio para rutina de proceso	435 ms
Longitud promedio para rutina de proceso	6884 Bytes
Duración promedio para rutina de adquisición	6.52 us
Longitud promedio para rutina de adquisición	2160 Bytes
Nivel de optimización	para máxima velocidad

8. REFERENCIAS

[1]M. Joseph, "Real-Time Systems: Specification, Verification, and Analysis". p 98 - 122. Prentice Hall. New York. 1996.

[2] A. Jhon, "Real-Time Signal Processing: The design and implementation of signal processing systems". p 151 - 240. Prentice Hall. New York. 1999.

[3] A. Ricardo, C. Germán, "Estimación de contornos del pitch en línea sobre DSP". Tesis de Pregrado. Universidad Nacional de Colombia - Sede Manizales. 2003.

[4] P. D. Lawrence, K. Maunch, "Real-Time Microcomputer System Design". McGraw Hill, New York, 1987.

[5] J. Lala, R. Harper, "Architectural Principles for Safety-Critical Real-Time Application". Proceedings of the IEEE, 1994.

[6] K. G. Shin, H. Kim, "Derivation and application of hard deadlines for Real-Time Control Systems". IEEE Transactions on Systems, Manufacturing, and Cybernetics, Vol 22, November 1992.

[7] K. G. Shin, P. Ramanathan, "Real-Time Computing: A New Discipline of Computer Science and Engineering". Proceedings of the IEEE, Vol 82, 1994.

[8] J. Stankovic, K. Ramanritham, "Tutorial: Hard Real-Time Systems". IEEE Computer Society, 1988.

[9] A. L. Chau, J. J. Medel, P. Guevara, "Sistemas Dedicados de Tiempo-Real". http://aleph.cs.buap.mx.

[10]B. Stuart, "Real-Time Computer Control: An introduction". p 129 - 172. Prentice Hall. New York. 1988.

[11]Texas Instruments, "TMS320C6201/6701 Evaluation Module Technical Reference".

[12]M. Sanjit K., Digital Signal Processing: A Computer-Based Approachp 499 - 647. Mac Graw Hill.

[13]Texas Instruments, TMS320C600 Peripherals Reference Guide.

[14]k. Hoover, TMS320C62x/C67x C Familiarization and Audio Sampling.Rose-Hulman Institute of Technology.

[15]Texas Instruments, Code Composer Studio Getting Started Guide.

[16]N. Seshan, DSPs vs. FGGAs in Signal Processing System Desing.Texas Instruments.

[17]C. G. Donald, Speech Processing and Synthesis Toolboxes. p 320-324. John Wiley & Sons, Inc. New York. USA. 2000.

[18]Texas Instruments, TMS320C6201/6701 Evaluation Module User's Guide.

[19]Q. Holger, S. Olaf, S. Manfred R., Robust pitch Tracking in the Car Environment. Drittes Physikalisches Institute, Universitat Gottingen and Robert Bosch Research & Development DaimlerChrysler Research and Technology AT&T Bell Labs.