EXPERIMENTOS DE LABORATORIO PARA PROCESAMIENTO DIGITAL DE SEÑALES EN TIEMPO REAL

David Augusto Rojas Vigo

Facultad de Ingeniería Electrónica, Universidad Nacional Mayor de San Marcos

RESUMEN: El artículo que se presenta intenta difundir a otras instituciones la manera por la cual se podría actualizar los tópicos del área de procesamiento digital de señales en tiempo real para los estudiantes de ingeniería electrónica y ramas afines. Para ello, se describe cuatro experimentos desarrollados con el DSP Starter Kit TMS320C6711, los cuales dan la base a los estudiantes para enfrentar con éxito a otros temas más avanzados. También, se hace una revisión de los artículos de destacados autores quienes se han interesado por el tema.

ABSTRACT: The paper that is presented tries to diffuse to other institutions the way for which it could modernize the topics of Digital Signal Processing in real time area for the students of electronic engineering and similar branches. For it, four experiments developed with the DSP Starter Kit TMS320C6711 is described, which give the base to the students to face with success other more advanced topics. Also, a revision of the papers of outstanding authors is made, which have been interested in the topic.

Palabras Clave: Laboratorio de procesamiento digital de señales, PDS, procesamiento en tiempo real, diseño y simulación de Procesamiento Digital de Señales con MATLAB.

I. INTRODUCCIÓN

El procesamiento digital de señales en tiempo real se ha convertido en un importante tema de la educación en ingeniería, debido a la creciente demanda de los productos relacionados con las comunicaciones y la multimedia que utilizan tecnologías DSP. Esto ha motivado que muchas universidades estén en constante desarrollo de metodologías renovadas de enseñanza sobre algoritmos, implementaciones y aplicaciones a sus estudiantes e ingenieros para enfrentar los desafíos de la industria.[1]

Por este motivo, se presenta una breve descripción del contenido de los experimentos que he desarrollado en el curso de laboratorio de Procesamiento Digital de Señales para estudiantes de pregrado de la Facultad de Ingeniería Electrónica en la Universidad Nacional Mayor de San Marcos con el fin de dar un aporte al uso de metodologías asociadas con herramientas de desarrollo de hardware y software que enriquezcan la enseñanza y ofrezcan una visión renovada de la electrónica a los estudiantes de nuestra institución. La metodología adoptada pone énfasis no solo en la simulación realizada con herramientas de Matlab sino también en la implementación a través del *DSP Starter Kit TMS320C6711*.

El entorno MATLAB es una herramienta muy útil de aprendizaje para la educación en procesamiento digital de señales, porque ofrece al estudiante una rápida transición de los conceptos teóricos a su aplicación real en forma experimental.[2] Sin embargo, la implementación de algoritmos en tiempo real requiere un conocimiento básico acerca del hardware con el que se va a trabajar.

En la etapa de diseño, es necesario utilizar el entorno visual de MATLAB para probar y desarrollar nuevos algoritmos, depurarlos y luego verificar su funcionamiento. No obstante, la implementación en un computador en condiciones normalmente ideales dista de su implementación en un hardware específico, pues debe tenerse en cuenta algunas consideraciones tales como por ejemplo, aritmética finita, tiempo de procesamien-

to, características de los periféricos, entre otras, que se presentan durante la etapa de implementación.

La metodología descrita brevemente en este artículo permite mostrar la manera en que los algoritmos matemáticos de procesamiento digital de señales pueden ser codificados para su implementación en un hardware programable (DSP), logrando complementar adecuadamente la teoría y capturando la atención del estudiante para que, aplicando los conceptos y las herramientas referidas pueda resolver problemas reales más complejos.

Los temas tratados y experimentados presentan la siguiente secuencia:

- Experimento No.1: El *kit* de desarrollo DSP TMS320C6711 y la codificación para tiempo real.
- Experimento No.2: Generación de señales en tiempo discreto.
- Experimento No.3: Diseño e implementación de filtros digitales básicos (FIR e IIR).
- Experimento No.4: Análisis espectral utilizando la trasformada discreta de Fourier (DFT).

Seguidamente se detallan en forma breve estos temas y en el anexo se muestra algunos de los resultados experimentales.

II. EXPERIMENTO No 1: EL KIT DE DESARROLLO TMS320C6711

El *DSP Starter Kit TMS320C6711* es una herramienta de desarrollo de Texas Instruments (TI) que incluye el hardware (tarjeta DSK) y el software (Code Composer Studio) para procesamiento de señales en tiempo real.

2.1 Tarjeta DSK

La tarjeta DSK incluye un procesador digital de señales, un reloj de 150 MHz para la unidad central de procesamiento (CPU) que le permite realizar 900 millones de operaciones de punto flotante por segundo (MFLOPS), dos reguladores de tensión que suministran 1.8V para el núcleo y 3.3V para la memoria y periféricos, dos memorias dinámicas de acceso aleatorio síncronas de 4MB x 32 bits (SDRAM) y una memoria de sólo lectura (ROM) flash de 128 kB. Contiene además, un codec de 16 bits, que trabaja con una frecuencia fija de 8kHz para la entrada y salida de señales analógicas, accesible a través de dos conectores de audio. La conexión entre la tarjeta DSK y el computador es a través del puerto paralelo estándar. La figura 1, muestra el diagrama de bloques simplificado.

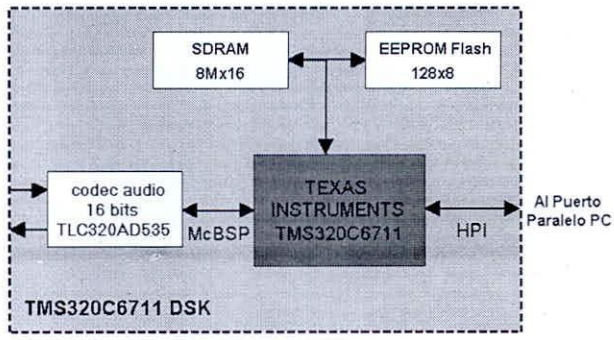

Figura 1. Diagrama de bloques de la tarjeta DSK.

2.1.1 Arquitectura del procesador TMS320C6711

El TMS320C6711 pertenece a la familia C6000 de TI, y está basado en una arquitectura avanzada de palabra de instrucción muy larga (VLIW) capaz de ejecutar más de ocho instrucciones por ciclo. El procesador ha sido diseñado tomando en cuenta la eficiencia del compilador C. Sin embargo, es posible que con un programa escrito en lenguaje C se pueda ejecutar entre el 80 y 90% de la velocidad de procesamiento, comparado con uno escrito en lenguaje *assembler*.[3]

En la Figura 2, se ilustra el circuito interno del DSP, contiene una CPU que realiza operaciones en punto flotante, una memoria interna, un controlador de acceso directo a memoria mejorado (EDMA), periféricos que incluyen una interfaz de memoria externa de 32 bits (EMIF), dos puertos seriales multi-canal con búfer (McBSP), dos temporizadores de 32 bits, un puerto interfaz para *host* de 16 bits (HPI), un selector de interrupciones y un lazo de bloqueo de fase (PLL), entre otros.[4]

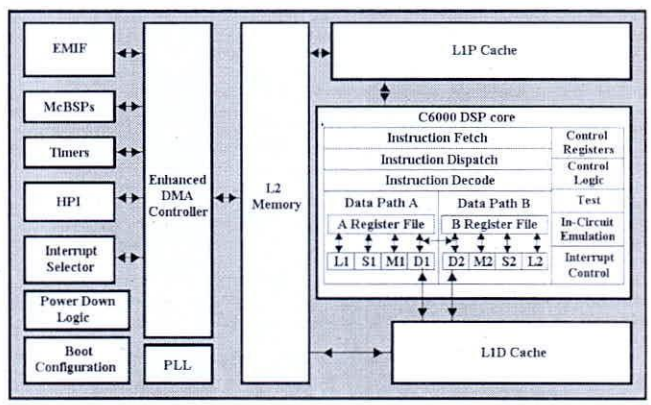

Figura 2. Diagrama de bloques del TMS320C6711.

2.1.2 Interrupciones

Una interrupción ocurre cuando un evento asíncrono o múltiples eventos también asíncronos requieren de un DSP para procesar tareas. Cuando ocurre una interrupción se detiene la CPU, los estados del proceso actual son almacenados en registros, permitiendo la recuperación de éstos posteriormente después de completada las tareas solicitadas por la interrupción. Puede haber fuentes internas o externas de interrupción, así como un conversor análogo-digital, un temporizador u otros periféricos. Su estructura interna permite tres tipos de interrupciones:

- de reiniciación,
- no enmascarables (NMI) y
- enmascarables.

2.2 El Code Composer Studio (CCS)

El CCS es una herramienta software de soporte que suministra un entorno de desarrollo integrado (IDE) semejante al presentado por Visual C++, el cual incluye un compilador C/C++, un ensamblador, un enlazador, un depurador, así como otros utilitarios. También soporta procesamiento, análisis, planificación e intercambio de datos en tiempo real.

El compilador C/C++ del CCS permite la construcción de programas en lenguaje C estándar dentro del lenguaje *assembler* C6000 usando un sofisticado proceso de optimización, que le permite generar un código compacto y eficiente. El ensamblador crea un archivo objeto en lenguaje máquina a partir del código fuente en *assembler*. Luego a partir del archivo enlazado, se produce un archivo ejecutable el cual luego es cargado y ejecutado en el DSP combinando archivos objeto y librerías. El depurador permite visualizar los datos gráficamente, tal que el usuario pueda hallar y corregir los errores rápida y eficientemente. El DSP/BIOS es un *kernel* de tiempo real escalable, diseñado para realizar tareas tales como: planificación, sincronización y comunicación con el *host*, realizando el análisis de un programa sin detener la ejecución del procesador.

El CCS retiene toda la información de una aplicación en un archivo de proyecto (con extensión *pjt*) almacenando el nombre de los archivos con código fuente, librería de objetos, opciones de las herramientas que generan código y nombre de los archivos de cabeceras (con extensión *h*). Cuando se desarrolla un proyecto con el CCS por lo menos necesita ser preparado por el programador los archivos de código fuente (con extensión *c*), un archivo fuente que defina los vectores de interrupción o de re-inicialización en *assembler* (con extensión *asm*) y un archivo de comandos del enlazador (con extensión *cmd*). El CCS también permite manipular diferentes opciones de compilación, donde una de ellas admite que las operaciones de punto flotante sean hechas a nivel de hardware, ya sea con unidades de punto fijo o punto flotante.

2.3 Codificación para proceso en tiempo real

El término "tiempo real" en un sistema con DSP se refiere a que el procesamiento de una muestra de la señal de entrada ocurra durante el periodo de muestreo.[5] En la figura 3, se observa que durante el intervalo de tiempo *n* y *n+1* la captura de una muestra en el instante *n* el cual es procesada para generar el nuevo valor de salida (tiempo de procesamiento) y luego queda en espera de la siguiente muestra en *n*+1 (tiempo de espera).

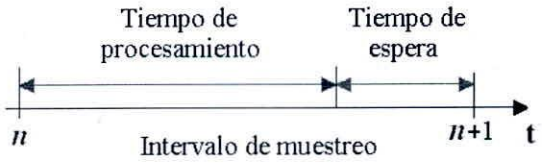

Figura 3. Procesamiento en tiempo real.

La codificación en tiempo real a través del DSP TMS320C6711 requiere una estructura básica de programación, basada en el manejo de una interrupción denominada *int11*, que se repite periódicamente cada 125 mseg. y determina el periodo de muestreo de la señal de entrada. Al recibirse una interrupción, el procesador salta a la rutina de servicio de interrupción (ISR) y la ejecuta, este tiempo determina el tiempo de procesamiento. Seguidamente retorna a la rutina principal y la ejecución se detiene hasta que se produzca una nueva interrupción, este tiempo se denomina el tiempo de espera. El siguiente código fuente en lenguaje C muestra la realización de este esquema en el CCS.

```
#include "LPF2500.cof"
short x[N];
short yn;
interrupt void c_int11()
{
 x[i]=input_sample();  // captura
 ...                   // procesamiento
 output_sample(yn);    // generación
```

```
return;
}
void main()
{
 comm_intr();
 while(1);
}
```

En la rutina main, comm_intr inicializa el *codec*, los puertos McBSP habilita la interrupción, la instrucción while detiene la ejecución y espera a que se produzca una interrupción. La ISR llamada c_int11, input_sample, captura la muestra actual de la señal de entrada que entrega el conversor análogo-digital, luego ésta señal digital es procesada y output_sample envía la muestra de salida hacia el conversor digital-análogo del *codec*.

Adicionalmente, se incluye un archivo cabecera (con extensión *cof*) utilizando la directiva include. Este archivo establece los parámetros (valores iniciales y constantes) del procesamiento. Por consiguiente, para cambiar sus características, sólo es necesario calcular los nuevos parámetros e incluirlos en esta cabecera. Las funciones input_sample, output_sample y comm_intr, no son parte de las funciones estándar del CCS, sino están definidas en los archivos de soporte generados por Rulph Chassaing.[6]

III. EXPERIMENTO No. 2: GENERACIÓN DE SEÑALES

Las señales generadas con osciladores digitales juegan un rol importante en los sistemas basados en DSP's, como por ejemplo para generar funciones en el diseño de los sistemas de comunicación.[7]

Los osciladores sinusoidales por ejemplo pueden ser vistos como una forma de un circuito resonante de dos polos, donde los polos conjugados complejos están en el borde del círculo unitario. En consecuencia se puede obtener una salida senoidal calculando un caso especial de un filtro digital IIR de segundo orden basado en la ecuación de diferencias 1.

$$y[n]=B_1*x[n-1]+A_1*y[n-1]+A_0*y[n-2] \quad (1)$$

donde $B_1=1$, $A_0=-1$, $A_1=2\cos(\theta)$ y $x[n]$ es la función delta. Debido a la naturaleza recursiva del algoritmo, la precisión es crucial para prevenir la acumulación del error, por ello se debe utilizar instrucciones de punto flotante para implementar el generador de señal senoidal.

Las operaciones más críticas para codificar la ecuación de diferencias son la multiplicación y acumulación de la señal y los coeficientes. El objetivo fue implementar un generador de onda senoidal de frecuencia arbitraria, por lo que este requiere que el DSP envíe periódicamente las muestras al *codec* de la tarjeta DSK.

3.1 Procedimiento experimental

- Simular el oscilador sinusoidal basado en el modelo matemático propuesto por la ecuación de diferencias utilizando MATLAB.
- Codificar en el CCS el modelo de generador senoidal para 1kHz, ejecutar y comprobar los parámetros utilizando un osciloscopio y un analizador de espectro.
- Codificar el modelo de generador senoidal propuesto por medio de tablas de búsqueda y generar una forma de onda senoidal en la salida.

Finalmente, como complemento al experimento y los conceptos utilizados se propuso las siguientes tareas:

- Codificar un generador de onda cuadrada cuya frecuencia es fija y con ciclo de trabajo del 50%.
- Simular en MATLAB y seguidamente codificar utilizando el CCS una señal modulada en amplitud con una señal senoidal de 200 Hz y una señal portadora de 1kHz.
- Describir las ventajas y limitaciones de las formas de codificación basadas en modelos y tablas.
- Describir las limitaciones de la tarjeta DSK para la generación de señales.

IV. EXPERIMENTO No.3: FILTROS DIGITALES

Una tecnología ya consolidada para la implementación de filtros digitales es el uso de los DSP's, pues estos poseen una arquitectura adecuada para la realización de las operaciones de multiplicación y acumulación de productos en un tiempo muy reducido.

Las técnicas de diseño de filtros digitales en general son utilizadas para determinar el conjunto de coeficientes o la respuesta impulsional que se aproxime a la respuesta en frecuencia u otras especificaciones del filtro deseado. Por lo tanto, es importante tener en cuenta que debido a la diferencia que existe en la estructura de los filtros FIR e IIR, las técnicas de diseño difieren considerablemente.[8]

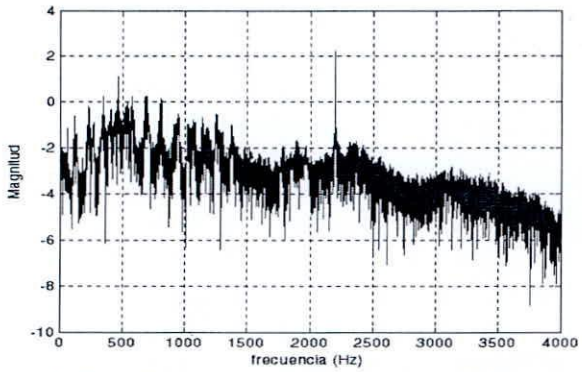

Figura 4. Espectro en escala logarítmica de la señal de voz con un tono de fondo de 2.2 kHz.

Para experimentar la respuesta del filtro se grabó aproximadamente tres segundos de una señal de voz con un tono de fondo de 2.2 kHz, cuyas componentes en frecuencia son mostradas en la figura 4. El objetivo fue eliminar el tono de señalización a través de un filtro supresor de banda FIR y un IIR con la tarjeta DSK.[9]

4.1 Filtros FIR

Entre las principales técnicas de diseño de los filtros FIR de fase lineal, se tienen: técnica de las series de Fourier, técnica de las ventanas, técnica del muestreo en frecuencia, las basadas en técnicas de optimización, entre otras.[10]

4.1.1 Procedimiento experimental

- Utilizar las funciones del *Signal Processing Toolbox* de MATLAB para diseñar por mínimos cuadrados un filtro FIR antipasabanda de orden 36, frecuencia de muestreo de 8kHz y con las características mostradas en la figura 5.[11]
- Almacenar en un archivo los coeficientes y el orden del filtro diseñado y utilizarlo para su codificación en el CCS.
- Conectar un generador de funciones a la entrada de la tarjeta DSK donde se está ejecutando el filtro FIR propuesto, verificar la respuesta del filtro evaluando la respuesta de la señal de salida para diferentes frecuencias utilizando un osciloscopio o un analizador de espectro.
- Aumentar el orden del filtro diseñado a 100 y analice su comportamiento respecto al anterior.
- Describir el efecto producido por el cambio en el orden del filtro con respecto a la respuesta deseada.
- Comparar los requerimientos de ambas implementaciones.

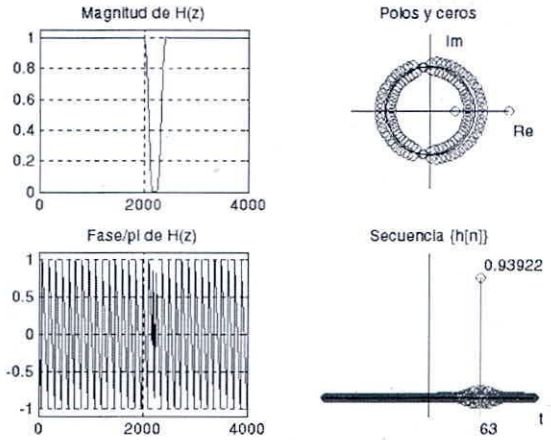

Figura 5. Características del Filtro FIR.

4.2 Filtros IIR

En este caso las principales técnicas de diseño de filtros IIR utilizadas, se tienen: técnica a partir de filtros analógicos, técnica por aproximación de derivadas, técnica de la transformada bilineal, entre otras.[12]

4.2.1 Procedimiento experimental

- Diseñar un filtro IIR antipasabanda utilizando el entorno interactivo de procesamiento de señales SPTOOL. En tal sentido, tal como se muestra en la figura 6, definir los siguientes parámetros: la frecuencia de muestreo, el algoritmo de diseño, el tipo de filtro (de acuerdo a sus características de frecuencia), la frecuencia de borde pasa banda inferior (*fp1*) y superior (*fp2*), el máximo rizado (*Rp*), la frecuencia de borde supresor de la banda inferior (*fs1*) y superior (*fs2*) y la máxima atenuación (*Rs*).
- Exportar el filtro diseñado hacia el *workspace* de MATLAB, almacenar en un archivo los parámetros del filtro y utilizarlo para su codificación en el CCS.
- Con un generador de funciones conectado como entrada a la tarjeta DSK donde se encuentra desarrollado el filtro IIR, verificar su respuesta utilizando un osciloscopio o un analizador de espectro.

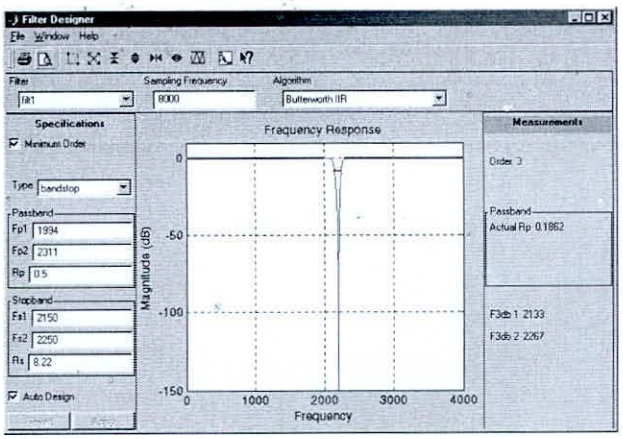

Figura 6 - Características del Filtro IIR.

Luego de haber realizado el experimento, se debe completar las siguientes tareas:

- Comparar la respuesta deseada con la obtenida con el filtro FIR diseñado para el mismo propósito.
- Realizar el diseño de un filtro IIR pasa-alto de orden 50, con una frecuencia de corte de 3kHz, con una frecuencia de muestreo de 8kHz y ejecutarlo en la tarjeta DSK.
- Comparar las realizaciones en forma directa y las de segundo orden en cascada.
- Describir los criterios para seleccionar el tipo de filtro a utilizar para una aplicación específica.

V. EXPERIMENTO No. 4: ANÁLISIS ESPECTRAL

La DFT es una de las herramientas más importantes en el procesamiento digital de señales, por lo tanto es fundamental conocer los conceptos que se hallen relacionados.[13] Así como, lo relacionado a su implementación a través de la Transformada Rápida de Fourier (FFT).[14] El objetivo en este experimento fue de implementar la FFT Radix-2 por decimación en el tiempo usando la tarjeta DSK, el cual requirió el uso del procesamiento por bloques de los datos de entrada, un tipo común de procesamiento en tiempo real.

5.1 Procedimiento experimental

- Utilizar MATLAB para generar una tabla de coeficientes *twiddle*, almacenarlo en un archivo e incluirlo en la codificación de la FFT en el CCS.
- Codificar el algoritmo de inversión de bits y el *butterfly* en el CCS.
- Utilizar los algoritmos anteriores para codificar la FFT de tiempo real y verificar la salida del DSP.

Seguidamente, para complementar la experiencia realizada, se propuso las siguientes tareas:

- Codificar una rutina que implemente una función FFT inversa usando la codificada para la FFT.
- Comparar los resultados obtenidos por la simulación respecto a su implementación.
- Determinar las limitaciones de la tarjeta DSK en el procesamiento de la señal de entrada.

VI. CONCLUSIONES

El diseño e implementación de sistemas DSP en tiempo real usando las herramientas provistas por MATLAB y el Kit de Desarrollo DSP ha hecho posible que el estudiante se haya familiarizado rápidamente con los algoritmos de PDS, utilizando un procesador DSP autónomo tal como lo propone Galanis,[15] entre otros.

El estudiante ha adquirido el conocimiento básico para afrontar con éxito otros temas de interés como de sistemas de comunicación propuesta por Tretter,[16] de control discreto utilizados por Colnaric[17] y Dixon[18] y de aplicaciones específicas como las descritas por Zoltowski.[19]

Se sugiere incluir racionalmente en los planes de estudios de pre y posgrado cursos de PDS de tal manera de poder transmitir el nivel adecuado de comprensión y cubrir en profundidad algunos tópicos que pueden ser esenciales en aplicaciones de nuestra realidad.

VII. AGRADECIMIENTOS

El autor expresa su sincera gratitud al Laboratorio de la Facultad de Ingeniería Electrónica, al Instituto de Investigación de la FIE-UNMSM, así como a los docentes del curso de procesamiento digital de señales, pues su apoyo ha sido fundamental para el desarrollo del presente trabajo.

REFERENCIAS

1. P. Hong , D. Anderson, D. Williams, J. Jackson, T. Barnwel, M. Hayes, R. Schafer, J. Echard (2004). DSP for Practicing Engineers: A Case Study in

Internet Course Delivery. IEEE Transactions on Education, vol. 47, no. 3. pp. 301-310.
2. C. Wright, T. Welch. (1999) Teaching real-world DSP using MATLAB. ASEE Computers in Education, Journal, vol. IX, pp. 1-5.
3. T. Instruments. (2000) TMS320C6000 CPU and Instruction Set Reference. Dallas.
4. T. Instruments (2001). TMS320C6000 Peripherals Reference Guide. Dallas
5. L. Karma (2004). Real-Time Digital Signal Processing, Arizona State University.
6. R. Chassaing (2002). DSP Applications Using C and the TMS320C6x DSK. Wiley, New York.
7. M. Galanis, E. Zigouris (2004). A Communication Laboratory based on the TMS320C6711 DSK. University of Patras, Greece.
8. T. Schlichter (1999). Digital Filter Design using MATLAB. Mississippi State University.
9. W. Gomes, R. Chassaing (2000). Filter Design and Implementation Using the TMS320C6x Interfaced with MATLAB. Dartmouth University, Massachussetts.
10. J. Proakis, D. Manolakis (1998). Tratamiento Digital de Señales. Prentice Hall.
11. R. Losada R. (2004). Practical FIR Filter Design in MATLAB. The MathWorks Inc. MathWorks.
12. A. Oppenheim, R. Schafer (2000). Tratamiento de Señales en Tiempo Discreto, 2da edición. Prentice Hall Hispanoamericana S.A.
13. A. Oppenheim, A. Willsky, S. Nawab (1998). Señales y Sistemas, 2da edición. Prentice Hall Hispanoamericana S. A..
14. R. Burden, D. Faires (2002). Análisis numérico, 7ma edición. Intenational Thomson Editores S.A.
15. M. Galanis, A. Papazacharias y E. Zigouris (2003). A DSP Course for Real-Time Systems Design and Implementation based on the TMS320C6211 DSK. University of Patras, Greece.
16. S. Tretter (2003). Communication Design Using DSP Algorithms: With Laboratory Experiments for the TMS320C6701 and TMS320C6711. Kluwer Academic/Plenum Publishers, New York.
17. M. Colnaric (2003). Design of embedded control systems. University of Maribor, Slovenia.
18. W. Dixon, D. Dawson, B. Costic, M. Queiroz (2002). A MATLAB-Based Control Systems Laboratory Experience for Undergraduate Students: Toward Standarization and Shared Resources. IEEE Transactions on Education, vol. 45, no. 3.
19. M. Zoltowski, J. Allebach, C. Bouman (1996). Digital Signal Processing with Applications: A New and Successful Approach to Undergraduate DSP Education. IEEE Transactions on Education, Vol. 39, No. 2. pp. 120-126.

ANEXO

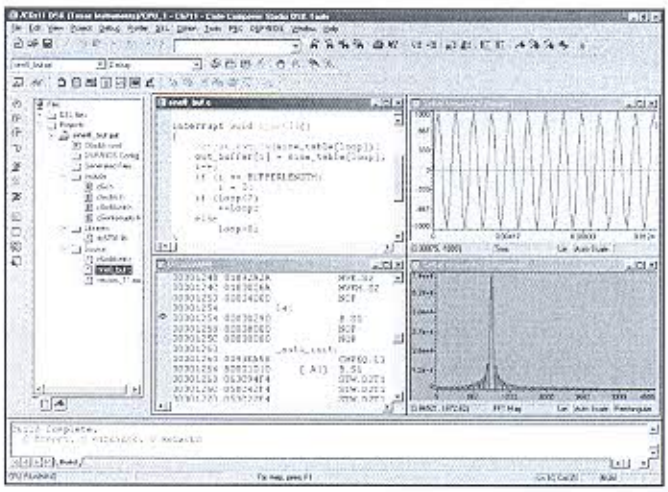

Figura A1. Ejecución en tiempo real de un oscilador sinusoidal desde el IDE del CCS.

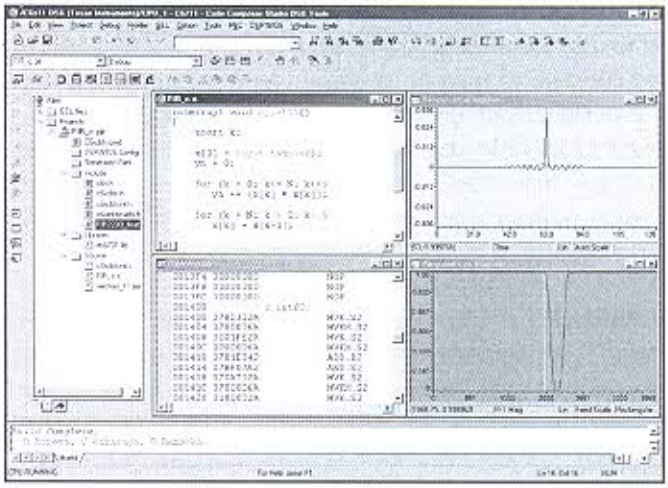

Figura A2. Ejecución en tiempo real de un filtro FIR desde el IDE del CCS.