OPTIMIZACIÓN DE CÓDIGO C, MEDIANTE SUBRUTINAS DE PUNTO FLOTANTE PARA EL DISEÑO DE SISTEMAS EMBEBIDOS

Jorge R. Osio, Federico Costantino, Sebastián Ledesma, José A. Rapallini, Antonio A. Quijano

Centro de Técnicas Analógico Digitales (CeTAD) - Codiseño Harware/Software (CoHS)
Facultad de Ingeniería. Universidad Nacional de La Plata
Calle 48 y 116, La Plata 1900, Argentina
josio@gioia.ing.unlp.edu.ar sledesma@barcala.ing.unlp.edu.ar
fcostant@barcala.ing.unlp.edu.ar **josrap@ing.unlp.edu.ar** quijano@ing.unl.edu.ar

RESUMEN

El Entorno de Diseño PeaCE se caracteriza por ser una herramienta eficiente para el prototipado rápido de sistemas embebidos, reconfigurables y de bajo costo, mediante técnicas de codiseño HW/SW. Una de las principales ventajas que ofrece es la generación de código eficiente en la síntesis de sistemas.

El código C generado está orientado a implementaciones en DSPs (variables en Punto Flotante de 32 bits).

Inicialmente se adaptó este código para el uso en procesadores genéricos (Microcontroladores de 8 bits), reemplazando funciones de Punto Flotante por funciones de 8 bits, con la consiguiente perdida de resolución [1].

El Diseño de Sistemas Embebidos simples que necesitan precisión numérica pero no gran procesamiento matemático, llevo a la creación de subrutinas de operaciones en Punto Flotante (suma, resta multiplicación y división) eficientes. Como aplicación se implementa un generador de señal sinusoidal para los MCUs 908 (utilizable para un modulador PSK).

1. INTRODUCCIÓN

Los sistemas embebidos [2] se aplican a infinidad de soluciones electrónicas, resolviendo problemáticas cada vez más complejas.

La planificación del sistema se realiza aplicando técnicas de codiseño hardware / software, tratando de desarrollar métodos y herramientas que pueden usarse para mejorar la calidad de la aplicación final.

El diseño de un sistema comienza con una descripción funcional que da origen a la especificación y a su simulación, obteniendo luego una síntesis del código a utilizar.

El Entorno de desarrollo PeaCE [3] es una de las herramientas más eficientes para el prototipado rápido de sistemas embebidos, que permite realizar la especificación, simulación y síntesis del mismo.

PeaCE integra diversos modelos de computación (para control y flujo de datos), capacidades poderosas de simulación especialmente para aplicaciones de Procesamiento de Señales Digitales (DSP).

El código C generado por PeaCE es muy eficiente para aplicaciones en DSPs. Para que este código sea portable por otras arquitecturas, se deben tener en cuenta algunas consideraciones. En el caso de Arquitecturas MCUs de 8bits, si se desea mantener la precisión del código C generado por PeaCE, al mismo se le deben agregar subrutinas que realicen operaciones en punto flotante, desarrolladas especialmente para estas arquitecturas, ya sea como "archivos include" o dentro del mismo código.

De está manera utilizando un programa traductor como el software ICC o el Codewarrior, se puede generar código de máquina para distintos microprocesadores genéricos.

2. FLUJO DE DISEÑO

La Figura 1 muestra el flujo de diseño a seguir para lograr el prototipado rápido de sistemas embebidos cuyas arquitecturas destino pueden ser, ya sea, DSPs, FPGAs o microprocesadores genéricos.

En este trabajo el flujo de diseño seguido es el que marcan los bloques de borde resaltado. En donde se realiza la descripción y simulación del sistema en PeaCE y se obtiene el correspondiente código C generado. Luego mediante subrutinas eficientes de funciones matemáticas, en punto flotante, se adapta el código C generado a una Arquitectura MCU.

Estás subrutinas de punto flotante realizan la suma, resta, multiplicación y división. Mediante estás operaciones y la utilización de algoritmos matemáticos se pueden implementar las demás operaciones. En este trabajo se utiliza el Teorema de Taylor y la formula de MacLaurin para la generación de una onda sinusoidal.

Por último se realiza la implementación del sistema en microcontroladores de 8 bits.

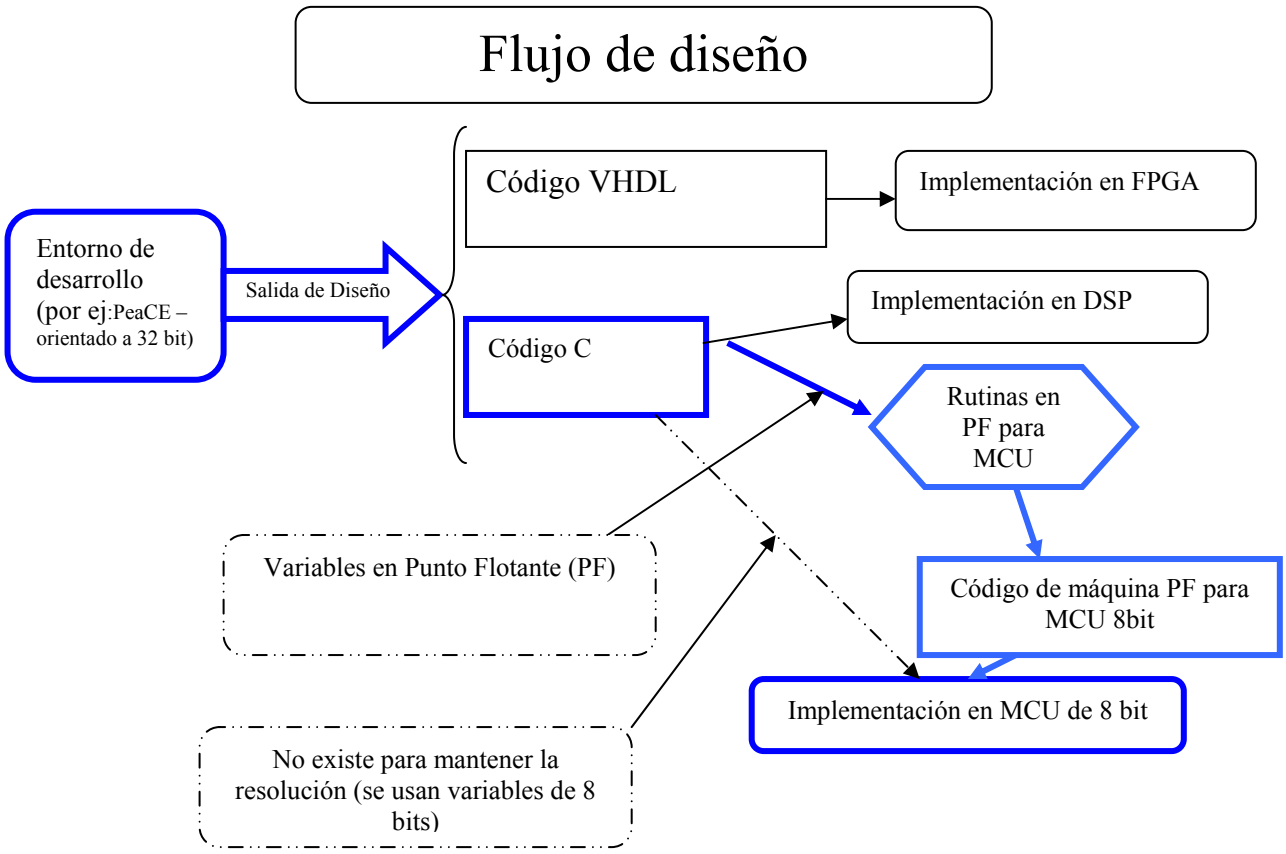

Figura 1. Flujo de diseño

3. DISEÑO DE SISTEMAS EMBEBIDOS EN PEACE

Se identifican los siguientes cuatro puntos como los temas indispensables en la metodología de codiseño HW/SW:

- La síntesis a nivel especificación
- Exploración sistemática del espacio de diseño
- Co-verificación y Co-simulación HW/SW

El flujo de Codiseño en PeaCE [3] es re-configurable. Cada paso de diseño es modularizado a fin de que los diversos algoritmos u otras herramientas CAD puedan ser integradas con un mínimo esfuerzo de diseño para traducir los archivos .xml (Los archivos de extensión .xml indican las entradas y salidas de los pasos de diseño, planificación, simulación).

El flujo de diseño comienza con la especificación del comportamiento del sistema con modelos formales: modelo de flujo de datos (SDF) para procesamiento de señales y un modelo de Maquinas de Estados Finitos (FSM) para el módulo de control del sistema. En el mas alto nivel, se usa un modelo de tarea (BP o DE) para modelar la interacción entre las tareas. La interfase gráfica del usuario en PeaCE, llamada Hae, permite al usuario dibujar un diagrama en bloques jerárquico reusando tantos bloques predefinidos como sea posible.

A continuación se detallan los pasos de diseño:

***Paso1*:** Simulación funcional de comportamiento del sistema. Desde la especificación se genera un código C, se compila y se ejecuta en la máquina del usuario por simulación.

Paso 2: Se especifican las posibles arquitecturas a ser exploradas.

Paso 3: Se estima la performance de software de cada bloque funcional en cada núcleo procesador examinando si está disponible en la librería de diseños.

Paso 4: Este paso traduce la especificación gráfica a tres tipos de archivos de texto, la cuál describe la topología de la gráfica, la información de desempeño del bloque, y las restricciones de temporización de tareas.

Paso 5: En este paso se realiza *particionamiento HW/SW* y la selección de componentes al mismo tiempo.

Paso 6: Después del particionamiento, PeaCE genera códigos particionados y co-simulados y el sistema para generar el seguimiento de memoria de los elementos de procesamiento. Usando el seguimiento de memoria e información del planificador, se explora la arquitectura de bus. La arquitectura final de bus se registra en un archivo de texto, archi.xml.

Paso 7: Después de determinar la arquitectura de bus, se verifica la arquitectura final antes de la síntesis. Este paso realiza co-simulación HW/SW en tiempo real para co-verificación.

Paso 8: Este paso es para generar los códigos para los elementos procesadores en una placa prototipada. Para un núcleo procesador, se genera un código C y para

la implementación de hardware FPGA se genera código VHDL.

4. CÓDIGO GENERADO POR PEACE EN LENGUAJE C

El código C generado por PeaCE (Figura 2) está orientado a aplicaciones DSPs con variables de punto flotante. Las operaciones definidas en lenguaje C que utilizan valores en punto flotante, como la función seno, son muy densas para usarlas directamente en el traductor de C a Código de Máquina. Una función seno, definida en C, aplicada directamente al programa cuando se compila en el software ICC genera un código de máquina que ocupa aproximadamente el 80% de la memoria en un Microcontrolador de la Familia HC908.

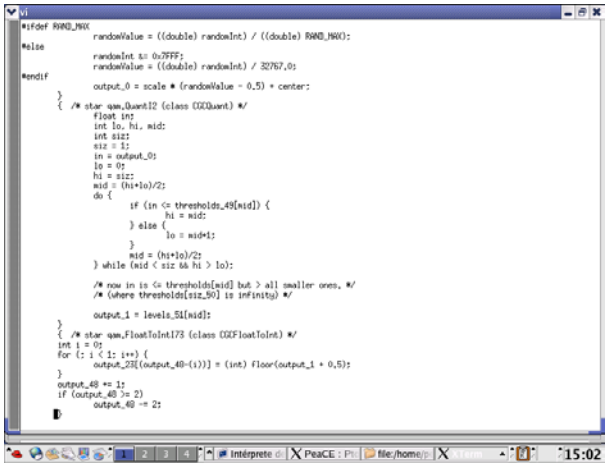

Figura 2. Código generado por PeaCE

5. SUBRUTINAS EN PUNTO FLOTANTE

En esta sección se describen detalles importantes sobre las rutinas de multiplicación, división, suma y resta de números en formato de punto flotante [4], siguiendo las especificaciones de punto flotante descriptas por la Norma 754 de la IEEE. Las operaciones se realizan con valores en punto flotantes en simple precisión, es decir, que cada número se representa con 32 bits.

Un número en punto flotante según la norma está formado por los siguientes campos: **S** es el bit de signo y **Exp** es el campo exponente. El signo del exponente **s** es de un bit y la **Fracción o mantisa** es de 23 bits**.**

Signo del número real S:	1 bit
Signo del exponente s:	1 bit
Exponente (entero |Exp|):	7 bits
Mantisa (número real |fracción|):	23 bits

5.1. Operaciones matemáticas en punto flotante

Se deben realizar varias verificaciones antes de realizar el producto de las mantisas.

Si dos números están expresados en punto flotante de la forma $\mathbf{m2^n}$ y $\mathbf{o2^p}$, donde **m** y **o** son significandos, **n** y **p** los exponentes, en una multiplicación se deben multiplicar los significandos y sumar algebraicamente los exponentes: $\mathbf{m\ 2^n * o\ 2^p = (\ m * o)\ 2^{(n+p)}}$

En una división se deben dividir los significandos y restar los exponentes: $\mathbf{m\ 2^n / o\ 2^p = (\ m / o)\ 2^{(n-p)}}$

Teniendo en cuenta esto, cuando se realiza una **multiplicación**, se debe tener en cuenta que Los significandos están representados sólo con la parte decimal, donde tácitamente el entero es un "1". Por lo tanto se deben desnormalizar los significandos poniéndolos en formato 1,x.....x y luego multiplicarlos. Al resultado se lo vuelve a expresar con solo la parte decimal. Los exponentes se deben sumar algebraicamente, pero como éstos están formateados con el bias, se deben desnormalizar y luego sumar.

El caso de la **división** es análogo, donde puede ser que al dividir los significados el resultado de menor que 1, y se deba correr la coma hacia la izquierda, es decir, restar un "1" al resultado luego de restar los exponentes.

En el caso de la **suma** o **resta,** si la diferencia entre los exponentes de los argumentos es mayor a los bits de la mantisa, el resultado es el "mayor" argumento (ya que el argumento menor no logra afectar el resultado). Para realizar la operación se debe desplazar a la derecha, la mantisa del argumento con menor exponente la cantidad de bits necesarios hasta igualar los exponentes. Si los argumentos tienen distinto signo, se debe complementar la mantisa del número de menor exponente. La mantisa resultado, es la suma de las mantisas. El exponente resultado, es el mayor de ambos. El signo resultado es el signo del de mayor exponente.

5.2. Datos importantes sobre las operaciones en el MCU.

Al hacer una operación matemática si el resultado es negativo, el microcontrolador lo expresara en CA2 y seteará la bandera N del CCR [5].

Cuando se realice la suma de los exponentes puede que esta se vaya de rango. En este caso será el infinito y directamente ponemos el valor del exponente en 11.....1.

Cuando se realice la suma de los exponentes puede que esta de cero. En este caso será infinito y se pone el valor del exponente en 11.....1 con el 1° bit en 1. En ambos casos no se multiplican las mantisas ya que el resultado será un NaN (no es un número).

Cuando los dos números sean muy chicos, entonces los exponentes normalizados serán menores a cero. Este es el caso en el que el resultado es cero.

6. APLICACIÓN

La aplicación consiste en la generación de una onda sinusoidal mediante algoritmos matemáticos que utilicen las subrutinas realizadas. Comparando los dos algoritmos matemáticos mas utilizados para la realización de una sinusoide, (el basado en el Teorema de Taylor y el basado en tablas).

6.1. Generador de onda sinusoidal mediante el Teorema de Taylor

Teorema de Taylor:

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f^2(x_0)}{2!}(x-x_0)^2+..+\frac{f^n(x_0)}{n!}(x-x_0)^n+E_n$$

Formula de MacLaurin es un caso particular de Taylor para $x_{0=0}$:

$$f(x)=f(0)+f'(0)x+\frac{f^2(0)}{2!}x\ +..+\frac{f^n(0)}{n!}x^n+E_n$$

Las derivadas de orden par, evaluadas en cero se anulan y las impares valen alternadamente 1 y -1. De esta forma se obtiene:

$$sen(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+..+(-1)^{n+1}\frac{x^{2n-1}(0)}{(2n-1)!}+E_{2n-1}$$

6.2. Generador de onda sinusoidal por medio de tablas.

Si se toman sucesivamente todos los valores de la tabla en un periodo de T segundos, se obtiene una onda senoidal con la frecuencia fundamental: F = 1/NT Hz

De esta forma se obtienen frecuencias a partir de la fundamental variando delta (Δ=2, Δ=3, Δ=4 ...).

f = Δ* F Hz; Δ ≤ N/2

El máximo valor de delta puede ser N/2 para obtener al menos dos muestras por ciclo.

i = N-1	sin[(N-1) • 360/N]
.	.
i	sin[(i) • 360/N]

N = tamaño de la tabla

i = índice; 0 ≤ i ≥ N – 1

6.3. Resultados Obtenidos

El método seleccionado para la generación de una sinusoide es el de Tablas ya que tiene una rápida ejecución y ocupa poco espacio en memoria, dependiendo de la cantidad de valores en la tabla. El de Taylor es de lenta ejecución y se deben usar muchos coeficientes para obtener buena resolución.

En la Figura 3 se puede observar la compilación del código optimizado para MCUs (sin errores) [6] [7].

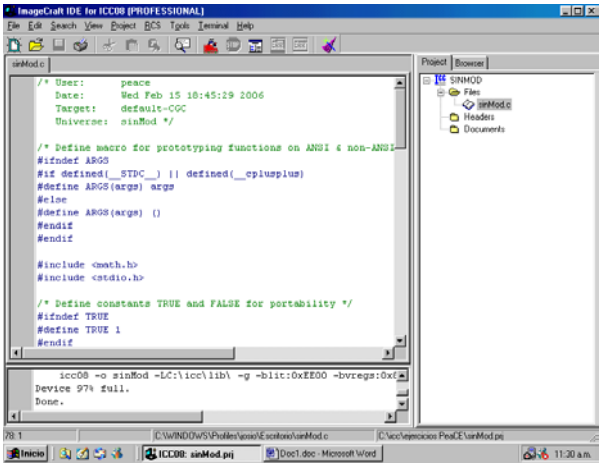

Figura 3. Compilación del código generado por PeaCe optimizado.

En la Figura 4 se muestra la calidad de la señal sinusoidal generada.

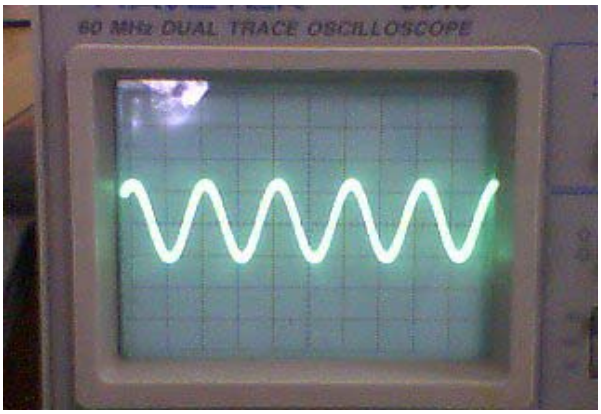

Figura 4. Resultados obtenidos en el osciloscopio.

7. CONCLUSIONES

Los resultados de la aplicación realizada demuestran las ventajas del ambiente de codiseño PeaCE, para ser utilizado con propósitos de investigación y generar prototipos funcionales en un corto tiempo. En definitiva, facilita todos los pasos de Codiseño desde la especificación del sistema hasta el prototipado. Además, se demuestra que mediante las subrutinas de operaciones, en punto flotante, creadas y la utilización de algoritmos matemáticos que hacen uso de ellas, se pueden diseñar sistemas embebidos simples que requieren de precisión numérica pero no gran procesamiento matemático.

8. REFERENCIAS

[1] Jorge Osio, José Rapallini, Antonio Quijano, “Análisis de Modelos Computacionales para Sistemas Embebidos,” Iberchip, San José de Costa Rica, Marzo de 2006.

[2] S. Edwards, L. Lavagno, E. A .Lee, and A. Sangiovanni-Vincentelli, "Design of Embedded Systems: Formal Models, Validation, and Synthesis," IEEE Proceedings, pp. 366-390, March 1997.

[3] User´s Manual: PeaCE User´s Manual, Version 1.0, Linux, CAP Laboratory of Seoul National University and the Pringet corporation, may 28, 2003.

[4] An American National Standard, “IEEE Standard for Binary Floating-Point Arithmetic”, Approved July 26, 1985 Reaffirmed May 21, 1991. American National Standards Institute.

[5] Ing. Gabriel Dubatti, http://www.ingdubatti.com.ar/artmath.htm

[6]“Embedded C Development tools, ” http://www.imagecraft.com, 1994.

[7] ”P&E Microcomputers Sistems,” http://www.pemicro.com, 2004.