Análisis de Modelos Computacionales para Sistemas Embebidos.

Jorge Osio, Federico Salguero, José Rapallini, Antonio 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 fsalguero@barcala.ing.unlp.edu.ar **josrap@ing.unlp.edu.ar** quijano@ing.unl.edu.ar

ABSTRACT

After a short summary about the principles of Hardware/Software Codesign (HSCo) for embedded systems, this paper considers an analysis of the software resources mostly used by the academic community with special emphasis on one of them (PeaCE).

An example shows how to specify a system considering the computer models, specially the Data Flow formal model. The required characteristics are defined, with the periodic update of parameters and dynamic behavior of functional blocks.

An application from the communication area is presented to show the advantages of the chosen system of HSCo. If can generate an efficient code (C and VHDL) very useful to be implemented on generic processors or FPGA, for rapid prototyping.

RESUMEN.

Después de una recapitulación de conceptos de Codiseño Hardware / Software (CoHS) para sistemas embebidos, se realiza un análisis de los recursos mas utilizados en el ambiente académico y se discuten las características de uno ellos, el PeaCE.

A través de un ejemplo, se muestra como se puede especificar un sistema, considerando los modelos computacionales, en particular el modelo formal "flujo de datos" (Data Flow – DF), definiendo las características necesarias, teniendo en cuenta su actualización periódica de parámetros y comportamiento dinámico de sus bloques funcionales.

Finalmente se presenta una aplicación del área de comunicaciones, demostrando las ventajas del ambiente de CoHS utilizado. En particular, en el prototipado rápido, dado que realiza la generación de código eficiente (C y VHDL) para su implementación en procesadores genéricos o FPGA.

Palabras clave: Codiseño HW/SW, Lenguajes de Alto Nivel, Modelos Computacionales, Sistemas Embebidos

Análisis de Modelos Computacionales para Sistemas Embebidos.

Jorge Osio, Federico Salguero, José Rapallini, Antonio 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 fsalguero@barcala.ing.unlp.edu.ar josrap@ing.unlp.edu.ar quijano@ing.unl.edu.ar

RESUMEN.

Después de una recapitulación de conceptos de Codiseño Hardware / Software (CoHS) para sistemas embebidos, se realiza un análisis de los recursos mas utilizados en el ambiente académico y se discuten las características de uno ellos, el PeaCE.

A través de un ejemplo, se muestra como se puede especificar un sistema, considerando los modelos computacionales, en particular el modelo formal "flujo de datos" (Data Flow – DF), definiendo las características necesarias, teniendo en cuenta su actualización periódica de parámetros y comportamiento dinámico de sus bloques funcionales.

Finalmente se presenta una aplicación del área de comunicaciones, demostrando las ventajas del ambiente de CoHS utilizado. En particular, en el prototipado rápido, dado que realiza la generación de código eficiente (C y VHDL) para su implementación en procesadores genéricos o FPGA.

1. INTRODUCCIÓN

Los sistemas embebidos juegan un papel muy importante en nuestra vida cotidiana, lo vemos en telecomunicaciones, instrumentación médica, automóviles, sistemas multimedia, etc., extendiéndose día a día y resolviendo problemáticas cada vez más complejas.

Para el desarrollo de los mismos, es necesario tener en cuenta los aspectos de hardware y de software desde que se elabora la idea. El área de codiseño hardware / software es la que se encarga de la planificación del sistema, tratando de desarrollar los métodos y herramientas que pueden usarse para mejorar la calidad de la aplicación final.

Todo proyecto de este tipo, comienza con una descripción del comportamiento que da origen a la especificación y a su simulación funcional, obteniendo luego una síntesis del código a utilizar. Existen distintas herramientas para resolver este problema, pero para su utilización debemos conocer que tipo de modelo computacional maneja y su posibilidad de adaptación a la problemática planteada.

De las herramientas mas utilizadas en esta área, PeaCE [1], Ptolemy [2] y Simulink [3], se puede determinar que Ptolemy es una buena herramienta para el modelado y la simulación de sistemas pero su capacidad en la síntesis de código está muy restringida por la implementación del sistema; puede producir código C desde una representación de flujo de datos (DF) pero la calidad de código no es satisfactoria. Apenas podría producir un código sintetizable VHDL, pero no del todo optimizado. No hay modo de sintetizar arquitectura desde Ptolemy. Para solucionar todos estos problemas, se creó el proyecto PeaCE que es una extensión de Ptolemy en el ambiente de Codiseño. Como está construido sobre la base de Ptolemy, básicamente PeaCE hereda todas las características positivas de Ptolemy, como la integración de diversos modelos de computación, capacidades poderosas de simulación especialmente para aplicaciones de Procesamiento de Señales Digitales (DSP), mejorando las restantes.

Simulink es un paquete de Software que permite modelar, simular y analizar sistemas dinámicos, esto es sistemas cuyas salidas y estados internos cambien con el tiempo. Es un entorno gráfico, donde se crea un modelo en bloques del sistema, utilizando librerías de bloques estándar y un editor que permite interconectar los bloques del sistema. El modelo representa gráficamente las relaciones matemáticas dependientes del tiempo a través de las entradas, estados y salidas del sistema. Con el agregado de paquetes adicionales, permite simular máquinas de estados finitos con jerarquía, flowcharts; generar código C para diferentes plataformas o código ADA, a partir del modelo del sistema. Además de poder simular en *tiempo real*, permite implementar modelos en *tiempo continuo*, en *tiempo discreto*, *mixto* (discreto y continuo), *máquinas de estado finito* (con jerarquía y concurrencia) y *flowcharts*. La gran ventaja que posee sobre PeaCE es que al estar mucho mas difundido tiene una gran cantidad de Librerías. La desventaja que posee es que está orientado a control, en cambio PeaCE

posee modelos de computación de control y de flujo de datos.

En conclusión se adoptará el Entorno PeaCE como mejor solución a los requerimientos antes nombrados, analizando sus principales virtudes en el desarrollo de este informe.

2. CODISEÑO HW/SW PARA SISTEMAS EMBEBIDOS.

Como el desarrollo de tecnología de implementación avanza a grandes pasos, el diseño de sistemas embebidos multimedia se ve perjudicado por la complejidad creciente del sistema y la necesidad de un rápido desarrollo del diseño . La práctica convencional en el diseño de sistemas embebidos

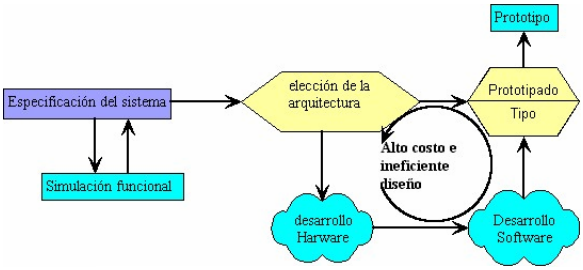

Figura 1: Flujo de diseño convencional

multimedia como se muestra en la **Figura** 1 realiza el diseño del algoritmo, diseño de hardware, y diseño de software separadamente y secuencialmente. Después de varias experiencias de diseño y de aplicar algunas técnicas que optimizan el desempeño, se decidió que una vez seleccionada la arquitectura de hardware, se debe determinar que partes de las funciones del sistema son implementadas con que componentes, para luego realizar el desarrollo de software optimizado y el prototipado de hardware concurrentemente. Luego de hacer el prototipo de Hardware, se programan los procesadores con los códigos de software desarrollados y se verifican.

Ya que la esperada tasa de crecimiento en la productividad de los diseños, aplicando los métodos de diseño convencionales, está muy por debajo de la complejidad del sistema, el ***Codiseño Hardware/Software*** (HW/SW) [4] ha surgido como una nueva metodología de diseño a nivel sistema, la cual no separa el diseño hardware y el diseño software (**Figura 2**). Una vez seleccionada la arquitectura de hardware, se explora la gran variedad de diseños factibles y se evalúa cada uno, estimando el desempeño esperado, luego se consideran todas las demás decisiones del diseño como *particionamiento hardware/software* e *implementación software*. Un desarrollo sistemático del espacio de diseño necesita separar especificación de funciones y arquitecturas. El mapeo es la obtención a partir del modelo computacional de un modelo físico. El mapeo explícito consiste en planificar parte de la especificación funcional en la arquitectura de los bloques construidos. Luego, se debe estimar la performance esperada y en base a esto, realizar más iteraciones de especificación de arquitectura y mapeo hasta que se logre una arquitectura óptima. Luego, se hace una verificación exacta de performance antes de la implementación final del hardware, en caso de que el costo de implementación del hardware sea muy alto (por ejemplo para la implementación del Sistema en Chip). Así Codiseño hardware/software incluye soluciones a problemas diversos de diseño incluyendo la especificación del sistema, particionamiento hardware/software, estimación de desempeño, co-verificación hardware/software, y síntesis del sistema. Un *Ambiente de Codiseño* es una herramienta de software que facilita la solución de esos problemas de diseño.

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
- Síntesis de código automática desde modelos a nivel sistema

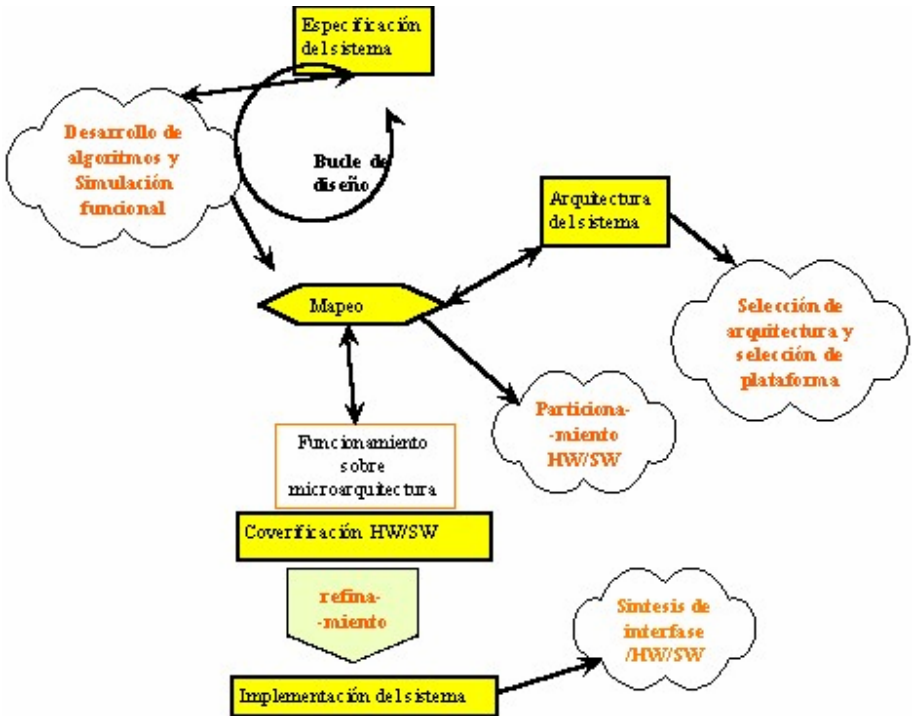

Figura 2: Codiseño HW/SW separando funciones y especificación de arquitecturas.

Entre estos, las herramientas de Codiseño comerciales le dan prioridad a la Co-simulación / Co-verificación HW/SW, ya que la necesidad mas urgente es tener un ambiente virtual para prototipado que permita desarrollar el software antes que se haga el prototipo del hardware. La Co-simulación Hardware/Software permite a los diseñadores evaluar o verificar el sistema bajo diseño antes que se fabrique. La Co-simulación se puede usar en las diferentes etapas de diseño con propósitos diferentes; puede usarse para evaluar el desempeño del sistema, verificar la precisión funcional y la temporización del sistema. La práctica actual de Codiseño HW/SW se basa en estas herramientas como se muestra en **Figura 3**.

Estas herramientas de codiseño tienen los siguientes problemas que **PeaCE** pretende solucionar:

1. Cada paso de diseño esta aislado, si bien esa integración de herramientas de diseño es problemática. PeaCE provee flujo de Codiseño libre de irregularidades, desde simulación funcional hasta síntesis de sistema.
2. El flujo de diseño puede ser pesado, dependiendo de la herramienta de diseño seleccionada. PeaCE posee una estructura configurable para la cual la tercera parte de la herramienta de diseño puede ser fácilmente integrada.
3. La exploración del espacio de diseño debe hacerse manualmente usando una herramienta de simulación TLM (Transaction level Modules -nivel de interacción entre los módulos). PeaCE

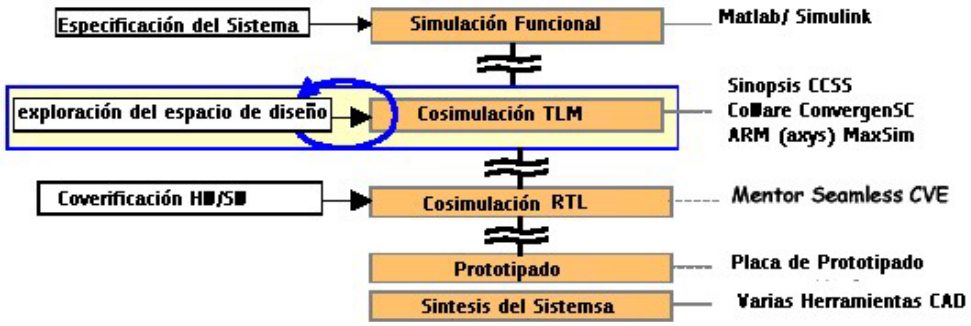

Figura 3: Práctica actual de codiseño

incluye una estructura interactiva DSE (Design Space exploration -Exploración del Espacio de Diseño) que consta de particionamiento HW/SW [4] y optimización de la arquitectura de comunicación.

El flujo de Codiseño en PeaCE 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). La secuencia de pasos de diseño se asocia con la conexión de diseño en la interfaz del usuario.

El flujo de diseño comienza con la especificación del comportamiento del sistema con modelos formales: modelo dataflow para computación 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 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. Tal re-uso de diseño reduce el tiempo de diseño. Ya que cada diagrama en bloques se dibuja con una regla de ejecución bien definida, el diseño puede entenderse con facilidad sin ayuda del diseñador. Esto facilita el mantenimiento del diseño y posibilita a la gente trabajar conjuntamente sin confusiones. A continuación se detallan los pasos de diseño:

Paso1: Simulación funcional de comportamiento del sistema. Esta simulación se puede integrar con otras herramientas, como por ejemplo MATLAB u Octave. Desde la especificación se genera un código C, se compila y se corre en la máquina del usuario por simulación. Esta capacidad es comparable con Simulink a fin de que el usuario pueda reemplazarlo con PeaCE. La diferencia principal entre PeaCE y las otras herramientas, es que no es un motor de simulación pero es una herramienta de generación de código equivalente a la simulación funcional. Hace simulación que se puede depurar fácil y rapidamente.

Paso 2: Se especifican las posibles arquitecturas a ser exploradas. En este paso, listamos todos los elementos de procesamiento y los núcleos procesadores, que están disponible en la librería de diseños.

Paso 3: Se estima la performance de software de cada bloque funcional en cada núcleo procesador examinando si no está disponible en la librería de diseños. En este paso, se usa un ISS (simulador de un conjunto de instrucciones) del núcleo procesador. El usuario puede saltearse este paso si todas las estimaciones de performance de bloques funciónales ya están registradas en la librería.

Paso 4: Este paso traduce la especificación gráfica a tres conjuntos 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. Tal interpretación modulariza a PeaCE de manera que el siguiente paso de diseño no dependa de su núcleo y de la interfase del usuario.

Paso 5: En este paso se realiza *particionamiento HW/SW* y la selección de componentes al mismo tiempo. La salida está escrita en un archivo de texto, lo cual describe el mapeo y la planificación de bloques funcionales encima de cada elemento procesador. Si bien PeaCE provee herramientas de particionamiento HW/SW , el usuario puede usar otra herramienta gracias al mecanismo de la interfaz de archivo.

Paso 6: Después del particionamiento, genera códigos particionados y co-simulados y el sistema para generar el seguimiento de memoria de los elementos de procesamiento. Peace provee una herramienta de Co-simulación basada en la técnica virtual de sincronización, la cual es rápida y exacta en tiempo. En este paso, se asume que la arquitectura de comunicación todavía no está determinada. Usando el seguimiento de memoria e información del planificador, exploramos la arquitectura de bus. Más adelante se explorarán otras arquitecturas de comunicación. 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. Podemos usar CVE (Co-verificación del Ambiente) (no posee irregularidades) en este paso o la herramienta de co-simulación usada en el *paso 6*, pero usando modelado exacto de arquitectura del bus.

Paso 8: Este paso es para generar los códigos para los elementos procesadores en una placa prototipada. Para un núcleo procesador, generamos un código C y un código VHDL para la implementación de hardware FPGA (FIELD-PROGRAMABLE GATE ARRAY- Campos de arreglos de puertas programables).

En los pasos 6 a 8, se deben generar códigos de acuerdo a la decisión de particionamiento hecha en el paso 5. Quiere decir que se deberán generar los códigos de la interfase entre elementos de procesamiento y el código envolvente para la herramienta de simulación (paso 6 y paso 7) o para la placa de prototipado (paso 8). Si se quiere usar una herramienta de simulación o una placa de prototipado diferente , entonces se debe crear un nuevo objeto Target (arquitectura destino) y nuevos bloques de la interfaz.

3. MODELOS COMPUTACIONALES

3.1 Especificación de comportamiento del sistema

Mientras una descripción secuencial de código " C " es la mas deseada para la simulación funcional en el procedimiento convencional de diseño, no es adecuada para la especificación inicial del sistema en Codiseño HW/ SW. Ya que un sistema se implementa como una conjunto de componentes concurrentes como procesadores programables, ASICs (Circuitos integrados de aplicación especifica), y componentes discretos de hardware, para una especificación inicial se prefieren modelos computacionales que expresen concurrencia [5]. Ellos especifican un sistema, como una red de bloques funciónales que se comunican por medio de eventos (o tokens "señales"). Los modelos de computación se diferencian por la forma en que se ejecuta cada bloque, y la manera en que se comunican entre ellos [6]. El gráfico de SDF y sus extensiones han sido una representación exitosa de algoritmos para DSP ya que su semántica está adecuadamente acoplada con flujo de funciones algorítmicas en aplicaciones con DSP. En un gráfico de flujo de datos estático, un nodo representa un bloque funcional y consume/ produce un número conocido de muestras de todos los arcos (conector que conecta a dos o mas bloques funcionales entre si) de entrada (y todos los arcos de salida) cuando se ejecuta. Un nodo es ejecutable sólo cuando todos los arcos contienen una cantidad especificada de señales.

El uso del *modelo de flujo de datos estático* como especificación inicial tiene varios beneficios sobre la *programación "C"* convencional:

- El análisis Estático de la especificación permite verificar la exactitud de la especificación funcional.
- El modelo de especificación puede refinarse para una implementación que facilita la validación del sistema por el principio de "construcciones correctas".
- Una especificación formal del modelo no tiende a ningún método específico de implementación; de esta manera también permite explorar el espacio de diseño.
- Representa el comportamiento del sistema sin ambigüedades a fin de que el mantenimiento del diseño puedan realizarse fácilmente.

A pesar de estas prestaciones, un modelo de flujo de datos estático no sirve completamente para la especificación de sistemas ya que tiene un par de problemas serios:

- La capacidad de expresión de un modelo de flujo de datos estático está muy restringida, pues se necesita que otro modelo especifique la estructura de control del comportamiento del sistema y la dirección de tareas múltiples.
- El modelo estático de flujo de datos resulta en conclusión ineficiente en cuanto a la síntesis si se mantiene el principio de "construcción correcta". El modelo de flujo de datos no sirve para manipular estructuras grandes de datos como por ejemplo tramas de video porque está prohibida la operación del puntero para una memoria global compartida. Esto se hace para evitar efectos secundarios independientemente de la planificación de ejecución del nodo.

El primer problema puede solucionarse usando una mezcla de modelos heterogéneos de computación para la especificación de sistemas en una sola estructura: Por ejemplo un modelo flujo de datos estático para la descripción del algoritmo para DSP, un Modelo FSM para la parte de control, y otro modelo para la especificación a nivel tarea. La estructura de Ptolemy es un ejemplo conocido que usa este método, integrando diferentes modelos de computación en una sola estructura de forma jerárquica [7].

El propósito del ambiente de Codiseño “PeaCE” es solucionar el segundo problema haciendo dos extensiones al modelo SDF.

PeaCE también adopta una metodología heterogénea que usa un modelo extendido FSM para la tarea de control y un modelo estático de flujo de datos para las tareas DSP. En el más alto nivel, PeaCE usa un nuevo *modelo de tarea* para la especificación a nivel tarea.

3.2 Ejemplo de especificación.

Para ilustrar las características de las dos extensiones del modelo SDF, se presenta como ejemplo un codificador/ decodificador de video H.263 [8]. Para estos sistemas se necesita la integración de voz y algoritmos de procesamiento de video.

Se deben considerar cuatro problemas cruciales en la especificación del sistema H.263.

- Se necesita un modelo interno de especificación de tareas DSP que serán el objetivo principal del particionamiento HW/SW. Ya que el codificador H.263 y los algoritmos de decodificación contienen ejecuciones de nodos de dependencia de datos como construcciones "if-then-else" y "for", entonces el modelo deberá soportar construcciones dinámicas. Y el resultado de la síntesis del modelo deberá ser tan eficiente como un código manualmente optimizado.

- Se deberá soportar la interacción entre las tareas de control y las tareas DSP. Las tareas modo tareas&control manejan el estado de tareas DSP según el requerimiento de cambio de modo, y da parámetros del usuario a la tarea DSP correspondiente asincrónicamente en el requerimiento del usuario.
- Las Tareas del H.263 tienen diversas semánticas de ejecución: manejada por datos, manejada por eventos, y manejada por tiempo.
- Se necesita especificar un nuevo tipo de puerto, llamado tipo *tasa dinámica de tamaño variable (DRVS)*, en la tarea de flujo de datos. Este tipo de puerto es muy útil para la codificación /decodificación de MP3, el tamaño requerido de datos de una trama codificada no puede determinarse hasta que el decodificador de MP3 comience la tarea de decodificación y la detección en tiempo de ejecución. Por lo tanto no existe modelo de computación que pueda especificar esta semántica de puerto.

En las siguientes sub-secciones se expone la solución de estos problemas.

3.2.1 Modelo SDF extendido para la especificación de tarea DSP

Se presentan dos extensiones del modelo SDF para solucionar el primer problema. La primera extensión , permite a un nodo ser condicionalmente ejecutado durante la simulación definiendo un flag de condición para el nodo. Cuando se realiza la compilación, sin embargo, todos los nodos son considerados como nodos estáticos, esto quiere decir que las tasas de muestra son conocidas y fijas. El flag de condición se puede cambiar desde afuera del nodo, sin riesgo de efectos secundarios, porque cuando se determina la renovación del flag después del tiempo de compilación se fija la planificación. Esta extensión, llamada " Synchronous Piggybacked Data Flow – flujo de datos sincrónicos acoplados (SPDF) ", se realiza simplemente introduciendo un actor SDF especial, acoplado, ocultando todo análisis en tiempo de compilación para realizar la operación [9] . Mediante SPDF, se pueden expresar ejecuciones de nodos con dependencias de datos en el contexto de modelo de flujo de datos estático.

La segunda extensión debe usar memoria fragmentada y punteros para poder acceder a las muestras de datos de estructuras de datos de tamaño grande, como por ejemplo tramas de video, también sin efectos secundarios. Esta extensión se denominó "Fractional Rate Data Flow – flujo de datos de tasa fraccionada (FRDF)" . En FRDF el número fraccionado de muestras puede ser producido o consumido [10]. La figura 4 (a) muestra una sub-gráfica SDF de un algoritmo del codificador H.263. El bloque ME recibe la trama actual y las tramas previas para computar las diferencias de movimiento de vectores y pixel, y el bloque D divide la trama de entrada en 99 macro bloques para ser codificados en el siguiente bloque EN. El bloque ME estima una trama de video de 144x176 como una muestra (unidad de datos), mientras el bloque EN es un macro bloque de 16x16 y el bloque D se usa para la conversión explícita de tipo de datos. El código sintetizado desde la gráfica de la Figura 4 (a) tiene un tamaño similar al código optimizado manualmente ya que el cuerpo de las funciones de bloques de la librería ya están optimizados. Pero, el requisito de memoria de datos es mucho mayor que el código optimizado a mano debido a los requisitos del buffer en los arcos a y b (ambos de tamaño de trama de 144x176). El modelo FRDF posibilita la conversión de tipo implícita desde el tipo de trama de video al tipo de bloque macro en el arco de entrada del bloque ME (Figura 4 (b)). La tasa fraccionada 1/99 señala que el bloque macro es 1/99 de la trama de video en cuanto a tamaño. Una gráfica FRDF puede sintetizarse en códigos eficientes con latencia más corta y menos memoria de buffer que la gráfica del flujo de datos de tasa entera. El requisito de buffer del Modelo FRDF es aun más pequeño que el código de referencia.

3.2.2 Modelo FSM extendido e Interacción con el Modelo SPDF.

PeaCE también posee un modelo extendido FSM, llamado FSM flexible (fFSM), para expresar el comportamiento de transición de estados en una tarea de control. El modelo fFSM soporta concurrencia, jerarquía,

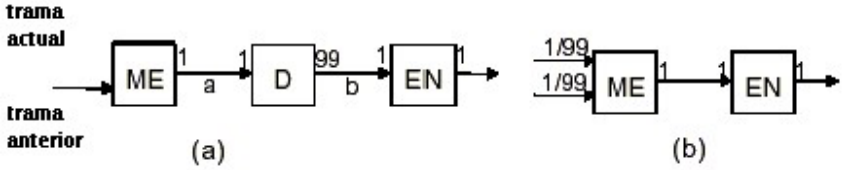

Figura 4: Ejemplo de una sub-gráfica de un codificador H.263: (a) representación SDF (b) representación FRDF.

y mecanismo de eventos internos como lo hace Harel's statechart [11]. A diferencia de Statechart [12], sin embargo, está prohibida la transición de Inter-jerarquía para hacer el modelo más modular y composicional. Un fFSM es la unidad básica de implementación como módulo software o módulo hardware.

Para definir la interacción entre modelos de "flujo de datos" y "FSM", se clasifican patrones de interacción en dos categorías: Una es interacción sincrónica y la otra es interacción asincrónica. En una interacción sincrónica, una tarea de control y una tarea DSP se comunican entre si intercambiando muestras de datos, esto se especifica explícitamente por un arco dirigido entre dos tareas en el modelo de *tarea de mas alto nivel*. Usando interacciones asincrónicas, un bloque de control juega el papel de un módulo supervisor para manejar el estado de ejecución de tareas DSP.
Se definen tres estados de una tarea de "flujo de datos" por su actividad actual: run (ejecutar), suspend (suspender) y stop (detener). Otra situación de interacción asincrónica ocurre cuando el bloque de control quiere cambiar un parámetro de un nodo de flujo de datos asincrónicamente. Una interacción asincrónica se especifica de manera parecida al statechart, usando la acción de escritura en un estado. La acciones de escritura son ejecutadas sólo cuando ocurre la transición de estados. De esta manera PeaCE soluciona el segundo problema.

3.2.3 Modelo de Tareas en PeaCE para especificación a nivel tarea.

Las tareas en el codificador/ decodificador H.263 tienen varias condiciones de activación y semántica de puertos que deberían ser claramente especificadas en el modelo de especificación a nivel tarea en el más alto nivel. En el modelo de tarea propuesto, se soportan tres tipos de tareas; las tareas periódicas disparadas por tiempo, las tareas esporádicas disparadas por I/O externas, y las tareas de función disparadas por datos. Y también se definen diversas semánticas de puertos definidas por tipo de puerto (cola o buffer), tamaño de datos (estáticos o variables) y tasa de datos (estática o dinámica). Por ejemplo, las semántica de puerto de datos para una tarea DSP especificada por una gráfica SDF internamente está clasificada como una cola de tamaño-estático de tasa-estática y el puerto de datos de una tarea FSM como una cola de tamaño-estático de tasa-dinámica.

El comportamiento interno de una tarea se representa como un modelo de flujo de datos o un modelo FSM formando una composición jerárquica de dos modelos de computación diferentes como en Ptolemy. Como un mecanismo de interacción entre el modelo interno y el modelo de tarea exterior, se propuso una aproximación de “envoltura de tarea” donde una envoltura de tarea se crea en el límite del nodo jerárquico (o un nodo de tarea) que traduce la semántica de ejecución exterior a la semántica de ejecución interna . Por ejemplo, si una tarea de flujo de datos está definida como una tarea periódica con buffer de puerto de tasa-estática de tamaño-estático, entonces la llegada de datos de entrada no activa la tarea mientras los datos llegados son almacenados en el buffer del puerto. Y los valores actuales almacenados en los buffers del puerto de entrada son enviados al interior en el accionar periódico. Tal traducción es el rol de la envoltura de tarea, y así es como PeaCE soluciona el tercer problema.

3.2.4 Puerto DRVS (tamaño variable y tasa dinámica)

Para solucionar el cuarto problema, se deja el tipo de puerto de tamaño-variable y tasa-dinámica (DRVS) en la tarea de flujo de datos. En el caso que una tarea tenga un puerto DRVS, se inicia la ejecución después de recibir un cierto número de muestras de entrada, y se suspende en la mitad de la ejecución si este necesita más muestras. Por consiguiente, los datos del puerto de entrada en un decodificador de MP3 se especifican como tipo DRVS.

4 . APLICACIÓN

A continuación se presentan los pasos mas sigificativos de diseño de un modulador QPSK, aplicando las metodologías de Codiseño mediante el entorno PeaCE. (Figura 5)

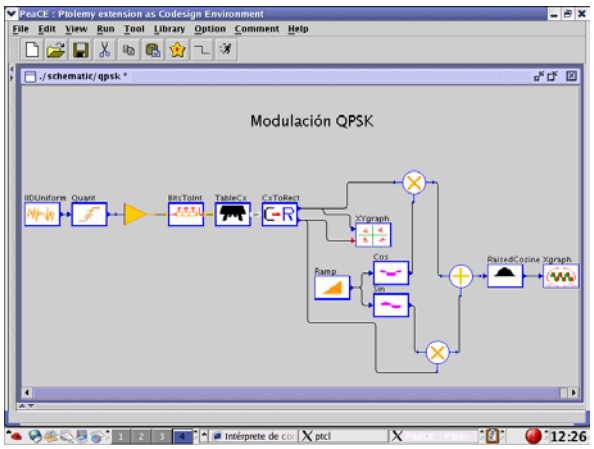

Figura 5: modulador QPSK

El bloque fuente es un generador aleatorio, que entregará las señales a modular; las mismas son cuantizadas mediante un bloque cuantizador “Quant” y luego en el bloque “BitsToInt” agrupa los datos en

palabras de 2 elementos. Estas palabras entran al bloque "TableCX", donde son mapeadas de la siguiente manera:

00 se mapea como (I, Q) ≡ (-1,-1)
10 se mapea como (I, Q) ≡ (1, -1)
01 se mapea como (I, Q) ≡ (-1, 1)
11 se mapea como (I, Q) ≡ (1, 1)

I = Canal en Fase
Q = Cañal en cuadratura

Si los datos no cambian de un periodo al siguiente, la fase de la portadora no cambia. Si hay un cambio de un bit, la portadora es desfasada 90º. Si ambos bits cambian, la fase de la portadora cambia 180º.

Luego del mapeo, el bloque "CxToRect" distribuye parte imaginaria por un a lado y la parte real por el otro.

Por último los valores de I y Q se multiplican por cos $w_c t$ y sin $w_c t$ respectivamente. La salida del modulador va a un circuito sumador donde se combinan ambas señales:

IQ = 11 → cos $w_c t$ + sin $w_c t$
IQ = 10 → cos $w_c t$ - sin $w_c t$
IQ = 01 → - cos $w_c t$ + sin $w_c t$
IQ = 00 → - cos $w_c t$ - sin $w_c t$

El bloque "Xgraph" grafica la señal modulada en QPSK y el bloque "XYgraph" grafica la constelación de la modulación QPSK.

La Ejecución de este sistema da como resultado la constelación QPSK, la gráfica de la señal modulada en QPSK y un archivo que contiene código C generado automáticamente por el sistema (Figura 6).

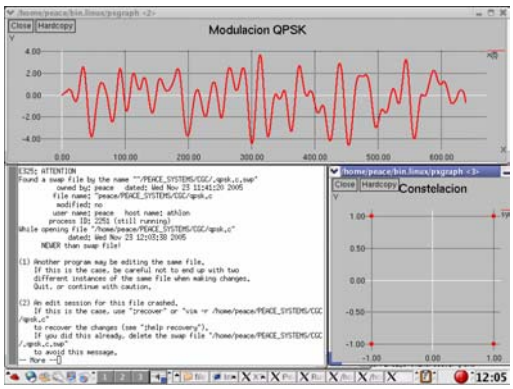

Figura 6: Ambiente de trabajo

Con este código se programa al microcontrolador, lográndose la implementación física del sistema.

Para la traducción al código "C" se utilizó el Software ICC08 [14] (Figura 7) con el que se obtuvo el "código ensamblador", que a través del Software ICS08 WinIDE [15] (Figura 8), permite la programación del componente elegido de la familia HC908.

Logrando, finalmente la implementación de un prototipo, el cual permite realizar pruebas experimentales.

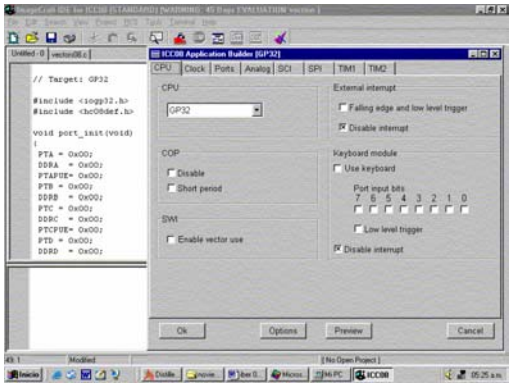

Figura 7: ImageCraft Development Environment

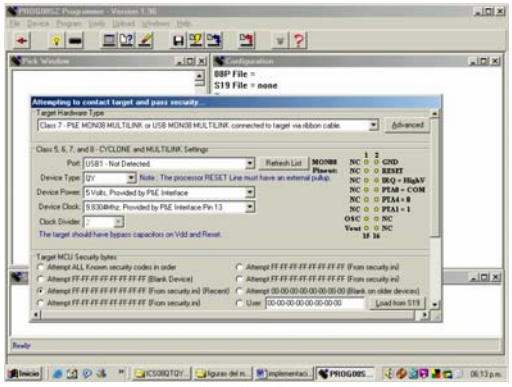

Figura 8: ICS08 WinIDE

Operando de manera similar, se obtiene código 'VHDL', implementandose el diseño en FPGA.

5. CONCLUSIÓNES

Los resultados de la aplicación realizada demuestran las ventajas del ambiente de codiseño PeaCE, para ser utilizado con propositos de investigación y generar prototipos funcionales en un corto tiempo, permitiendo:

- Crear sus propios bloques o hacerle modificaciones al programa en si. Esto es posible dado que se basa en programación orientada a objetos [10].
- Permite la especificación del comportamiento del sistema con una

composición heterogénea de tres modelos de computación: *SPDF* para las tareas de procesamiento de señal, *fFSM* para las tareas de control, y el *Modelo de tarea* de mas alto nivel.

- Se logra la validación fácil del diseño, el análisis estático y el principio de "construcción correcta".

En definitiva, facilita todos los pasos de Codiseño desde la especificación del sistema hasta el prototipado. Obteniendose resultados adisionales como:

- La estimación de performance de software.
- La selección de componentes y el mapeo
- La Co-simulación con exactitud en tiempo y estimación de performance manejada por señales.

REFERENCIAS:

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

[2] User´s Manual: *Ptolemy User´s Manual*, Version 0.7, Linux, University of California at Berkeley, College of Engineering, Department of Electrical Engineering and Computer Sciences, 1997.

[3] Guillermo Gastaldi, "Diseño de un medidor de impedancia en tiempo real", UNLP, Departamento de Electrotecnia, Laboratorio CeTAD, marzo, 2002.

[4] Franke, D.W. and Purvis, M.K.: Hardware/Software Codesign: A perspective, 13th Int. Conf. on Software Engineering, IEEE CS Press, Los Alamitos, California,1991, pp. 344- 352.

[5] 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.

[6] E. A. Lee and A. Sangiovanni-Vincentelli, "A Framework for Comparing Models of Computation," IEEE TCAD, 17(12), pp. 1217-1229, December, 1998.

[7] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems," Int. Journal of Computer Simulation, special issue on Simulation Software Development, vol. 4, pp. 155-182, April, 1994.

[8] "H.263 Video Coding", *http://www.4i2i.com/h263_video_codec.htm*

[9] C. Park, J. Chung, and S. Ha, "Extended Synchronous Dataflow for Efficient DSP System Prototyping," Design Automation for Embedded Systems, Kluwer Academic Publishers, pp.295-322, March 2002.

[10] H. Oh and S. Ha, "Fractional Rate Dataflow Model for Memory Efficient Synthesis," Languages, Compilers, and Tools for Embedded Systems (LCTES), 2002.

[11] Martin Glinz, "Statecharts For Requirements Specification – As Simple As Possible, As Rich As Needed", Institut für Informatik, Universität Zürich Winterthurerstrasse 190 CH-8057 Zurich, Switzerland.

[12] "Statechart web Page" , *http://www.2003canadagames.ca/statechart.html*

[13] "PeaCE: Codesign Environment" , *http://peace.snu.ac.kr*

[14]"Embedded C Development tools, " http://www.imagecraft.com, 1994.

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

.