Instituto Tecnológico de Costa Rica

Vicerrectoría de Investigación y Extensión

Dirección de Proyectos

Informe Final de Proyecto de Investigación

Documento I

Establecimiento de un marco de desarrollo para algoritmos de procesamiento de audio y video en plataformas embebidas

5402 1360 2801

Adscrito a:
Escuela de Ingeniería Electrónica

Investigador principal:
Dr. José Pablo Alvarado Moya, EIE

Estudiantes asistentes:
Michael Grüner Monzón
Bayron Pérez Vega

25 de abril, 2011

Agradecimientos

Este proyecto a sido apoyado financieramente por RidgeRun Engineering y Texas Instruments, quienes donaron el hardware necesario y los recursos económicos para los estudiantes asistentes. En el Tecnológico de Costa Rica el tiempo del investigador fue apoyado por la Escuela de Ingeniería en Electrónica.

Los autores quieren agradecer particularmente a Todd Fischer, Esteban Zúñiga y a Marta Peraza de la empresa RidgeRun Engineering por su continuo aporte en las labores tanto técnicas como administrativas del proyecto.

Resumen

Proyecciones indican que en 2010 se alcanzaron tres sistemas embebidos por cada ser humano en el planeta, y el mercado sigue creciendo. El caso particular de aplicaciones embebidas que utilizan procesamiento digital de señales es de particular interés para Costa Rica, puesto que las universidades locales están en total capacidad de preparar a los ingenieros requeridos y este nicho en el mercado de servicios de ingeniería tiene un alto valor agregado.

Este informe presenta los resultados de una experiencia de colaboración entre RidgeRun Engineering, una compañía estadounidense instalada en Costa Rica y el Tecnológico de Costa Rica (TEC), una universidad estatal costarricense, quienes decidieron trabajar conjuntamente en un plan para mejorar las destrezas y habilidades de estudiantes de ingeniería en el área de implementación de algoritmos embebidos para el procesamiento de señales.

Información y conocimiento sobre tecnología de punta ha sido transferida de RidgeRun al TEC y a cambio documentación detallada sobre cómo realizar aplicaciones ha sido generada en el TEC. Esta documentación permite a los estudiantes sin ninguna experiencia en Unix o diseño e implementación de algoritmos de procesamiento digital de señales e imágenes, producir al final de un semestre aplicaciones para una plataforma Zoom EVM basada en un sistema-en-chip OMAP-L138 de Texas Instruments.

Con la implementación de tres algoritmos de procesamiento de audio como demostración de capacidades del sistema se ha dado inicio a la creación de una biblioteca de algoritmos para el procesamiento digital de señales. El TEC así ha creado las capacidades necesarias para futuras colaboraciones en el desarrollo de algoritmos avanzados.

Abstract

Projections indicate that in 2010 there were three embedded devices for each human being on Earth, and the market keeps growing. The particular case of embedded applications that use digital signal processing is of special interest for Costa Rica, since the engineers required are being already trained at the local universities and the engineering services market has a high added value.

This work describes the experience of a collaboration project between RidgeRun Engineering, an US-American company installed in Costa Rica and "Tecnológico de Costa Rica" (TEC), a Costa Rican State University, who decided to work together on a plan to improve the skills of the students in the embedded DSP algorithm implementation.

The know-how has been transfered from RidgeRun to TEC, and in response detailed documentation on all necessary steps has been generated. This documentation allows students without any experience in Unix nor in DSP algorithm design to produce at the end of one semester applications running in an Zoom EVM based on Texas Instruments' OMAP-L138 heterogeneous system-on-chip.

Índice general

Índice de figuras

Índice de abreviaturas

ALSA *Advanced Linux Sound Architecture*

API Interfaz de programación de aplicación (*Application Programming Interface*)

CCS Code Composer Studio

DSP Procesador Digital de Señales (*Digital Signal Processor*)

FOSS Software libre y de código abierto (*Free and Open Source Software*)

GPP Procesador de Propósito General (*General Purpose Processor*)

GUI Interfaz Gráfica de Uso (*Graphical User Interface*)

IDE Entorno Integrado de Desarrollo (*Integrated Development Environment*)

NFS Sistema de Archivos en Red (*Networking File System*)

PDI Procesamiento digital de imágenes

PDS Procesamiento digital de señales

RTOS Sistema Operativo de Tiempo Real (*Real Time Operating System*)

SDK Kit de desarrollo de software (*Software Development Kit*)

SoC Sistema en Chip (*System on Chip*)

TI Texas Instruments

VISA Video Image Speech Audio

XDAIS eXpressDSP Algorithm Interoperability Standart

XDM eXpressDSP Digital Media

1. Introducción

Sistemas embebidos son actualmente ubicuos, debido a que los microcontroladores se utilizan dentro de prácticamente todo dispositivo sofisticado abarcando desde equipo médico hasta sistemas de aviación, desde satélites espaciales producidos por unidad hasta los sistemas de consumo masivo incluyendo teléfonos celulares, televisores, consolas de videojuegos, PDA, etc. En 2008 cerca del 90% de todos los dispositivos computacionales corresponden a sistemas embebidos [10]. Proyecciones indicaron que en 2010 el número de dispositivos embebidos alcanzaría 16 mil millones (casi tres dispositivos por cada ser humano en el planeta) e indican que se alcanzarán 40 mil millones de dispositivos para 2020 (de 5 a 10 dispositivos por ser humano). Las áreas de Procesamiento Digital de Imágenes (PDI) y Procesamiento Digital de Señales (PDS) progresan rápidamente y proveen los fundamentos para las aplicaciones mencionadas anteriormente. Los costos asociados con el desarrollo de sistemas PDS embebidos son altos, debido en parte a los altos costos de licencias de software especializado y las costosas habilidades y destrezas de ingeniería requeridas, costos que se escalan con los tiempos requeridos para la culminar un producto maduro para el mercado.

Las compañías que producen microcontroladores y sistemas en chip (SoC) tienen estrategias agresivas para hacer el uso de sus productos atractivos. Eso incluye proveer herramientas y bibliotecas que reduzcan el tiempo para lanzar un producto al mercado, así como incorporar nuevas funcionalidad tecnológicas. Una alternativa atractiva para estas compañías es apoyar el desarrollo de software libre y abierto (FOSS) que simplifique la implementación de aplicaciones particulares, mejora el mantenimiento del software y acelera los procesos de depuración, prueba y validación exponiendo el código a una comunidad de desarrolladores de software. Adicionalmente, el código fuente se encuentra siempre disponible para ser revisado y como ejemplo de código real para las fases de entrenamiento de nuevos desarrolladores.

Por otro lado, el uso de FOSS en educación, investigación y extensión tiene varias ventajas incluyendo costos reducidos en licencias de software, y una transferencia más simple de resultados de investigación en la docencia y la extensión. Adicionalmente, FOSS ayuda a posicionar las universidades en la comunidad internacional, lo que a su vez permite atraer financiamiento para apoyar nuevas iniciativas de investigación y desarrollo. Finalmente, FOSS ofrece una base común a la industria y a la universidad para integrar cooperativamente esfuerzos de desarrollo de software en un campo de común interés.

1.1 Razones para la colaboración industria-universidad en Costa Rica

Se estima que más del 70% de los ingresos de procesadores de señal está relacionado con el sector de comunicaciones, con una fracción principal asociada a los teléfonos móviles. Se espera que el mercado de estos procesadores crezca en otros segmentos de la electrónica

de consumo y en el sector industrial, particularmente la electrónica médica, en la cual existe una demanda creciente de innovación en los diseños [3].

Nuevos productos de uso doméstico, móviles o vehiculares se caracterizan por gráficos y audio refinados y capacidades de manejo de datos, lo que aumenta las demandas en las prestaciones del hardware para ejecutar algoritmos PDS. En [3] se estima que el volumen del mercado actual de 1500 millones de procesadores digitales de señal (DSP) (US$5300 millones) crecerá en los próximos cinco años a una tasa promedio de 11% por año.

Los anteriores hechos justifican por qué iniciativas como ARTEMIS [18] en Europa han sido creadas para integrar las estrategias de desarrollo para sistemas embebidos entre los diversos sectores involucrados, incluyendo a las universidades y a la industria.

En Costa Rica los sectores involucrados en el desarrollo de sistemas embebidos crecen. Compañías como HP/ProCurve, Avionyx, Teradyne, Softtek, InbandSoft Canam Technologies y RidgeRun Engineering, entre otras, tienen ya procesos de desarrollo de sistemas embebidos en curso, algunos de los cuales necesitan métodos avanzados del procesamiento digital de señales e imágenes. También es conocido que Intel realiza esfuerzos para posicionar su plataforma Atom en el mercado de sistemas empotrados, lo que indudablemente alcanzará su planta en Costa Rica.

Esto abre la oportunidad a las universidades costarricenses de establecer iniciativas de colaboración con la industria para apoyar la investigación académica que encuentre aplicación directa en los productos en desarrollo. Al mismo tiempo dicha investigación provee el ambiente académico donde futuros ingenieros y estudiantes de postgrado desarrollan las habilidades y destrezas requeridas por la industria. Este tipo de vínculos son promovidos en el ámbito internacional como una manera de atar efectivamente la creación de conocimiento en la academia con el desarrollo de productos en la industria; de esta manera las universidades contribuyen directamente a la economía local [17].

1.2 Objetivos y estructura del informe

Este trabajo presenta los resultados de un proyecto conjunto entre el Tecnológico de Costa Rica (TEC) y RidgeRun Engineering durante el año 2010, quienes iniciaron una actividad de colaboración para aumentar las habilidades y destrezas de futuros ingenieros en los campos de sistemas empotrados y DSP y adicionalmente buscar una base común para facilitar investigación futura para la innovación industrial.

El costo del desarrollo de aplicaciones con algoritmos avanzados de procesamiento digital de señales e imágenes (PDS/PDI) en sistemas embebidos está ligado a los costos de herramientas de hardware y software así como bibliotecas de software especializadas. Los presupuestos limitados de las universidades estatales nacionales limitan la adquisición de tales herramientas. Las bibliotecas de código propietario cerrado dificultan o incluso imposibilitan la modificación de algoritmos existentes para que se ajusten a los requisitos específicos de investigaciones académicas o del desarrollo de productos.

El presente proyecto persigue la generación de conocimiento en la Escuela de Ingeniería Electrónica sobre la cadena de desarrollo completa para aplicaciones avanzadas de PDS/PDI en sistemas embebidos, haciendo dicho conocimiento disponible para actividades de docencia e investigación en la universidad. Puesto que el aporte del TEC en la actividad a provenido de la Vicerrectoría de Docencia, es en esta área donde se ha dado prioridad, sin perder de vista la futura aplicación en investigación formal.

Los entregables se plantean en forma de software abierto bajo la licencia BSD y documentación disponible libremente. Esto tiene las siguientes ventajas:

1. el TEC mejora las capacidades de sus alumnos, quienes son entrenados en los complejos detalles del desarrollo e implementación de las aplicaciones embebidas y el diseño de algoritmos PDS/PDI.
2. el TEC puede de brindar cursos de actualización a sus alumnos y otros ingenieros que requieran desarrollar aplicaciones PDS/PDI embebidas.
3. La visibilidad internacional de RidgeRun y el TEC aumenta a través de la documentación y software FOSS.
4. La industria puede financiar actividades de investigación en las áreas de PDS/PDI al reducirse los tiempo de respuesta del TEC a niveles competitivos utilizando estándares industriales. Estudiantes de todos los niveles académicos pueden encontrar posibilidades de investigación como parte de estos proyectos.
5. La cooperación industria-universidad en el área PDS/PDI empotrado mejora las posibilidades de encontrar tecnologías con propiedad intelectual explotable.
6. De particular interés para el TEC: la industria provee hardware de punta y la infraestructura de software requerida.

Así, el objetivo general del proyecto ha sido establecer la cadena de desarrollo para algoritmos avanzados de PDS/PDI en sistemas embebidos basados en el SoC OMAP-L138 de Texas Instruments. Se persiguió en este primer año realizar dicha cadena enteramente con software libre y abierto, de modo que sea posible instalar todo el software de desarrollo necesario sin restricciones por licencias de software propietario, que tienen asociado costos fuera del alcance de la mayoría de estudiantes (alrededor de US$ 2000 por licencia) y los obligaría a trabajar de forma presencial.

Como objetivos específicos se planteó:

1. Revisar y documentar todos los pasos requeridos para implementar un algoritmo PDS/PDI en un sistema empotrado siguiendo el marco eXpressDSP de Texas Instruments.
2. Diseñar e implementar una biblioteca de software para aplicaciones PDS/PDI siguiendo el estándar eXpressDSP.
3. Diseñar un marco de verificación para los algoritmos de la biblioteca.
4. Diseñar un marco de medición de desempeño de los algoritmos de la biblioteca.

En el planteo de estos objetivos no se consideraron tareas de complejidad no despreciable para el desarrollo del proyecto, lo que obligó a agregar nuevas tareas, como el establecer

el ambiente de desarrollo abierto para la plataforma embebida utilizada, y la salida al mercado en setiembre de 2010 por parte de Texas Instruments de un marco de trabajo completamente nuevo complementario al eXpressDSP. En el momento de preparación de la propuesta original para este proyecto, este nuevo marco de trabajo no estaba disponible ni era conocido pero su papel para este proyecto es prioritario, por lo que su estudio debió integrarse en el proyecto.

Como estrategia metodológica para documentar y detectar todos los detalles de implementación requeridos se seleccionaron dos estudiantes asistentes sin ninguna experiencia previa con sistemas empotrados o con desarrollo de algoritmos PDS, quienes en su proceso de aprendizaje debieron documentar todo obstáculo encontrado en el camino.

Con toda la experiencia documentada es entonces posible continuar el proyecto pues el proceso de aprendizaje para nuevos asistentes se acelera considerablemente.

Este informe está estructurado de la siguiente manera: El próximo capítulo revisa la documentación técnica asociada a la implementación de aplicaciones para PDS/PDI en arquitecturas híbridas. El capítulo 3 presenta las tareas realizadas para alcanzar los objetivos general y específicos, y los resultados se sintetizan en el capítulo 4. El informe finaliza con un resumen de conclusiones en el capítulo 5 y un listado de los aportes en el capítulo 6.

2. Estado del arte

El objetivo de este proyecto ha sido establecer la cadena de desarrollo para algoritmos avanzados de PDS/PDI en sistemas embebidos basados en el SoC OMAP-L138 de Texas Instruments. Esto permite contar tanto con plataforma de desarrollo de hardware y software como con de la base de conocimiento sobre la que se podrá desarrollar investigación formal en un futuro cercano.

La revisión bibliográfica se ha orientado a la descripción de herramientas tecnológicas y documentos técnicos sobre los componentes de hardware y software utilizados, que por el nivel de la tecnología utilizada son relativamente complejos.

2.1 Plataforma de desarrollo

Texas Instruments cuenta con un programa universitario [16] en donde en colaboración con universidades brinda sugerencias para abordar diversos de temas incluyendo sistemas embebidos y procesamiento digital de señales en niveles básico, intermedio y avanzado. Los programas allí propuestos utilizan al Code Composer Studio [15] como herramienta de desarrollo principal, u otras herramientas propietarias como Matlab de Mathworks [20, 25] o LabVIEW de National Instruments [11]. Estas herramientas de software propietario tienen elevados costos para estudiantes locales y para la universidad, lo que limita su uso generalizado. Algunas de ellas pueden ser utilizadas en laboratorios del TEC, puesto que allí se cuenta con las licencias, ya sea donadas (como es el caso del Code Composer Studio) o adquiridas bajo convenios universitarios; sin embargo, la transferencia a la industria local iría acompañada por costos aún más elevados por licencias de software propietario.

El actual proyecto ha buscado alternativas abiertas para el desarrollo que no estén atadas a dichos costos, y que aseguren las libertadas ofrecidas por el software libre y abierto (FOSS) [26]. Esto tiene claras ventajas para el desarrollo de algoritmos en investigación, la incorporación de los resultados a tareas de docencia y la cooperación directa con la industria en el desarrollo innovador de productos.

2.2 Bibliotecas especializadas de TI

Este proyecto ha arrancado con la premisa de elaborar una biblioteca de software especializado en aplicaciones embebidas de PDS/PDI. Como puede suponerse, ya Texas Instruments provee bibliotecas especializadas para este contexto, cuya revisión fue necesaria para delimitar los alcances del proyecto y no replicar trabajo.

La primera biblioteca es la llamada DSPLIB (*DSP Library*) [28, 31] que contiene algoritmos en C optimizados para microcontroladores DSP de Texas Instruments. Estas bibliotecas ofrecen rutinas básicas para aplicaciones computacionalmente demandantes

con restricciones de tiempo real e incluyen filtros adaptativos, correlación, transformada rápida de Fourier, filtros FIR e IIR, productos y transposiciones matriciales, etc. Arquitecturalmente las funciones ofrecidas por esta biblioteca se encuentran en un plano de abstracción puramente de procesamiento digital, es decir, no han sido ideadas para ofrecer comunicación entre procesadores o procesos, sino simplemente para aplicar algoritmos concretos a datos disponibles directamente en memoria. La biblioteca a desarrollar debe ofrecer entonces funcionalidades a un nivel superior de la arquitectura de software, particularmente en plataformas heterogéneas como la OMAP-L138.

Para el procesamiento digital de imágenes la biblioteca IMGLIB [27] (*DSP Image/Video Processing Library* brinda al mismo nivel arquitectural de la DSPLIB con algoritmos frecuentemente utilizados en tareas de compresión, procesamiento de video, vision industrial e imágenes médicas. En concreto se brindan algoritmos de morfología matemática [8] como dilatación y erosión, generación de histogramas, aproximación de gradientes como el sobel, filtros de mediana y convoluciones para máscaras de 3×3, entre otros.

Las anteriores bibliotecas aportan algoritmos frecuentemente utilizados que se han optimizado para toda la familia de microcontroladores DSP de Texas Instruments, lo que brinda una capa de abstracción de hardware, cuando lo que se requieren son estos algoritmos básicos. Se persigue que la nueva biblioteca propuesta en este proyecto aproveche las prestaciones disponibles del hardware, sin tener la necesidad de trabajar a nivel de ensamblador. Para ello Texas Instruments ofrece los llamados operadores intrínsecos [2, 19] en donde una capa en C abstrae de forma optimizada operaciones ofrecidas por el microcontrolador DSP. Estos operadores deben preferirse sobre el uso del ensamblador pues permiten portar el código desarrollado a otras plataformas de hardware sin mayor complicación.

2.3 eXpressDSP

La figura 2.1 muestra un diagrama simplificado de los componentes involucrados en el desarrollo de aplicaciones para las arquitecturas heterogéneas de Texas Instruments. El concepto está orientado a asegurar la interoperabilidad del software, necesaria para reducir costos de desarrollo de aplicaciones de procesamiento de señales. Los módulos de procesamiento implementados con este estándar por un *productor*, y llamados en la jerga propia de este marco de desarrollo como *algoritmos* (por ejemplo codificadores multimediales) son integrados fácilmente por los *clientes* en sus aplicaciones, independientemente del modelo exacto de SoC utilizado, y sin tener que lidiar con aspectos internos de los algoritmos o su código fuente. Un aspecto comercial relevante es que el marco eXpressDSP permite esconder del cliente todos los aspectos de implementación que quizá el productor quiera proteger al vender un módulo de procesamiento (o algoritmo) como producto.

Este marco eXpressDSP [29] permite a una aplicación que ese ejecuta en el GPP utilizar las capacidades computacionales del microcontrolador DSP por medio de varias capas conceptuales especializadas cada una en necesidades particulares. Algunas capas son

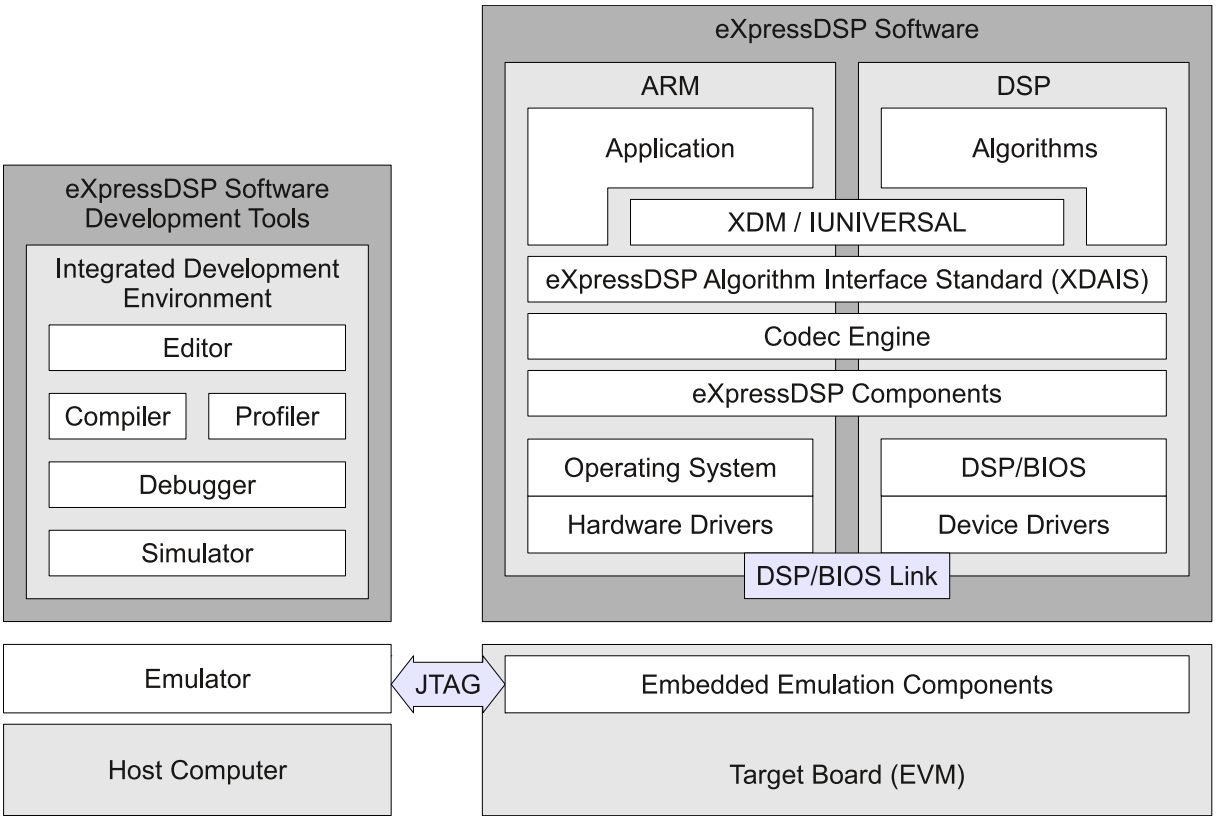

Figura 2.1: Marco de desarrollo eXpressDSP de Texas Instruments [29]

agnósticas a la aplicación (como XDAIS) mientras que otras (como XDM) se especializan en algoritmos de vídeo, imagen, habla y audio (VISA). El concepto completo tiene más componentes y subcomponentes que los ilustrados aquí.

Aunque las ventajas de este marco son claras, personas con poco conocimiento en las técnicas avanzadas de ingeniería de software así como sin experiencia previa en desarrollo de algoritmos PDS/PDI necesitan de dos a tres meses para comprender y manejar los conceptos involucrados y crear todos los archivos con código fuente, rutas, Makefiles, configuradores, etc. Aun cuando el Code Composer Studio [15] simplifica estas tareas, recuérdese que el objetivo del proyecto se orientó a ofrecer una metodología abierta de desarrollo.

2.3.1 La interfaz XDAIS/XDM

Las regulaciones de eXpressDSP se engloban en el llamado XDAIS ("eXpressDSP Algorithm Interoperability Standart"). Éste consiste en un conjunto de reglas y guías las cuales aseguran la correcta convivencia de varios algoritmos dentro de un mismo sistema, independientemente de su procedencia. Para este propósito distintas interfaces de programación de aplicaciones (API) se proporcionan al desarrollador para facilitar el cumplimiento de dichas reglas. Por ejemplo, IALG determina cómo reservar y manejar la memoria utilizada por un algoritmo, IRES define cómo solicitar recursos e IDMA3 define

cómo utilizar el acceso directo a memoria.

XDM (eXtremeDSP Digital Media) extiende a XDAIS para casos en los que la aplicación desarrollada es estrictamente de multimedia. Esta API pone a disposición interfaces designadas como VISA ("Video", "Imaging", "Speech" y "Audio"), y por sus características de interoperabilidad han sido acogidas por la industria. Similar a XDM, IUNIVERSAL [14] es una API que extiende XDAIS para permitir el desarrollo de algoritmos genéricos (no multimediales).

Para este proyecto se prestó especial atención a IUNIVERSAL por su flexibilidad de implementar algoritmos no convencionales (esto es, algoritmos que no calzan en XDM), en donde existe mayor potencial de innovación e investigación futura. La figura 2.2 ilustra la secuencia de funciones que deben ser invocadas por un algoritmo implementado con IUNIVERSAL. Las tareas de cada función se describen a continuación.

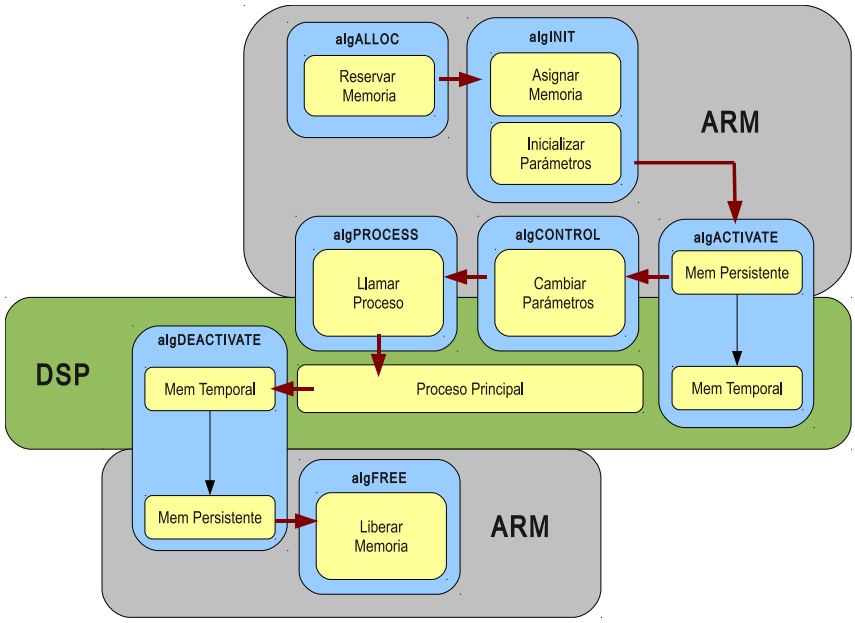

Figura 2.2: Secuencia de pasos ejecutados por un algoritmo IUNIVERSAL.

1. algAlloc especifica la cantidad de memoria que el algoritmo necesita. El programador elige entre distintos niveles de memoria disponibles en el sistema embebido, de acuerdo a las necesidades que la aplicación demande. En el caso del ejemplo desarrollado, se reservó memoria persistente y memoria temporal de trabajo para los búferes anulares del filtro y del reverberador así como para los búferes que contendrán los espectros en el filtrado en el dominio de la frecuencia.

2. algInit inicializa los objetos y demás elementos y asigna los segmentos de memoria previamente reservados. Es aquí donde las variables de atenuación, normalización, reverberación y similares de los filtros implementados son inicializadas.

3. algActivate es llamada automaticamente antes de iniciar el proceso principal del algoritmo, por lo que se utiliza para copiar los datos a procesar de la memoria

persistente a la temporal para aprovechar que el acceso a la segunda es hasta un factor de treinta veces más rápida que la primera.

4. algMoved es llamada automáticamente por la interfaz cuando se necesita mover el algoritmo a otra posición de memoria.

5. algControl permite modificar parámetros en tiempo real. En el caso de los ejemplos implementados, modifica los parámetros de atenuación y de reverberación, así como la amplitud de las bandas en el ecualizador.

6. algProcess contiene el proceso principal del algoritmo. Para el paso de argumentos y búferes de datos IUNIVERSAL establece campos particulares en las estructuras de datos. Para los filtros filtro peine y el reverberador en esta función se realizan las respectivas ecuaciones de diferencias usando búferes anulares en la memoria temporal, reservada anteriormente. Para la etapa de filtrado en frecuencia se hizo uso de las funciones para el cálculo de la FFT (DSPF_sp_fftSPxSP) y la iFFT (DSPF_sp_ifftSPxSP) en la biblioteca DSPLIB de TI para procesadores de la familia C674x.

7. algDeactivate complementa a algActivate y se usa para trasladar los resultados procesados en la memoria temporal a la memoria persistente, de modo que estén disponibles en la siguiente llamada al proceso.

8. algFree permite liberar la memoria reservada por el algoritmo en algAlloc.

2.3.2 La creación de los Codecs

El marco eXpressDSP de Texas Instruments incluye en sus especificaciones cómo se debe empaquetar la implementación de un algoritmo para asegurar la portabilidad del mismo, y la posibilidad de incluir implementaciones para diferentes plataformas. Al paquete se le denomina en este entorno de TI *Codec*, independientemente de la tarea concreta que realiza.

Para empaquetar el algoritmo se usa la herramienta RTSC (Real Time Software Components), que a su vez hace uso de XDCtools. El último es una extensión del lenguaje C y provee un set de instrucciones para la creación, prueba e instalación de los componentes RTSC. Aprovechando la estructura regular que siguen los algoritmos, se proveen al desarrollador asistentes que generan una implementación genérica (o plantilla) con las funciones XDM, siguiendo las especificaciones XDC y listo para ser empaquetado como codec. De esta forma se eliminan las tareas rutinarias. La interpretación del codec se realiza mediante la herramienta "Codec Engine" (CE), más específicamente, se utiliza "CE Algorithm Creator".

Estos archivos de configuración y la utilización de las extensiones de C, entre otros, incrementan el nivel de conocimiento requerido y por tanto la complejidad del proceso de desarrollo. Sin embargo, eXpressDSP brinda la flexibilidad necesaria para acceder

a los diferentes recursos de hardware disponibles en el sistema embebido sin sacrificar portabilidad e interoperabilidad.

2.3.3 La creación del servidor

Si el desarrollo se realiza para una plataforma híbrida como los sistemas en chip (SoC) de la familia OMAP-L138 que contienen procesadores de propósito general y DSP, se debe crear un servidor para manejar las llamadas remotas del algoritmo. Esta herramienta se llama "Codec Server". El servidor se implementa mediante archivos de configuración XDC y por medio de asistentes se genera un servidor genérico que se modifica según corresponda. Utilizando este mismo servidor múltiples codecs pueden ser ejecutados remotamente. Para ello CE utiliza "CE engine integrator". Todos los archivos necesarios para crear estos módulos son automaticamente creados y configurados por los asistentes disponibles.

2.3.4 Dependencias adicionales

El marco de programación hace uso de una capa inferior en el proceso llamada "Framework Components" (FC). Esta capa provee una serie de componentes que facilitan la implementación de XDAIS/XDM. Entre los componentes que conforman FC se puede mencionar DSKT2, DMAN3, RMAN, IRESMAN, ACPY3 y SCPY; todas herramientas para el manejo de recursos como memoria y demás. Además se utiliza DSP/BIOS-Link que provee servicios de comunicación entre el GPP y el DSP, DSP/BIOS que opera como RTOS en el DSP, LPM (Local Power Management) para el manejo de energía en el DSP y Linux Utilities para el acceso a los recursos de hardware. Todas estas herramientas son accedidas automaticamente por las interfaces de Codec Engine [13] y sus API. Sin embargo, todos los recursos anteriores deben de estar instalados y correctamente configurados en la computadora local para poder desarrollar este tipo de software, lo cual agrega complejidad al proceso de instalación.

2.3.5 La aplicación

Una vez que se cuenta con un con un codec y un servidor configurados adecuadamente, se procede a programar la aplicación que ejecuta el codec. Esto se logra implementando una aplicación en C en la que se instancian las funciones IUNIVERSAL y funciones especiales de CE que interacúan con los componentes RTSC.

Programada la aplicación, se utiliza la herramienta "XDC Configuro" [12] para interpretar todos los archivos de configuración escritos anteriormente en el codec y en el sevidor. Seguidamente, el proceso de compilación cruzada y enlazado es llevado a cabo utilizando el interprete suministrado en la cadena de herramientas de "Code Sourcery". Por último se instala la aplicación en el sistema empotrado y está lista para ser ejecutada.

Debe anotarse que en la arquitectura híbrida utilizada (OMAP-L138) el ARM9 utilizado

como GPP tiene controlador de memoria (MCU) para ofrecer las ventajas de memoria virtual; sin embargo, el procesador digital por asuntos de eficiencia de acceso no cuenta con MCU. Esto hace que la memoria en ambos procesadores tiene direcciones distintas. Gran parte de la complejidad en los desarrollos de aplicaciones para estas arquitecturas radica en el manejo de la memoria, en los distintos niveles de acceso, para así pasar eficientemente los bloques entre la aplicación del GPP y los algoritmos de procesamiento del DSP, y en el DSP para pasar entre los distintos niveles de memoria para optimizar la velocidad de los procesos.

2.4 C6Run

Cuando se inició el proyecto la única forma "simplificada" de trabajar con el SoC híbrido era a través del marco eXpressDSP; sin embargo, en setiembre 2010, Texas Instruments propuso un nuevo concepto (a menos de tres meses de la conclusión del primer año del proyecto), pues es bien conocida la elevada complejidad del concepto completo lo que dificulta a nuevos ingenieros alcanzar un nivel de productividad alto cuando se desea respetar dicho marco de trabajo. La nueva herramienta de desarrollo de software, denominada C6Run [1], es ideal para dar los primeros pasos en programación de un DSP, que es de interés en este proyecto, donde se pretende transferir conocimiento y crear habilidades de futuros ingenieros. A C6Run la acompañan las herramientas C6Accel y C6Flo que se sitúan en niveles superiores de abstracción para simplificar aun más el desarrollo.

El marco C6Run no asegura interoperabilidad de módulos ni el encapsulado para protección intelectual de la aplicación, lo que elimina gran parte de las complejidades en la ingeniería de software asociadas a dichas características. Internamente este sistema ha sido diseñado utilizando eXpressDSP, pero ocultándolo de los desarrolladores. Así mismo, C6Run oculta las complejidades asociadas a la comunicación entre el GPP, que tiene MCU, y el DSP que no cuenta con MCU. La memoria se maneja de modo que al usuario le sea relativamente indiferente cómo se comunican los bloques entre los dos procesadores.

El objetivo de C6Run no es reemplazar esquemas establecidos de desarrollo de algoritmos de PDS para sistemas empotrados, sino simplificar el primer contacto con dicho desarrollo a usuarios principiantes, ocultando los marcos de desarrollo avanzados, pero más complejos.

C6Run es un paquete de herramientas que utiliza como entrada archivos fuente escritos en el lenguaje C y genera a partir de ellos programas ejecutables en el procesador de propósito específico que permiten llamar a funciones diseñadas para ser ejecutadas en el DSP.

C6Run posee dos herramientas en particular: C6RunLib y C6RunApp.

1. C6RunLib: permite construir bibliotecas estáticas que se enlazan con una aplicación escrita para el GPP y brindan acceso a funciones ejecutables en el DSP. El desarrollador decide qué funciones forman parte de la biblioteca, y la herramienta las

compila para el DSP, construyendo además la interfaz de invocación para el GPP. La biblioteca genera se vincula al código del usuario de igual forma que cualquier otra biblioteca de código.

2. C6RunApp: permite construir aplicaciones que son ejecutadas en su totalidad en el DSP, debido a que actua como un compilador cruzado para el DSP, pero que construye automáticamente código para el lado del GPP en el que se relegan todas las operaciones de entrada/salida que el DSP no está equipado para manejar.
   La aplicación se construye para ser invocada desde el sistema operativo GNU/Linux ejecutándose sobre el GPP, pero todo el procesamiento en el código es realizado por el DSP.

3. Métodos y materiales

La implementación de una biblioteca completa de software es un proceso que toma años/persona y que usualmente va acompañado de aplicaciones concretas que imponen necesidades algorítmicas particulares. En este primer año no existió una aplicación concreta, sino la necesidad de explorar la arquitectura empotrada y sus prestaciones para implementar algoritmos de PDS/PDI. De este modo, se identificaron tres algoritmos relativamente básicos, pero de complejidad superior a las funcionalidades ofrecidas por las bibliotecas propias de Texas Instruments (ver sección 2.2).

El desarrollo de algoritmos embebidos para el procesamiento digital de señales (PDS) se separa en dos fases de igual complejidad: diseño e implementación. El diseño se relaciona con la especificación matemática del sistema, en el que se debe asegurar el cumplimiento de condiciones tanto para señales como para sistemas en los dominios temporales, de frecuencia u otros. Criterios típicos de diseño involucran limitaciones en los retardos de grupo, respuestas de fase, respuestas en magnitud, longitudes u ordenes de filtros permitidos, etc.

Por otro lado, la implementación está asociada con la manera en que el algoritmo diseñado es realizado, lo que puede involucrar desde un algoritmo en software para un computador de propósito general, hasta un circuito integrado de aplicación específica. Las dificultades de implementación se asocian tanto con el manejo de los recursos de la arquitectura (como por ejemplo los distintos niveles memoria, traspaso de datos, latencias, y disponibilidades de tiempo para los cálculos por muestra), como con la búsqueda de arquitecturas reutilizables que permitan reducir el plazo de comercialización en el desarrollo de productos que utilicen algoritmos PDS.

En el caso del presente trabajo se analizan estrategias de *implementación* en software para una arquitectura embebida basada en un sistema-en-chip (SoC) heterogéneo de Texas Instruments OMAP L138 [30], que incluye una arquitectura ARM9 para el procesador de propósito general y un procesador de señal de arquitectura C674x de TI para las tareas específicas.

La primer estrategia de implementación utilizada se basa en el marco eXpressDSP propuesto por Texas Instruments [29] para permitir la reutilización de código al establecer una interfaz claramente definida (XDAIS, por eXpressDSP Algorithm Interoperability Standart) que permite la interoperabilidad de algoritmos PDS de tiempo real, ofrecidos por diversos fabricantes.

Dicha arquitectura, si bien es cierto permite un elevado control en cuanto al manejo de recursos de comunicación entre los sistemas de propósito general y de PDS, ha crecido en su evolución a tal punto que implica periodos considerables de entrenamiento para ingenieros que incursionan en el desarrollo de algoritmos de este tipo, y que dificultan la revisión y mantenimiento de código ya existente.

La complejidad alcanzada en dichos sistemas ha obligado a proponer nuevas estrategias

que simplifiquen el desarrollo de algoritmos PDS, principalmente cuando no se requiere una reutilización "cerrada" de los códigos. Es en este contexto que se han propuesto las herramientas C6Run, C6Accel y C6Flo para reducir los tiempos de puesta en marcha de algoritmos PDS [1].

La última plataforma de implementación sirve como punto base de comparación y utiliza un computador de propósito general (PC).

3.1 Algoritmos utilizados en el marco de comparación

La biblioteca desarrollada en este trabajo incluye entonces tres algoritmos: un filtro peine de supresión de armónicos de 60 Hz, un reverberador lineal y un ecualizador. La selección de estos tres filtros obedece por un lado a razones didácticas, pues estos deberán ser implementados por estudiantes en un curso introductorio al procesamiento digital de señales: el filtro peine tiene una estructura simple de filtro lineal IIR en el dominio del tiempo, con parámetros constantes que no pueden ser modificados por el usuario. El reverberador es también un filtro lineal IIR, pero con parámetros modificables por el usuario durante la operación del filtro. Finalmente, se ha elegido la implementación de un ecualizador en el dominio de la frecuencia, como ejemplo de uso de la transformada discreta de Fourier y de integración de varios conceptos teóricos fundamentales.

Desde un punto de vista de implementación, los tres algoritmos hacen un uso complejo de la memoria que va más allá de las funcionalidades ofrecidas por la biblioteca DSPLIB.

Adicionalmente a estos algoritmos, en la tesis [32] desarrollada en el marco del proyecto en RidgeRun se exploró la conversión de espacios de color de YCrCb a RGB de un empaquetado Y420 a un RGB565, y otro algoritmo de reverberación pero de naturaleza FIR.

3.1.1 Filtro peine de 60 Hz

Este filtro se utiliza para eliminar el ruido producido por la red eléctrica, que introduce la frecuencia fundamental de 60 Hz y sus armónicos.

La función de transferencia utilizada para el filtro peine es

$$H(z) = \beta \frac{1 - z^{-k}}{1 - \alpha z^{-k}}$$

en donde α regula el ancho de la banda de rechazo y β normaliza la respuesta de modo que ninguna componente espectral sea amplificada. El orden k del filtro permite seleccionar las frecuencias concretas que deben ser suprimidas. La respuesta en magnitud del filtro está dada por

$$|H(\omega)| = \beta \sqrt{\frac{2\cos(k\omega) - 2}{2\alpha\cos(k\omega) - (\alpha^2 + 1)}}$$

Para rechazar la frecuencia de F_0 =60 Hz y sus armónicos se utiliza $k = F_s/F_0$, donde F_s corresponde a la frecuencia de muestreo utilizada en el dispositivo de audio. Para obtener una banda de rechazo B, α se elige de modo que en las frecuencias de corte $F_0 \pm \frac{B}{2}$ se obtenga potencia mitad (la respuesta en magnitud debe ser $1/\sqrt{2}$).

Lo anterior se obtiene con

$$\beta = \frac{1+\alpha}{2} \qquad \alpha = \frac{1-\sqrt{1-\gamma^2}}{\gamma} \qquad \gamma = \cos\left(\frac{\pi B k}{F_s}\right)$$

La ecuación de diferencias que representa el sistema anterior está dada por

$$y(n) = \beta x(n) - \beta x(n-k) + \alpha y(n-k)$$

cuya implementación directa requiere dos búferes anulares de longitud k para almacenar tanto entrada como salida. La Forma Directa II permite expresar la ecuación anterior como:

$$v(n) = x(n) - \alpha v(n-k)$$
$$y(n) = \beta(v(n) - v(n-k))$$

que requiere un único búfer anular de longitud k.

Los parámetros α, β y k se determinan una única vez antes de filtrar las señales en función de F_0, F_s y B, y se mantienen constantes.

3.1.2 Reverberador de Audio

El reverberador de audio utilizado está descrito por

$$H(z) = \frac{1-\alpha}{1-\alpha z^{-k}}$$

Aquí, α representa la constante de atenuación, y debe cumplir $|\alpha| < 1$ para asegurar estabilidad. El retardo de reverberación está determinado por la constante entera k, y equivale a $\tau = k/F_s$. La implementación de este filtro lineal se basa en su ecuación de diferencias

$$y(n) = (1-\alpha)x(n) + \alpha y(n-k)$$

Esta Forma Directa I es suficiente para fines de implementación, pues requiere únicamente un búfer anular de tamaño k.

La dificultad en este filtro se encuentra en cómo diseñarlo de modo que el usuario pueda modificar los parámetros k y α en-línea.

3.1.3 Ecualizador de Audio

Si bien es cierto implementaciones estándar de ecualizadores de audio utilizan bancos de filtros IIR de segundo o tercer orden implementados en el dominio del tiempo, para los

fines didácticos y demostrativos perseguidos se implementó un ecualizador en el dominio de la frecuencia, pues esta técnica es compleja de implementar y más eficiente cuando se utilizan filtros FIR de longitudes superiores a 50 taps.

El punto de partida es un esbozo de la respuesta en frecuencia deseada, dado por 16 puntos del espectro distribuidos logarítmicamente e interpolados linealmente. El principio se esboza en la figura 3.1 para cinco bandas, donde al lado izquierdo se muestra la representación logarítmica utilizada tradicionalmente y al lado derecho la representación lineal, utilizada en la implementación. En tiempo de ejecución el usuario puede cambiar

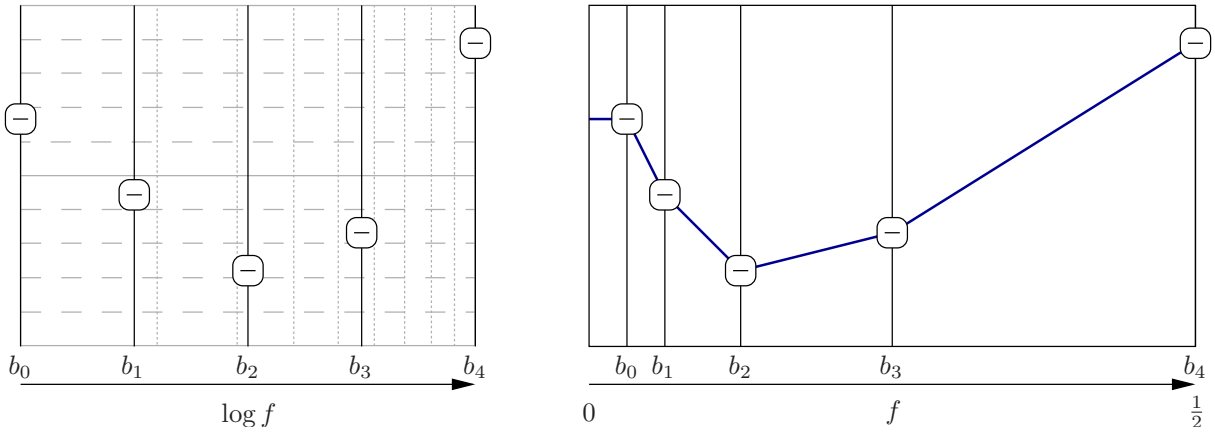

Figura 3.1: Estimación de la respuesta en magnitud para el ecualizador con cinco bandas.

el factor de amplificación/atenuación de cada banda.

La respuesta en frecuencia normalizada teórica abarca un rango de $f = 0$ a $f = 1$, pero puesto que se trabaja con sistemas reales, dicha respuesta presenta simetría hermítica, por lo que únicamente la mitad del espectro se almacena y trabaja. La respuesta en fase debe determina indirectamente en los siguientes pasos.

Sea L el tamaño de cada bloque de muestras de entrada $x_m(n)$ que se va a procesar, y sea M el tamaño del filtro FIR en el tiempo. La convolución de la respuesta al impulso del filtro con el m-ésimo bloque de la señal de entrada produce la salida $y_m(n)$ con longitud $N = L + M - 1$ muestras que se calcula en el dominio de la frecuencia con

$$\hat{y}_m(n) \overset{N}{\circ\!\!-\!\!\bullet} \hat{Y}_m(k) = H(k)X_m(k), \quad k = 0, 1, \ldots, N-1$$

donde $X_m(k)$ corresponde a la transformada discreta de Fourier de la entrada $x_m(n)$ y $H(k)$ es la respuesta en frecuencia del filtro, completadas ambas a N muestras.

Ya sea si se utiliza el método de solapamiento y almacenamiento (*overlap-save*) o el método de solapamiento y suma (*overlap-add*) el bloque de entrada $x_m(n)$ se completa de L muestras a N con las últimas $M - 1$ muestras del bloque anterior (en el primer caso), o con $M - 1$ ceros (en el segundo caso). El filtro se completa de M a N muestras con L ceros.

La respuesta en frecuencia ilustrada en la figura 3.1 tiene N muestras y su correspondiente señal temporal también. Ésta debe modificarse para asegurar que su respuesta equivalente

en el tiempo tenga longitud máxima L. Para ello se utiliza la técnica de enventanado temporal con una ventana de Hanning [24].

El cálculo de $H(k)$ debe hacerse cada vez que el usuario cambie en nivel de alguna banda. En cuanto a los objetivos didácticos, con esta técnica se utilizan los algoritmos FFT e iFFT para el cálculo de la DFT, se deben manejar los conceptos de aliasing, correspondencia entre convolución y producto entre los dominios temporal y de frecuencia, respectivamente, y se deben comprender las implicaciones de causalidad del sistema y de que las señales tengan valores reales para su manejo en el dominio de la frecuencia.

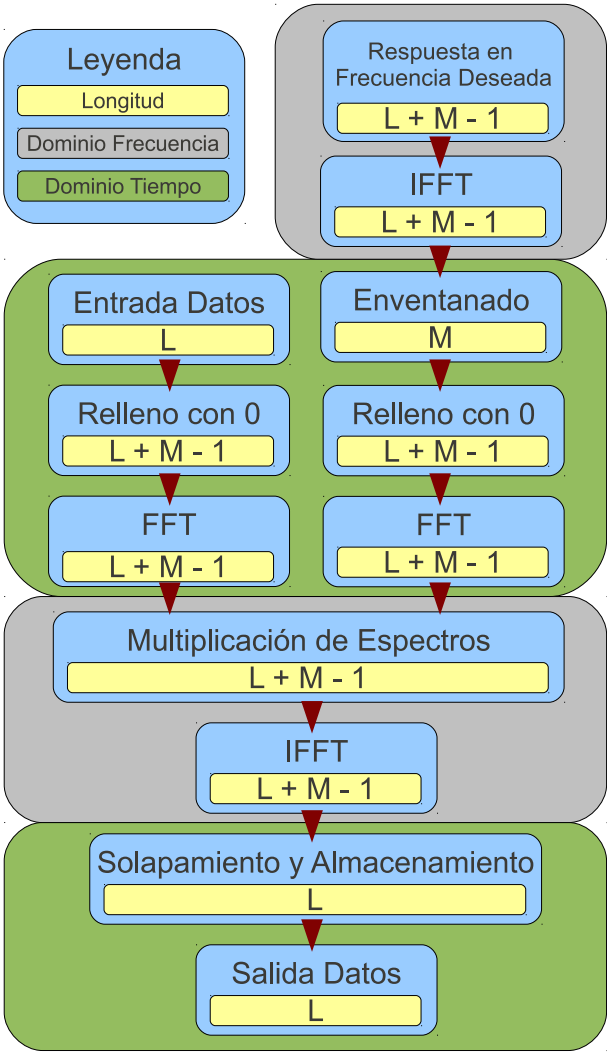

Figura 3.2: Pasos necesarios en el procesamiento en el dominio de la frecuencia de una señal de duración indefinida.

3.2 Fases del proyecto

El desarrollo de aplicaciones embebidas para PDS/PDI involucra varias fases, por las que ha pasado naturalmente el desarrollo del actual proyecto. Puesto que en cualquier futuro

proyecto de investigación, estudiantes de grado y post-grado deben estar en capacidad de manejar todo el proceso de desarrollo, abarcando desde la instalación del entorno de desarrollo (puesto que las actualizaciones de software ocurren con una frecuencia entre dos a cuatro versiones por año), ha sido de interés documentar todas las fases en la que estudiantes pueden tener dificultades.

La metodología de documentación involucró contratar dos estudiantes asistentes que no tuviesen experiencia previa ni con el desarrollo de algoritmos PDS/PDI ni con sistemas embebidos, y se les solicitó describir todos los pasos, especialmente aquellos en los que ellos tuviesen mayores dificultades. El resultado de dicho proceso se encuentra disponible en [9].

3.2.1 Primera fase: selección del hardware y software

Con respecto al hardware, para soportar el desarrollo de actividades docentes y de investigación en el procesamiento digital de señales embebido, se eligió una plataforma versatil: una tarjeta de evaluación Zoom (EVM) basada en el sistema en chip (SoC) OMAP-L138. Este SoC tiene arquitectura híbrica combinando un procesador de propósito general (GPP) ARM9 y un procesador digital de señal (DSP) TI C674x [30]. La elección en este SoC se basa por un lado en el hecho de que TI abarca el 65% del mercado de procesadores de señales digitales, convirtiéndolos en líderes del sector. Adicionalmente la asociación de RidgeRun con Texas Instruments permitió tener acceso al hardware requerido para la colaboración.

La tarjeta Zoom EVM permite el desarrollo de proyectos académicos que exponen al desarrollador a problemas reales de implementación de aplicaciones embebidas y a los retos de la programación eficiente de microcontroladores DSP.

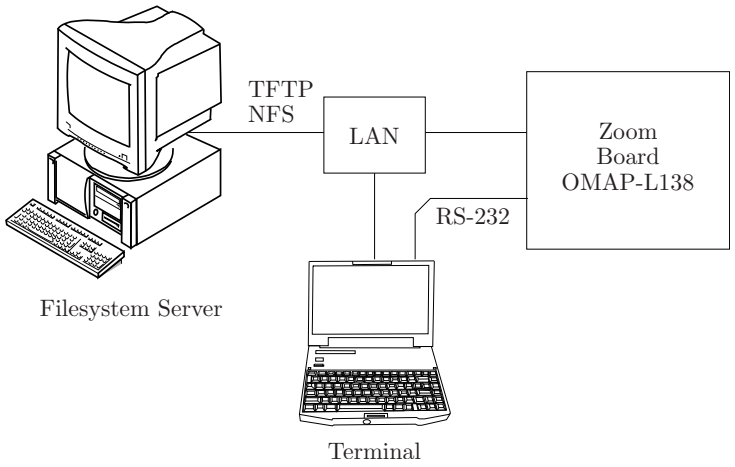

Figura 3.3: Configuración de hardware seleccionada para el desarrollo de software embebido para PDS/PDI.

La figura 3.3 muestra las conexiones de hardware elegidas para el desarrollo de software embebido. Esta es una configuración estándar que permite el desarrollo efectivo de apli-

caciones: en un computador anfitrión se realiza la compilación cruzada para la tarjeta objetivo (en este caso, el OMAP-L138 de la Zoom EVM) y los códigos ejecutables se almacenan directamente en el sistema de archivos montados remotamente vía NFS en la tarjeta embebida. La tarjeta es controlada por el desarrollador utilizando una terminal conectada por puerto serial RS232.

En el laboratorio el servidor de sistema de archivos es compartido entre varias terminales, mientras que en una configuración personal un único computador se desempeña como terminal y como servidor de archivos.

El uso de JTAG y Code Composer Studio se reserva para futuras actividades que prevean el uso de herramientas con licencias propietarias.

Puesto que el proyecto se ha orientado al uso de software libre y abierto, el sistema operativo elegido para el desarrollo de aplicaciones es GNU/Linux, para el cual se dispone de una plétora de herramientas de desarrollo libres y acceso a una comunidad colaborativa de desarrolladores. El kit de desarrollo de software (SDK) de RidgeRun para el OMAP-L138 de TI se eligió para instalar todo el software requerido en la tarjeta de evaluación y en el computador anfitrión.

3.2.2 Segunda fase: configuración del entorno de desarrollo del sistema embebido

Kit de Desarrollo de Software

Desde una perspectiva docente, una de las mayores dificultades de la plataforma de desarrollo es el desconocimiento generalizado por parte de estudiantes sobre sistemas Unix en general y de GNU/Linux en particular. La mayoría de estudiantes están acostumbrados y dependen de Interfaces Gráficas de Uso (GUI) para operar sus computadores en sistemas operativos monousuario. El salto a conceptos basados en terminales para operar al sistema embebido y el uso de un sistema operativo multiusuario ha representado un obstáculo no despreciable. Sin embargo, la mayoría de estudiantes que interesados en el área de PDS/PDI y sistemas embebidos es consciente de que estas son destrezas necesarias en el futuro empleo y trabajan fuerte para desarrollarlas.

De este modo es un reto proveer un entorno con una curva de aprendizaje no tan fuerte para poder lidiar con todos los detalles de un –para los estudiantes nuevo– desarrollo basado en terminal y adicionalmente tratar con las complejidades de la teoría del procesamiento de señales.

La primer tarea para que una persona pueda trabajar en el desarrollo es instalar el SDK. Por ello, se generó documentación detallada de este proceso de instalación utilizando en particular el SDK de RidgeRun en un computador GNU/Linux.

El tiempo involucrado en este proceso tomó alrededor de cuatro meses, pero expuso las ventajas de la colaboración industria-universidad: el apoyo de RidgeRun en el proceso

fue necesario para tratar con un sinnúmero de variables y problemas encontrados por los asistentes e investigadores en este proceso de instalación; adicionalmente retroalimentación a RidgeRun permitió mejorar su SDK agregando dependencias de software faltantes, y depurando configuraciones por defecto incorrectas, de modo que al final se contara con un proceso más ágil de instalación.

Al mismo tiempo los asistentes detectaron las dificultades comunes que un estudiante de ingeniería afrontará cuando instale el GNU/Linux en un sistema embebido:

1. conceptos de seguridad en Unix (cambio de permisos en archivos y directorios)
2. utilización de Makefiles,
3. búsqueda de información valiosa en las salidas de texto de los procesos de compilación e instalación
4. configuración de un servidor de TFTP,
5. configuración de un servidor NFS,
6. instalación de software en el sistema embebido,
7. acceso al sistema embebido via terminal,
8. el concepto de *bootloader*,
9. eventualmente el uso de terminales ssh para acceder remotamente al computador anfitrión, cuando éste difiere del computador terminal.

La experiencia acumulada se utilizó para preparar un documento cómo-hacer (*How-to*) disponible en [9].

Siguiendo la documentación generada, un usuario con experiencia en GNU/Linux necesita no más de dos días para completar todo el proceso (incluyendo los tiempo de descarga de más de 1 GB de paquetes de instalación de RidgeRun y de Texas Instruments). Un estudiante sin experiencia previa en GNU/Linux requiere de una a dos semanas para lograr configurar la tarjeta Zoom EVM, posterior a la instalación de GNU/Linux en el computador anfitrión.

Una primera aplicación "Hola-Mundo"

La instalación del SDK es un proceso que depende de diversos factores fuera del control de la persona que lo instala. El objetivo del proveedor de un SDK es brindar una manera simple de instalar todo el software necesario para desarrollar nuevas aplicaciones para la plataforma embebida.

La siguiente tarea consiste entonces en producir una aplicación "hola mundo". De nuevo, la carencia de conocimiento sobre sistemas Unix es el mayor obstáculo que se reduce proveyendo en este proyecto con descripciones detalladas de todos los pasos necesarios:

1. cómo crear directorios desde la terminal
2. utilizar un editor de texto para editar código (sin contar con un Entorno Integrado de Desarrollo o IDE)
3. tratar con variables de entorno (por ejemplo, `PATH`)

4. compilación cruzada
5. el concepto de sistema de archivos remoto
6. ejecutar en el sistema embebido un programa utilizando la terminal

La creación de la aplicación "hola-mundo" tomó a los asistentes dos semanas, pero con la documentación producida no tomará más de dos horas a una persona sin experiencia de configurar y ejecutar.

Aplicaciones de audio de baja latencia

Para futuros proyectos de investigación es deseable producir aplicaciones de procesamiento de señales de baja latencia, para las cuales un SoC OMAP-L138 es ideal. Por ello, como paso siguiente se desarrollo un marco de captura, procesamiento y ejecución de audio utilizando ALSA (Advanced Linux Sound Architecture) [5]. ALSA implementa una capa en un nivel bajo de abstracción justo encima del hardware de audio. Así, los programas deben estar parametrizados de forma específica para la plataforma de hardware utilizada. Una mejor solución involucrará portar una capa más elevada de abstracción (por ejemplo Jack [4]) para la Zoom EVM u otras plataformas utilizadas en esta colaboración, puesto que aislaría los detalles del hardware y permitiría concentrarse en los detalles de la aplicación de PDS.

El desarrollo de tres aplicaciones ALSA (para captura, reproducción y dúplex) tomó seis semanas. El código y la documentación se encuentran disponibles en [9], y permiten a una persona dedicarse a la elaboración de los algoritmos PDS.

3.2.3 Tercera fase: eXpressDSP

Esta fase constituyó sin duda alguna la más larga de todo el proyecto, requiriendo alrededor de seis meses comprender todos los pasos involucrados. Aún cuando se produjo como parte de este proyecto una guía detallada de cómo crear un "algoritmo" (módulo de procesamiento) conforme al eXpressDSP (ver [9]), se considera, desde una perspectiva de docencia, que la complejidad del proceso es muy elevada para ser utilizada en cursos introductorios. Para investigación este concepto es problemático por los periodos de permanencia de estudiantes de grado, quienes laboran usualmente tan solo 6 meses en sus proyectos de graduación. Este concepto podría ser utilizado en tesis de maestría, donde se espera la participación por parte de los estudiantes de 12 a 18 meses en un proyecto, lo que sí permitiría la apropiación de las metodologías involucradas.

Una de las mayores dificultades en esta fase estuvo asociada al exceso de información obsoleta sobre versiones previas, incompatibles con el software actual, así como el continuo cambio de versiones de documentos y programas. Asociada a esta dificultad está la ventaja de que los productos de Texas Instruments cuentan con abundante documentación.

Es un reto para la universidad divisar estrategias para poder seguir el rápido progreso comercial en el área de PDS en sistemas embebidos. La complejidad involucrada a los

procesos de implementación se ha elevado a un nivel donde no es posible dedicarse en el aula exclusivamente a los fundamentos teóricos, requiriéndose además un entrenamiento práctico que cambia velozmente con cada producto que ingresa al mercado.

3.2.4 Cuarta fase: C6Run

El nuevo concepto C6Run [1] propuesto por Texas Instruments en Setiembre de 2010 no estaba previsto ni era conocido cuando se planteó la propuesta para este proyecto. Sin embargo, puesto que su objetivo es simplificar todo el proceso de desarrollo, particularmente en las fases iniciales de aprendizaje de desarrollo de aplicaciones PDS/PDI con sistemas empotrados, es completamente pertinente al proyecto y se decidió darle prioridad por sobre los últimos objetivos específicos orientados establecimiento de marcos de verificación y medición de desempeño.

El desarrollo de aplicaciones preliminares con herramientas C6Run tomó alrededor de tres meses, donde las mayores complejidades encontradas radicaron en incompatibilidades de las versiones de software entre el SDK de RidgeRun y las versiones requeridas de las herramientas de Texas Instruments.

3.2.5 Quita fase: Aplicación genérica

Una fase paralela consistió en implementar los tres algoritmos para comparación en un PC de propósito general. Esto permite comparar las estrategias de implementación, e identificar características deseables en cuanto a arquitectura para la futura definición de la interfaz de una biblioteca de DSP multi-plataforma.

La implementación de los filtros en un computador de propósito general utilizando C/C++ es un proceso relativamente directo, que se simplifica por el hecho de no haber limitaciones de recursos de memoria y porque la velocidad de cualquier procesador moderno supera los requisitos para el tratamiento en tiempo real de señales de audio muestreadas a $F_s = 44\,100\,\text{kHz}$, al menos con complejidades similares a los tres filtros descritos en la sección anterior, a costas de un mayor consumo energético.

Como capa de abstracción del hardware de audio se utiliza Jack [4], por sus características de baja latencia y un mayor nivel de abstracción que ALSA, que permite hacer demostraciones de procesamiento de audio en tiempo real.

Una interfaz gráfica realizada con Qt [21] sirve de centro de control y configuración de los filtros. Un hilo de ejecución separado se encarga de leer datos de archivos de audio y pasar los datos de la interfaz gráfica a los procesos de tratamiento digital de señales que maneja el Jack directamente.

Desde una perspectiva didáctica, en el código en la PC se exploran maneras eficientes de realizar búferes circulares, implementación de ecuaciones de diferencias en las Formas Directas I y II, y se tienen libertades para la elaboración de las interfaces gráficas. Para

las actividades prácticas es recomendable proveer un marco de trabajo que se encargue de todo el manejo de archivos de audio, y de la capa de abstracción del hardware para que el estudiante se concentre en las tareas propias de implementación de algoritmos de PDS.

El código fuente para la implementación se encuentra disponible en [9].

4. Resultados y análisis

4.1 Comparación de Estrategias de Implementación

El primer objetivo específico se relaciona con la revisión y documentación de todos los pasos para implementar algoritmos PDS/PDI en un sistema empotrado, siguiendo el marco eXpressDSP. Puesto que Texas Instruments en el transcurso del proyecto propuso un marco de trabajo nuevo (C6Run) y para tener un punto conocido de comparación, se extendió el objetivo a contar con las tres estrategias de implementación: eXpressDSP, C6Run y programación genérica en un PC.

La implementación de los filtros de audio mediante distintas estrategias proporciona una perspectiva de las diferentes ventajas y desventajas que estas presentan. Los principales factores que fueron tomados en cuenta para la comparación son los siguientes:

1. Instalación: complejidad del proceso de descarga y configuración de la herramienta.
2. Implementación: facilidad con que se se logra la programación de un algoritmo utilizando la herramienta.
3. Flexibilidad: nivel de control que se tiene sobre el manejo de recursos de memoria con la herramienta.
4. Construcción: dificultad para realizar el proceso de preprocesado, compilado, ensamblado y enlazado de una aplicación.
5. Portabilidad: grado en que el algoritmo programado puede ser implementado en diferentes plataformas embebidas.

Con estos parámetros se forman conclusiones cualitativas basadas en las experiencias obtenidas en el proceso de implementación de los ejemplos anteriormente descritos mediante las distintas estrategias. La referencia utilizada para valorar los diferentes factores fue el proceso de implementación de los mismos filtros de audio en el computador de propósito general.

En la implementación en la PC, la herramienta utilizada fue el paquete GNU/GCC, el cuál no presenta problemas de instalación por formar parte de las distribuciones de GNU/Linux. C6Run por su parte, presenta un proceso de instalación comparativamente complejo. La descarga del mismo y de sus dependencias se realiza mediante un script. La configuración, sin embargo, depende de qué tan bien se integren los paquetes de TI con el SDK utilizado para el desarrollo de aplicaciones empotradas. Incompatibilidades entre versiones de los distintos paquetes utilizados pueden requerir fuertes inversiones de tiempo para poner al sistema de desarrollo en producción. Por último eXpressDSP también presenta una instalación compleja y trabajosa. En el presente trabajo se utilizó el SDK de RidgeRun por lo que las dependencias y sus rutas son automáticamente instaladas; sin embargo, la configuración del SDK es un trabajo engorroso si no se tiene experiencia previa, lo que es importante considerar en el caso de estudiantes que tiene su primer contacto con DSP, y que se confrontan tanto con la problemática algorítmica como con la implementación.

En cuanto a la implementación de los algoritmos, C6Run permite al usuario reutilizar el mismo código desarrollados para el computador, por lo que la implementación mediante esta herramienta no presenta problemas. El proceso de reemplazar ciertas instrucciones específicas por funciones especiales es realizado automáticamente por herramientas que también son proporcionadas en el paquete. Por otra parte la implementación mediante eXpressDSP resulta un proceso sumamente complejo, en especial si no se tiene experiencia previa. La necesidad de programar el algoritmo utilizando las API proporcionadas, sumada a los archivos de configuración del codec, servidor y de la aplicación, hacen de este proceso una experiencia laboriosa y extensa si se compara con las anteriores estrategias.

La implementación mediante C6Run no pone a disposición del usuario ningún tipo de flexibilidad en cuanto al manejo de recursos del sistema. Internamente la herramienta hace uso de éstos automáticamente por lo que no es posible ningún tipo de optimización y/o especificación de los mismos. Lo anterior presenta una gran desventaja en aplicaciones en las que se tengan restricciones de memoria y hasta de latencias, según se vea afectado el rendimiento del sistema por esta causa. La herramienta eXpressDSP por su parte ofrece al usuario gran flexibilidad en cuanto al manejo de memoria. Mediante los archivos de configuración, transferencias manuales de datos a los distintos niveles de memoria y funciones dedicadas, eXpressDSP ofrece la oportunidad de optimizar el manejo de recursos y tener control de los mismos según se requiera.

C6Run emula las instrucciones del compilador GCC, por lo que el proceso de construcción de una aplicación es bastante cómodo en especial si se tiene experiencia previa en el uso del último. La construcción de una aplicación mediante eXpressDSP requiere más tiempo. Esto es debido a que es un proceso de múltiples pasos utilizando diversas herramientas y opciones, lo que puede generar problemas difíciles de rastrear. Además, el Makefile proporcionado de ejemplo debe ser bastante modificado para que pueda ser utilizado en la aplicación del usuario.

C6Run, está diseñado para ser portado a las distintas arquitecturas de TI C6000 únicamente. La característica anterior se ve opacada por la portabilidad de los codecs creados mediante eXpressDSP, los cuáles por su naturaleza son independientes del marco de trabajo. Esto quiere decir, que con el correspondiente archivo de configuración, un mismo codec puede ser ejecutado en las diferentes arquitecturas de TI. Esto resulta bastante conveniente para el desarrollador que desee reutilizar código, vender un codec genérico, o crear bibliotecas para sistemas empotrados.

4.2 Biblioteca de software

Con RidgeRun no se llegó a ningúna aplicación concreta para definir la biblioteca de software PDS/PDI a desarrollar. Aplicaciones solicitadas por clientes a la empresa permitirá identificar algoritmos que pueden ser implementados como parte de proyectos de investigación en el TEC. En este momento se trabaja con algoritmos de procesamiento de imágenes que pasarán a formar parte de la biblioteca. En la tesis [32] se realizaron dos

algoritmos utilizando eXpressDSP: un reverberador de respueta finita al impulso y otro de cambio de espacios de color. En la actualidad se desarrollan dos tesis de licenciatura que agregarán un algoritmo de generación de anaglifos y un algoritmos de generación de imágenes HDR a la biblioteca.

Los tres algoritmos de audio utilizados en la comparación del objetivo anterior son los primeros integrantes de la biblioteca, denominada dispTEC (digital image and signal processing - TEC). El sitio de dicha biblioteca es [9] en donde se ha colocado tanto el código como la documentación generada.

Un aspecto de peso es que la variedad de entradas, salidas y parametrizaciones de algoritmos, así como las capacidades de configurabilidad estáticas y dinámicas, ya dejan prever que la definición de interfaces no será única en esta biblioteca: si se emplea eXpressDSP (lo que dificultaría su uso por principiantes), ya el estándar establece para cada "codec" cómo denominar a las funciones de proceso, control, inicialización, etc. Si se utiliza C6Run, la interfaz la establece cada desarrollador, pero deberá asemejarse a las definiciones de eXpressDSP, de modo que expertos acostumbrados a eXpressDSP puedan hacer uso de los algoritmos rápidamente. Es evidente que debe encontrarse un equilibrio entre la complejidad de eXpressDSP y un concepto simplificado pero consistente. Por ello, primero se dejará crecer el número de algoritmos disponibles en una especie de proto-biblioteca. Una vez disponibles, puede iniciarse con el diseño de una interfaz unificada.

Otro aspecto importante es el nivel de ejecución: un "codec" de eXpressDSP está ideado para ser llamado desde el GPP pero ser ejecutado en el DSP. Esto dificulta una cascada eficiente de algoritmos en el DSP y obliga al trasiego de datos entre los dos procesadores con los costos en tiempo involucrados por la copia de bloques de memoria. El mismo problema ocurre con algoritmos implementados con C6Run. Por ello, la biblioteca dispTEC tendrá dos niveles: una capa de manejo inter-procesador, y una capa inferior de acceso exclusivo al DSP, al mismo nivel que se encuentran las bibliotecas IMGLIB y DSPLIB de Texas Instruments. De este modo, la implementación de funciones para el DSP pueden hacer uso eficiente de la capa DSP para interconectar módulos tanto de la misma dispTEC como de las bibliotecas de TI, y el usuario que desea utilizar en sus aplicaciones del GPP llama a las funciones de la capa superior (figura 4.1).

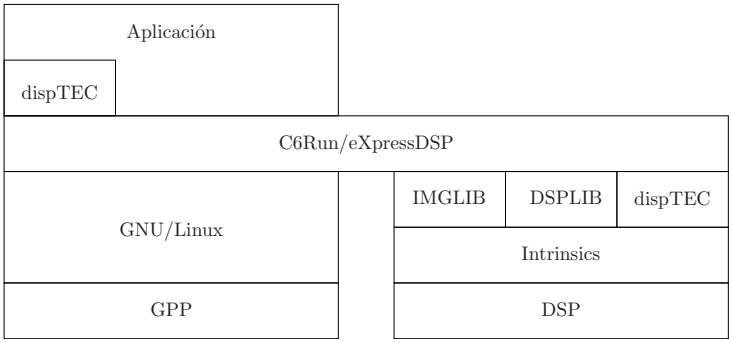

Figura 4.1: Ubicación conceptual de dispTEC en sistemas heterogéneos.

Lo que se ha previsto a partir de los algoritmos implementados hasta ahora es:

1. Una estructura de datos con información de configuración estática
2. Una estructura de datos con información de configuración dinámica
3. Una función para inicialización del módulo que utiliza la configuración estática.
4. Una función de procesamiento de datos
5. Una función de cambio de funcionalidad del módulo de acuerdo a la configuración dinámica. Esta función debe poder ser invocada aun cuando la función de proceso está activa.
6. Una función de cierre de operación

4.3 Marco de verificación

Como marco de verificación se identificó a la biblioteca CUnit [6] para elaborar las unidades de prueba de cada algoritmo. Es una biblioteca establecida que ofrece las funcionalidades básicas para creación de pruebas y los mecanismos para la ejecución de las pruebas de forma automática. La selección de dicha biblioteca es relativamente sencilla: el lenguaje por defecto de implementación de algoritmos para PDS/PDI en sistemas embebidos es C, así que se requiere un marco de pruebas en ese lenguaje. Por otro lado, existe el concepto xUnit como un estándar de facto para la implementación de pruebas, donde x representa lenguajes como Java, C++, C, etc. Se eligió entonces este marco estandar por generalidad.

Puesto que no se ha definido aun la interfaz para la biblioteca dispTEC, no ha sido posible definir la interfaz con que se enlazarán ambas bibliotecas en el marco de verificación. La razón principal para delegar este objetivo a una etapa posterior fue la salida al mercado del nuevo marco C6Run, al que se le dedicó el tiempo por tener mayor impacto en el objetivo general.

4.4 Marco de medición del desempeño

Para medir el desempeño de algoritmos en ejecución en el GPP basta con acceder a los pseudo-archivos creado en `proc`. En el directorio nombrado con el número de proceso aparecen archivos con la información de uso de memoria y de CPU.

En la tesis [32] se exploraron mecanismos de medición de desempeño utilizando módulos de GStreamer. Esto ocurre a alto nivel, y es deseable incluir la funcionalidad de medición de desempeño a más bajo nivel para no depender de marcos de trabajo externos como GStreamer. En dicha tesis se compara la mejora de desempeño cuando los algoritmos se ejecutan en el GPP en comparación a la descarga de algunos de los algoritmos en el DSP. Dependiendo de la configuración, la mejora entre una ejecución de una cadena de procesamiento totalmente en el GPP y descargando módulos al DSP puede ascender hasta en un 81%. Sin embargo, estos datos son relativos a la aplicación concreta, que en este caso

involucró la decodificación AAC de flujos de audio en una tubería de GStreamer, además de la ejecución del algoritmo de reverberación. Estas mediciones utilizan lo mencionado anteriormente.

No se ha encontrado aún cómo acceder directamente desde el lado del DSP a las funcionalidades del DSP/Bios (la capa sustituta de sistema operativo que se ejecuta sobre el DSP) para medición de desempeño. Por el momento la medición ha sido indirecta por medio de la descarga del GPP al delegar funcionalidades al DSP.

5. Conclusiones

El mercado de sistemas embebidos crece continuamente y ha alcanzado a la industrian instalada en Costa Rica. Los sistemas embebidos modernos presentes en aplicaciones multimedia, dispositivos médicos o controles de estabilidad vehicular necesitan procesar enormes cantidades de datos. Para Costa Rica, el mercado especializado en el diseño e implementación de algoritmos PDS/PDI embebidos es atractivo, pero el desarrollo del área es posible solo si las universidades locales preparan a los ingenieros con todas las destrezas y habilidades requeridas, y si éstas cuentan a su vez con el conocimiento para aportar en colaboraciones universidad-empresa con soluciones innovadoras a los problemas presentes en el mercado.

El desarrollo de algoritmos de PDS/PDI tiene dos componentes de igual complejidad: los fundamentos teóricos que soportan las tareas de procesamiento y los aspectos de implementación. Un curso introductorio al DSP debe encontrar un balance entre los dos componentes. Tradicionalmente, herramientas para el prototipado rápido han sido empleadas para desarrollar la experiencia práctica, pero los estudiantes no son confrontados con los aspectos de implementación encontrados en aplicaciones PDS/PDI de tiempo real. Las herramientas de prototipado rápido permiten ilustrar los principios teóricos, y por lo tanto deben considerarse como apoyo para la docencia de este primer componente.

La implementación de algoritmos PDS/PDI crece en complejidad, y en la actualidad hay un amplio espectro de plataformas de implementación que incluye computadores de propósito general, sistemas embebidos, hardware reconfigurable (por ejemplo, FPGA) e incluso el diseño de ASIC. Las últimas son lo suficientemente complejas para requerir cursos completos de postgrado; las primeras son ideales para introducir en el pregrado la implementación de algoritmos PDS/PDI.

La colaboración entre RidgeRun y el TEC brindó en el primer año el "saber-cómo" necesario para incorporar en el curso EL-5805 Procesamiento Digital de Señales el desarrollo de algoritmos PDS/PDI utilizando el SoC OMAP-L138 de Texas Instruments, un dispositivo con tecnología de punta que integra un procesador de propósito general con un DSP.

El SDK de RidgeRun permite agilizar la instalación del entorno de desarrollo, pero tiene aun espacio para mejoras. Conversaciones con los ingenieros de la empresa indican que pronto se brindará el entorno pre-compilado, lo que facilitará enormemente la puesta en marcha de aplicaciones para la plataforma empleada.

5.1 Estrategias de implementación

Las tres estrategias de implementación evaluadas en el marco de este proyecto tienen sus ventajas y desventajas claras desde una perspectiva didáctica. Utilizar un computador de propósito general es accesible a prácticamente todo estudiante de ingeniería y por la

cantidad de recursos disponibles es ideal para dar los primeros pasos en el campo de los algoritmos DSP. Sin embargo, no es apto para el entrenamiento en problemáticas del desarrollo de algoritmos DSP para sistemas empotrados.

Para los primeros pasos en el desarrollo de algoritmos PDS para plataformas empotradas el uso de C6Run es recomendado, pues permite iniciar con estructuraciones de código que separen las tareas que se deben ejecutar en cada procesador. Como este esquema oculta los detalles de los marcos avanzados de desarrollo, permite a un estudiante concentrarse en aspectos de uso de recursos, y los algoritmos como tales.

El objetivo final de todo desarrollador de algoritmos PDS es poder desarrollar en plataformas complejas como eXpressDSP que ofrecen un control completo sobre los recursos, pero con la complejidad asociada. El tamaño de este marco es tal que no se recomienda su manejo directo en un curso introductorio de DSP, pero sí para un segundo curso de técnicas avanzadas de implementación, por los conceptos de ingeniería de software para la reutilización y control de recursos allí incorporados.

En la actualidad se continúa la depuración de las tres plataformas para ofrecer una estructura de código equivalente que permita a estudiantes apreciar las ventajas y desventajas de cada una de ellas.

6. Aportes

En el contexto de este proyecto hubo participación en varias actividades que se describen a continuación.

6.1 Donación

Como parte de este proyecto el TEC cuenta ahora con 10 tarjetas Zoom EVM, módulos JTAG, licencias del Code Composer Studio, y hardware de conexión adicional, por un monto de más de US$ 70 000.

6.2 Conferencia de Prensa

El 18 de Febrero de 2010, iniciando el proyecto, se organizó junto con RidgeRun una conferencia de prensa para informar sobre la donación de equipo y el inicio del proyecto, con la colaboración de CINDE (figura 6.1).

Figura 6.1: Conferencia de prensa y donación de equipo

Se tuvo una cobertura de 11 medios escritos y televisivos, incluyendo a Canal 13, Canal 11, La Nación, El Financiero, La República, Tico Times, entre otros.

6.3 Embedded Technologies Conference

El 19 de Febrero de 2010 se realizó la ETC2010, primera conferencia sobre sistemas empotrados en el TEC, en donde RidgeRun y HP/ProCurve expusieron a estudiantes sus actividades en el área de sistemas embebidos en el país.

El 21 y 22 de marzo de 2011 se realizó en la Ciudad de la Investigación de la UCR la segunda conferencia de sistemas embebidos ETC2011, que esta vez contó con sesión

académica con presentación de artículos, y con sesiones industriales donde expositores de la industria nacional e internacional expusieron sobre los trabajos actuales. Se contó con expositores del calibre de Jason Kridner de Texas Instruments y de Bil Johnson de Hewlett Packard. Esta conferencia contó con la participación de CINDE, RidgeRun, HP, Teradyne, Avionyx, UCR, Cenfotec y el TEC en la organización, y contó con la participación de estas empresas además de Intel y otras del sector de tecnologías móviles. Más información se sintetiza en el sitio de la conferencia [7], que pretende realizarse ahora anualmente.

Figura 6.2: Conferencia de Jason Kridner de Texas Instruments en la ETC2011

La inauguración del evento contó con la prensa nacional así como el Ministro de Ciencia y Tecnología, M.Sc. Alejandro Cruz, y representantes del CINDE, el TEC, la UCR y las empresas participantes.

6.4 Publicaciones

En el contexto de la ETC2011 se publicaron dos artículos [23, 22] en donde se presentan los resultados de este proyecto. Otro artículo [33] presenta los resultados de la tesis [32] desarrollada en el contexto de este proyecto en RidgeRun.

6.5 Cursos

Los resultados de este proyecto se aplicarán directamente al curso electivo EL5805 Procesamiento Digital de Señales, y se evalúa en este momento la integración de algunos conceptos en el curso CE-3102 Análisis Numérico para Ingeniería.

Adicionalmente, ya se cuenta con los conocimientos y la infraestructura para poder incorporar en cursos de la maestría en electrónica, por impartirse a partir de 2012, aspectos avanzados de implementación de algoritmos PDS/PDI.

Bibliografía

[1] Daniel Allred. C6run dsp software development. White Paper SPRY143, Texas Instruments, Dallas, Texas, September 2010.

[2] Eric Biscondi. *C Implementation of the TMS320C62xx Intrinsic Operators, SPRA616*. Texas Instruments, December 1999.

[3] Databeans. Electronics industry market research and knowledge network. digital signal processing (DSP). electronics.ca publications, December 2009.

[4] Paul Davis y Torben Hohn. JACK Audio Connection Kit [online]. March 2011 [visitado el March 6th, 2011]. URL http://jackaudio.org.

[5] ALSA Developers. Advanced Linux Sound Architecture (ALSA) project homepage [online]. 2008 [visitado el April 22nd, 2011]. URL http://www.alsa-project.org.

[6] CUnit Developers. CUnit: A Unit Testing Framework for C [online]. 2001 [visitado el April 22nd, 2011]. URL http://cunit.sf.net.

[7] RidgeRun Engineering. Embedded technology conference [online]. 2011 [visitado el April 22nd, 2011]. URL http://www.embeddedconference.cr.

[8] R.C. González y R.E. Woods. *Digital Image Processing*. Prentice-Hall, 3rd edición, 2008.

[9] Michael Gruner, Bayron Pérez, y Pablo Alvarado. dispTEC [online]. 2010 [visitado el March 5th, 2011]. URL http://disptec.sourceforge.net.

[10] Marius Guran. The architecture and the applications vision of the embedded systems. In *6th Workshop on European Scientific and Industrial Collaboration on promoting Advanced Technologies in Manufacturing WESIC'08*, Bucharest, September 2008.

[11] National Instruments. Ni labview [online]. 2011 [visitado el April 20, 2011]. URL http://www.ni.com/labview.

[12] Texas Instruments. Xdc consumer user's guide, spruex4 [online]. 2007 [visitado el April 20, 2011]. URL http://focus.ti.com/ug/spruex4/spruex4.pdf.

[13] Texas Instruments. Codec Engine [online]. 2010 [visitado el November 22nd, 2010]. URL http://processors.wiki.ti.com/index.php/Category:Codec_Engine.

[14] Texas Instruments. Getting started with IUNIVERSAL [online]. 2010 [visitado el December 1st, 2010]. URL http://processors.wiki.ti.com/index.php/Getting_started_with_IUNIVERSAL.

[15] Texas Instruments. Code composer studio (ccstudio integrated development environment [online]. 2011 [visitado el April 20, 2011]. URL http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html.

[16] Texas Instruments. Ti university program [online]. 2011 [visitado el April 20, 2011]. URL http://focus.ti.com/general/docs/gencontent.tsp?contentId=76595.

[17] Jisun Kim, Tugrul U. Daim, y Timothy R. Anderson. University technology transfer: A conceptual model of impacting factors and phased process. In *PICMET 2009 Proceedings*, págs. 2798–2811, Portland, Oregon, USA, August 2009.

[18] Hermann Kopetz. The ARTEMIS cross-domain architecture for embedded systems. In *Design, Automation & Test, Europe Conference and Exhibition*, volumen 7, págs. 16–20, April 2007.

[19] Gayathri Krishnamurthy. *C Implementation of the TMS320C64x Intrinsic Operators, SPRAA75*. Texas Instruments, December 2004.

[20] MathWorks. Matlab [online]. 1994 [visitado el April 20, 2011]. URL http://www.matlab.com.

[21] Nokia. Qt - Cross-platform applications and UI framework [online]. 2011 [visitado el March 6th, 2011]. URL http://qt.nokia.com.

[22] B. Perez, M. Gruner, y P. Alvarado. Diseño de algoritmos pds para arquitecturas híbridas de texas instruments: una comparación de estrategias de implementación. In *Proceedings of the Embedded Technologies Conference ETC2011*, págs. 31–37, San José, Costa Rica, Marzo 2011.

[23] B. Perez, M. Gruner, y P. Alvarado. dispTEC: industry-university collaboration in Costa Rica on the field of embedded systems and dsp. In *Proceedings of the Embedded Technologies Conference ETC2011*, págs. 1–5, San José, Costa Rica, Marzo 2011.

[24] J. G. Proakis y D. G. Manolakis. *Tratamiento Digital de Señales*. Prentice Hall, 1998.

[25] M. J. Roberts. *Señales y Sistemas. Análisis mediante métodos de transformada y MatLab*. McGraw Hill, 2005.

[26] Richard Stallman. The gnu operating system and the free software movement. In *OpenSources: Voices from the Open Source Revolution*. O'Reilly, January 1999. URL http://hdl.handle.net/2038/1589.

[27] Texas Instruments. *TMS320C64x Image Library, SPRC094*, November 2003.

[28] Texas Instruments. *TMS320C67x DSP Library, SPRC121*, July 2003.

[29] Texas Instruments. *TMS320 DSP Algorithm Standard. Rules and Guidelines. User's Guide. SPRU352G*, February 2007.

[30] Texas Instruments. *OMAP-L138 Low-Power Applications Processor, SPRS586B*, August 2010.

[31] Texas Instruments. *TMS320C67x DSP Library: Programmer's Reference Guide, SPRU657C*, January 2010.

[32] Esteban Zuñiga Mora y Jorge Hidalgo Chaves. Plataforma de software empotrado para la implementación de algoritmos de audio y vídeo en el dsp de la arquitectura omap-l138. Tesis de licenciatura, Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica, Cartago, Costa Rica, Agosto 2010.

[33] E. Zúñiga y J. Hidalgo. Plataforma de software empotrado para la implementación de algoritmos de audio y video en el dsp de la arquitectura omap-l138. In *Proceedings of the Embedded Technologies Conference ETC2011*, págs. 51–57, San José, Costa Rica, Marzo 2011.