Port del Firmware CIAA para plataformas basadas en FPGA con softcore LEON 3

Gerardo L. Puga
Facultad de Ingeniería
Universidad Nacional de La Plata
La Plata, Buenos Aires, Argentina
Email: gerardo.puga@ing.unlp.edu.ar

***Resumen*—El presente trabajo aborda el desarrollo de una capa de portabilidad para el Firmware CIAA que permite ejecutar aplicaciones basadas en este último sobre sistemas de lógica programable FPGA que incluyan el softcore de código abierto LEON.**

Este desarrollo amplía la base de plataformas de hardware del proyecto CIAA, permitiendo ejecutar aplicaciones basadas en la API del firmware oficial del proyecto sobre cualquier kit de desarrollo en FPGA que cuente con recursos suficientes para sintetizar un sistema mínimo basado en el procesador LEON 3.

I. Introducción

El Proyecto de la Computadora Industrial Abierta Argentina (Proyecto CIAA) [1] es un esfuerzo conjunto llevado adelante por un numeroso conjunto de entidades pertenecientes a las esferas tanto públicas como privadas de la República Argentina, el cual tiene por objeto impulsar la modernización de los procesos productivos de la industria nacional al crear una serie de recursos tecnológicos de amplia disponibilidad, alta calidad y bajo costo que puedan ser utilizados como base para el desarrollo de productos competitivos con las ofertas externas.

Uno de los resultados más importantes de este esfuerzo es la Computadora Industrial Abierta Argentina (CIAA), la cual es una plataforma electrónica programable con calidad de grado industrial y diseño abierto. Esta plataforma se encuentra formada por un sistema de cómputo basado en un microprocesador de alto rendimiento y por una serie de interfaces de entrada/salida estándar que pueden interactuar de forma directa con todo tipo de actuadores y sensores industriales, como relés, interruptores, luces, fines de carrera, etc.

Por su finalidad industrial la CIAA fue diseñada utilizando técnicas y componentes de alta calidad que incrementan su confiabilidad cuando se la utiliza en ambientales agresivos para la electrónica, como los que se encuentran frecuentemente en las plantas de producción: ruido eléctrico, polvo, vibraciones, humedad, etc. Esta característica inmediatamente separa la CIAA de otras plataformas de cómputo como las Raspberry Pi, Arduino, etc., que no se encuentran preparadas para funcionar en este tipo de ambientes.

Otro hito del Proyecto CIAA fue la creación de la plataforma llamada Edu-CIAA, o CIAA Educativa. Ésta es una plataforma funcionalmente muy semejante a la CIAA, pero cuyo diseño se encuentra orientado hacia la utilización como herramienta educativa en secundarios técnicos e instituciones terciarias, y como plataforma de entrenamiento de bajo costo en los cursos de programación de la CIAA industrial.

Muy temprano en el proyecto CIAA se tomó la decisión de crear múltiples variantes de cada computadora, basadas cada una de ellas ellas en los microcontroladores de diversos fabricantes (NXP, Freescale, Microchip, entre otros) para eliminar de esta forma el riesgo de generar dependencia dentro del proyecto de los productos de un fabricante particular. Así es que existe una CIAA-NXP, una CIAA-Freescale, y varios tipos de Edu-CIAA.

Para evitar que esta multiplicidad de diseños de hardware fragmentara el conjunto de usuarios y simplificar la portabilidad de las aplicaciones entre las variantes de las computadoras CIAA es que se proyectó la creación de una interfaz de programación común a todas ellas. Esta interfaz se denominó Firmware CIAA [2], y es uno de los pilares del Proyecto CIAA. Este software unifica todas las versiones de hardware bajo una misma interfaz de programación que incluye un sistema operativo de tiempo real (FreeOSEK) y una serie de controladores para dispositivos entrada/salida con interfaces tipo POSIX, entre otros elementos.

El presente trabajo amplía el abanico de plataformas de hardware que es posible programar a través del Firmware CIAA al habilitar su utilización sobre plataformas basadas en sistemas de lógica programable de tipo FPGA. Esta variante presenta una flexibilidad que no comparten otras plataformas CIAA, las cuales tienen rígidamente definidas desde el momento de su diseño todas sus características finales: una plataforma de cómputo basada en lógica programable puede ser reconfigurada para cada proyecto individual, adecuando las características del sistema de procesamiento y en cierta medida de los dispositivos de entrada/salida disponibles a aquello que demanden las circunstancias.

Esta flexibilidad inherente hace que este tipo de sistemas tenga excelentes condiciones como herramienta para el prototipado rápido de sistemas, el desarrollo de sistemas con interfaces especializadas, y la creación de bancos de prueba para otros dispositivos, entre otras aplicaciones.

La amplia disponibilidad de kits de desarrollo para FPGA comerciales que pueden ser utilizados como plataforma de hardware para esta variante del Firmware CIAA le otorga un enorme campo de acción sin ni siquiera requerir la exis-

tencia de una versión de referencia de hardware provista desde el mismo Proyecto CIAA. La riqueza de interfaces de entrada/salida incluidas en los kits FPGA (interruptores, displays, video, audio, conversores A/D y D/A, PS/2, USB, puertos Ethernet, por dar una lista de los más frecuentes), ya probadas y funcionales, permite que en muchos casos todas las necesidades de entrada/salida de un diseño puedan ser satisfechas simplemente seleccionando un kit adecuado.

En este trabajo el procesador del sistema que ejecuta el Firmware CIAA es un System-on-Chip (SoC) sintetizado dentro de la FPGA y basado en el procesador LEON 3. Este último es un procesador RISC de 32 bits basado en la arquitectura SPARC V8 y cuyo código fuente se encuentra disponible en lenguaje de descripción de hardware VHDL bajo una licencia GPL. Es un procesador desarrollado por la firma Cobham Gaisler AB que es principalmente utilizado en aplicaciones de tipo aeroespacial asociado a FPGAs tolerantes a radiación, pero cuya flexibilidad, disponibilidad de herramientas y licencia GPL lo hacen atractivo para cualquier tipo de sistemas de procesamiento en FPGA.

El presente trabajo consiste entonces en la implementación de una capa de compatibilidad que permita la ejecución de aplicaciones basadas en el Firmware CIAA sobre sistemas de procesamiento LEON 3 en circuitos de lógica programable FPGA. El alcance del trabajo fue definido con el criterio de enfocar el desarrollo en aquellos elementos críticos necesarios para poner el sistema en marcha y validarlo: desarrollo de rutinas de bajo nivel del sistema operativo (inicialización, gestión de interrupciones, gestión de contextos de tarea), integración de la arquitectura al sistema de compilación y generación del sistema operativo, adaptación del banco de ensayos funcionales del sistema operativo, y desarrollo de los controladores del temporizador y de la consola serial.

La organización de lo que resta del trabajo es la siguiente. En la Sección II se describen las características de la plataforma, tanto desde la perspectiva del hardware subyacente, como de las interfaces de software del Firmware a las que prestará servicio el código desarrollado. La Sección III describe de forma resumida las características más sobresalientes del diseño realizado. La validación de port del Firmware CIAA se discute en la Sección IV, describiendo primeramente las plataformas de prueba utilizadas y luego la metodología y resultados obtenidos. Por último la Sección V cierra el presente trabajo con un resumen de los logros.

Este artículo se encuentra basado en un trabajo de fin de carrera de la Carrera de Especialización en Sistemas Embebidos de la Universidad de Buenos Aires, el cual puede ser consultado en la referencias bibliográfica [3].

II. Plataforma

II-A. *Firmware CIAA*

El Firmware CIAA es una pieza de software no monolítica, dividida en una serie de módulos que agrupan sub-interfaces del sistema. Además de código propiamente dicho, el firmware está formado también por un sistema de compilación, carga y depuración de las aplicaciones, un banco de ensayos funcionales para validación del sistema operativo y un sistema de generación estática del sistema operativo.

El firmware incluye un sistema operativo de tiempo real (FreeOSEK) que implementa el estándar de la industria automotriz OSEK [4]. Este es un sistema operativo estático, de muy bajo consumo de recursos, y que provee un conjunto de funcionalidades mínimas necesarias para la implementación de un sistema multitarea [5]. Este sistema operativo requiere de una etapa de generación previa a la compilación, durante la cual se lee la configuración de la aplicación y se la utiliza para generar de forma automática una serie de archivos de código de sistema operativo donde se encuentran declarados de forma estática todos los recursos materiales requeridos (espacios de memoria, bloques de control de dispositivos, bloques de control de tareas, etc.).

Para simplificar el acceso a los principales dispositivos de entrada/salida el firmware cuenta con una interfaz de acceso a los mismos con estilo POSIX que permite utilizar las llamadas *open*, *read*, *write*, *ioctl* y *close* sobre una serie de dispositivos virtuales que representan a las interfaces de hardware (puertos digitales, conversores, puertos seriales, etc.). Esto provee una interfaz portable a través de los diferentes modelos de hardware de la CIAA y libera al usuario de la programación de las interfaces más comunes.

El sistema de compilación permite la inclusión o exclusión de módulos en función de las necesidades de cada proyecto individual, pudiendo por lo tanto desarrollar aplicaciones tanto de tipo *bare-metal* como con sistema operativo OSEK con/sin interfaz POSIX.

A los fines del desarrollo del presente trabajo son particularmente relevantes los módulos del firmware denominados *rtos* (que contiene el sistema operativo FreeOSEK), *drivers* (controladores de bajo nivel de acceso a dispositivos), *posix* (interfaz tipo POSIX para acceder a los dispositivos a través de archivos de sistema) y *base* (rutinas de soporte de la arquitectura). También fue necesario realizar extensiones a los mecanismos de configuración de proyectos, y de generación y validación del sistema operativo FreeOSEK.

II-B. *Sistemas basados en LEON 3*

El procesador LEON 3 es un softcore que implementa el estándar IEEE-1754 de definición de la arquitectura SPARC versión 8 (SPARC V8) [6]. Es un procesador de tipo RISC de 32 bits, con unidad de punto flotante opcional, instrucciones de división y multiplicación por hardware opcionales, pipeline de 7 etapas y una eficiencia de aproximadamente 1,4 DMIPS/MHz. El procesador puede ser configurado en tiempo de diseño para modificar sus características (tamaño de caches, políticas de reemplazo, tipo de multiplicadores por hardware, presencia de la FPU, predictor de saltos, entre otros parámetros) dándole al diseñador la posibilidad de encontrar la relación de compromiso más adecuada entre la cantidad de recursos lógicos utilizados y potencia de cómputo del procesador.

El procesador LEON 3 forma parte de una biblioteca de núcleos IP más extensa denominada GRLIB [7], la cual

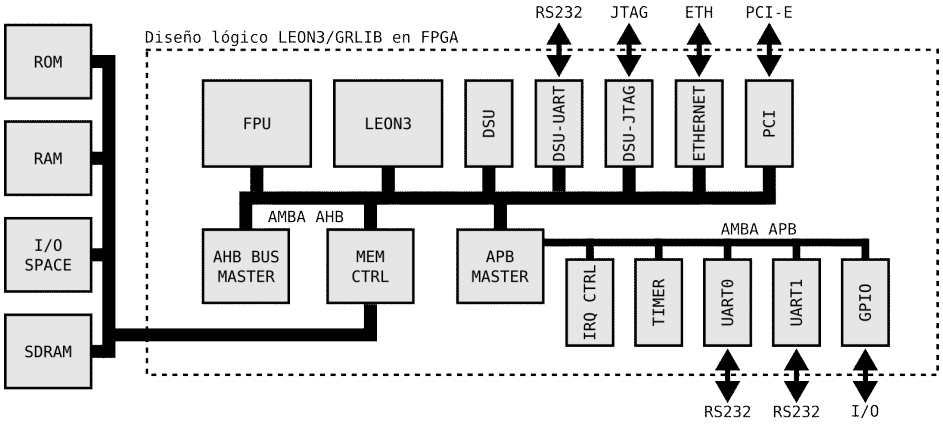

Figura 1. Sistema LEON 3 típico, compuesto por el procesador y por otros núcleos de la biblioteca GRLIB.

Cuadro I
LISTADO DE MÓDULOS ESENCIALES PARA EJECUTAR EL FIRMWARE CIAA SOBRE UN SISTEMA LEON 3.

Módulo	Función
LEON3	Procesador LEON3 que rige el sistema.
MCTRL	Controlador de memoria. Permite mapear una memoria Flash/ROM, una RAM y un rango de direcciones de IO en el mapa de memoria del procesador.
GPTIMER	Módulo temporizador de propósito general, utilizado para activar periódicamente el sistema operativo y permitirle medir el transcurso de intervalos de tiempo. El sistema debe contar con al menos un módulo GPTIMER, que contenga al menos un temporizador independiente.
IRQMP	Controlador de interrupciones del sistema. Utilizado para activar y desactivar interrupciones, así como también para simularlas durante el proceso de validación del sistema operativo OSEK.
DSU3	Unidad de soporte de depuración. Permite cargar y ejecutar programas en memoria, programar la memoria Flash/ROM, examinar el contenido de la RAM, depurar dispositivos, configurar *breakpoints*, etc. Debe estar acompañada de al menos una de las interfaces externas de la DSU: AHBUART, AHBJTAG, etc.

conforma una colección de núcleos IP compatibles entre sí que puede ser utilizada en el diseño de Systems-on-Chip alrededor de un bus central de tipo AMBA. Además del procesador LEON 3, la biblioteca también se encuentra formada por controladores de memoria, controladores de interrupciones, unidades de punto flotante, módulos de comunicación (UARTs, USB, SPI, I2C, Ethernet, etc.), gestores DMA, entre muchos otros módulos. Todos los módulos de la GRLIB se encuentran descriptos en el lenguaje de descripción de hardware VHDL.

Tanto el procesador LEON 3 como la biblioteca GRLIB son sostenidos por la firma Cobham Gaisler AB. La mayor parte de los módulos se encuentran disponibles bajo una licencia de código abierto de tipo GPL v2 que permite la reutilización y modificación del código. Solamente los módulos más avanzados, como por ejemplo las versiones tolerantes a fallas del procesador o los controladores SpaceWire para aplicaciones aeroespaciales, se encuentran cubiertos por licencias comerciales que deben ser adquiridas individualmente a través de Cobham Gaisler AB. La lista completa de los módulos que conforman la GRLIB así como de las licencias que aplican a cada uno puede ser consultada en [8].

En la Fig. 1 puede verse un ejemplo de sistema típico creado utilizando la GRLIB. En esta figura puede verse el bus central de tipo AMBA que coordina los demás integrantes del sistema. El bus tiene dos partes: un bus complejo de alta velocidad AHB donde se conectan los núcleos IP centrales que demandan grandes cantidades de ancho de banda y bajas latencias, y uno o más buses APB de baja velocidad para dispositivos periféricos (UARTs, I2C, etc.). Los elementos más importantes del sistema son el procesador LEON 3, el controlador de interrupciones, el controlador de memoria, el temporizador y uno o más canales de comunicación. El sistema cuenta también con una unidad de depuración (DSU, *Debug Support Unit*); esta última se complementa con una o más interfaces externas que permiten acceder al módulo de depuración a través de un canal de comunicación exterior (USB, UART, Ethernet, etc.).

El flujo de trabajo utilizado con la biblioteca GRLIB consiste en diseñar un sistema interconectando los núcleos IP necesarios, y configurar los parámetros de estos últimos de acuerdo a los requerimientos del sistema (por ejemplo la cantidad de UARTs, tamaño de los FIFOs, presencia o ausencia de unidad de punto flotante, etc.). Una vez definido el sistema, éste es sintetizado, sus puertos de entrada salida mapeados a pines, y finalmente la configuración resultante es programada en la memoria de la FPGA destino.

La DSU forma parte del sistema de depuración del procesador, el cual se completa externamente mediante un programa denominado GRMON. El GRMON es un software que corre en una computadora de escritorio y permite controlar las operaciones de carga y ejecución de programas en el procesador. El canal de comunicación entre la DSU y GRMON puede ser una conexión de tipo serial RS232, USB, JTAG, Ethernet u otra. GRMON puede utilizarse de forma independiente, siendo capaz de cargar programas en el sistema, darles inicio,

establecer *breakpoints*, programar las memorias Flash del sistema, etc., o también puede utilizarse para generar una pasarela entre el programa de depuración GNU GDB y el hardware. Esta última modalidad es la utilizada en el presente proyecto, por ser compatible con el esquema de carga de programas y depuración utilizado por el Firmware CIAA con otras arquitecturas (Cortex-M, por ejemplo).

Con el fin de satisfacer los objetivos del presente trabajo se determinó el conjunto mínimo de módulos de la biblioteca GRLIB que es necesario incorporar al diseño del SoC LEON 3 sintetizado en la FPGA es el que aparece en el Cuadro I, donde se discute la función de cada uno dentro del sistema.

III. Desarrollo realizado

El diseño de la capa de compatibilidad del firmware para el procesador LEON 3 consistió en el desarrollo de las rutinas de bajo nivel del sistema operativo (inicialización, gestión de interrupciones, gestión de contextos de tarea) y de los controladores del temporizador del sistema y de la consola serial, además de la integración de la nueva arquitectura al sistema de compilación del firmware.

En las sub-secciones siguientes se resumen las características más relevantes de los principales elementos desarrollados. Una cobertura más detallada de estos temas puede hallarse en la Memoria de Especialización en la que se basa este trabajo [3].

III-A. Inicialización

A diferencia de lo que sucede con los microcontroladores utilizados en el CIAA Industrial y la Edu-CIAA, los sistemas LEON 3 no tienen una estructura fija sino que sus características son dependientes de los parámetros de configuración seleccionados para cada proyecto particular. Esto significa que la cantidad y tipo de los dispositivos presentes en cada sistema debe considerarse variable, y que incluso la distribución de estos en mapa de memoria debe considerarse inicialmente desconocida durante la inicialización del sistema. Por esta razón para simplificar el desarrollo de software de sistema la GRLIB define un mecanismo denominado de *Plug-and-Play* (PnP) que permite detectar de forma automática la cantidad, tipo y posición en el mapa de memoria de todos los dispositivos que forman parte del sistema.

Utilizando el mecanismo de PnP el sistema operativo determina durante el arranque la posición y características del controlador de interrupciones y el temporizador del sistema, y los configura para permitir la administración de las interrupciones externas y la medición de intervalos de tiempo. La frecuencia del reloj de sistema es también inicialmente incierta, pero puede determinarse automáticamente sabiendo que el *bootloader* del sistema siempre deja configurado el *preescaler* del primer temporizador del sistema para que para que tenga una frecuencia de salida de 1 MHz

Al ser el LEON 3 un procesador de arquitectura SPARC V8 el sistema operativo debe asumir el control de la gestión de las ventanas de registros. Las ventanas de registros son un sofisticado mecanismo de pila utilizado por los procesadores

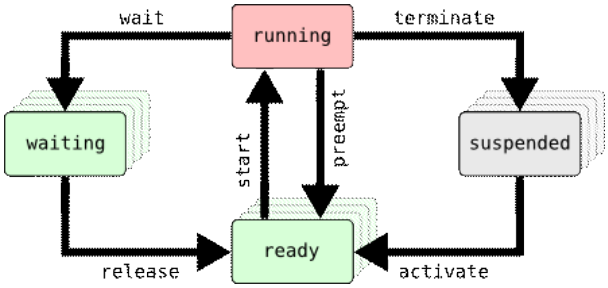

Figura 2. Estados de ejeución de las tareas en OSEK.

de tipo SPARC para reducir la cantidad de accesos a memoria durante los llamados a subrutinas y la atención a rutinas de servicio de interrupción.

Otros eventos relevantes llevados a cabo durante la inicialización son el reemplazo de la tabla de interrupciones del procesador por una que se encuentra administrada por la capa de compatibilidad del firmware, la inicialización de la caches del procesador y la inicialización de los contextos iniciales de todas las tareas de la aplicación de usuario.

III-B. Gestión de Interrupciones en FreeOSEK

El sistema operativo OSEK define dos tipos de interrupciones con habilitación independiente, denominadas categorías ISR1 e ISR2.

Los mecanismos de enmascaramiento provistos por el procesador LEON 3 solamente permiten bloquear interrupciones enmascarando todas las excepciones del sistema (internas y externas) o enmascarando las interrupciones externas según un criterio basado en los niveles de prioridad de ellas (fijos).

Estos mecanismos son suficientes para proteger segmentos de código cuya ejecución deba llevarse a cabo de forma atómica, pero no permiten gestionar las interrupciones de dos categorías ISR1 e ISR2 de forma separada.

Para lograr esta funcionalidad es que se utiliza un nivel de enmascaramiento adicional a través del controlador de interrupciones IRQMP de la GRLIB. Este último es un controlador de interrupciones que permite habilitar y deshabilitar cualquiera de las interrupciones de forma individual. Durante la etapa de generación del sistema operativo se determinan a partir de la configuración de la aplicación dos máscaras de bits que enumeren las interrupciones en cada categoría, y luego estas máscaras se utilizan para habilitar o deshabilitar grupos de interrupciones en el controlador IRQMP.

III-C. Cambios de Contexto en FreeOSEK

La arquitectura de procesadores SPARC V8 no define un mecanismo para la implementación de los cambios de contexto en sistemas multitarea, como si lo hacen por ejemplo los procesadores Cortex-M mediante la interrupción dedicada PendSV. Para implementar esta funcionalidad se debió recurrir a una solución basada exclusivamente en software y que se encuentra integrada con la gestión de interrupciones.

El sistema reconoce dos modos de funcionamiento, Contexto de Usuario y Contexto de Interrupción. El primero es el

modo definido por la ejecución de código de la aplicación de usuario de forma ordenada y secuencial, e incluyendo la ejecución de código de sistema operativo como consecuencia de llamadas a servicios. En este modo de operación la aplicación multitarea se encuentra ejecutando una o múltiples tareas de forma concurrente, pudiendo a lo sumo (pero no necesariamente) encontrarse una de ellas en posesión del recurso procesador (estado de ejecución *running* del sistema operativo OSEK, ver Fig. 2), mientras que las demás se encuentran repartidas entre los estados *ready*, *waiting* y *suspended*.

El Contexto de Interrupción es la ejecución continuada de código de rutinas de servicio de interrupción como consecuencia de una interrupción externa o interrupción por software. Esto puede deberse a la ejecución de una única rutina de servicio de interrupción, o a la ejecución de una colección de ellas de forma anidada. La relación entre contexto de usuario y de interrupción puede verse en la Fig. 3.

Debido al mecanismo de gestión de ventanas de registros de los procesadores SPARC el software de sistema debe gestionar los ingresos y egresos en Contexto de Interrupción (y entre niveles de interrupciones) envolviendo la rutina de interrupción de usuario en un marco que garantice que exista al menos una ventana libre al entrar y por lo menos una ventana ocupada al salir de la rutina de servicio. Para implementar los mecanismos de cambio de contexto en el port del Firmware CIAA se extendió la funcionalidad de este código de forma de tal que almacene el contexto de la tarea que se encuentra en modo *running* en el momento de ingreso en Contexto de Interrupción, y restaure el mismo durante el regreso a Contexto de Usuario. Debe señalarse que la identidad de la tarea en *running* al entrar y al salir puede o no ser la misma en ambos casos, dependiendo de que durante la ejecución del código de interrupción se produzca algún evento que provoque una modificación en este sentido (mediante el envío de eventos a tareas, el vencimiento de alarmas, etc.).

El mecanismo anterior cubre todos los casos de cambio de contexto que son consecuencia directa de la ejecución de una interrupción externa, pero existen otros casos en los que el cambio se debe a la ejecución de una llamada directa al sistema operativo por parte de la aplicación. Ejemplos de esto son la activación y la terminación de una tarea, el envío de eventos entre tareas, etc. Para implementar estos casos las interfaces correspondientes fueron implementadas usando interrupciones por software, quedando de esa forma automáticamente abarcadas por el mecanismo descripto previamente.

III-D. Controlador Serial

Para permitir un mecanismo de entrada/salida flexible durante el proceso de depuración y uso del port LEON 3 del Firmware CIAA se implementó un controlador de UART en el módulo *drivers*. Durante la inicialización del módulo este controlador autodetecta el tipo y la cantidad de UARTs presentes en el sistema usando el mecanismo de auto-descubrimiento de hardware *Plug-and-Play* de la GRLIB.

En función de la cantidad y tipo del hardware presente se registran con el sistema operativo una serie de dispositivos con las correspondientes rutinas de callback necesarias para que la capa POSIX del módulo homónimo pueda acceder a ellos para realizar las operaciones *open*, *read*, *write*, *ioctl* y *close*.

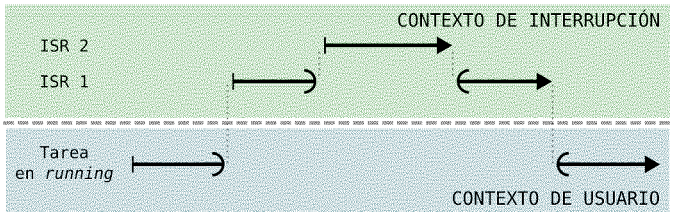

Figura 3. Relación de Contexto de Interrupción con el Contexto de Usuario, representando una transición de modo con interrupciones anidada y regreso al modo original.

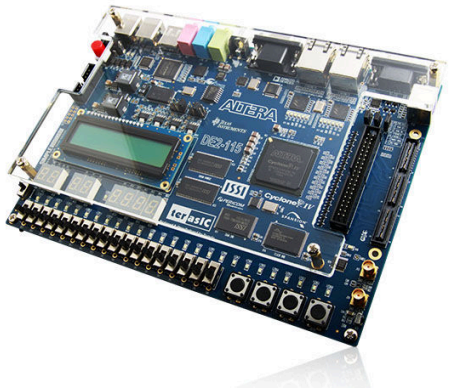

Figura 4. Kit de desarrollo DE2-115.

IV. Resultados

IV-A. Modelos de prueba

El port LEON 3 del Firmware CIAA fue ensayado utilizando dos modelos de hardware diferentes. El primero de ellos es un sistema LEON 3 sintetizado en la FPGA Cyclone IV de un kit de desarrollo Terasic DE2-115. Ver Fig. 4. El sistema mínimo implementado en esta FPGA consiste de un único procesador LEON 3, acompañado de los núcleos IP mínimos necesarios para ejecutar el port del Firmware (listados anteriormente en el Cuadro I), además de núcleos APBUART que implementan UARTs de comunicación externa. La frecuencia del sistema está provista por un reloj de 50 MHz.

El segundo modelo de ensayo se realizó utilizando el simulador TSIM comercializado por Cobham Gaisler AB. TSIM permite simular sistemas completos centrados alrededor de procesadores LEON 3, incluyendo no solamente el procesador sino también todos los módulos auxiliares (buses, caches, controladores de memoria, de interrupciones, UARTs, etc.). Al igual que ocurre con la configuración de un sistema LEON previo a la síntesis, todos los núcleos IP simulados se encuentran parametrizados para modificar sus características funcionales. La simulación es muy precisa, reflejando ciclo a ciclo la evolución del sistema en un hardware físico equivalente al simulado.

Al igual que GRMON, TSIM se presenta como un *target* remoto para GDB que permite la carga y ejecución de programas, lo que lo hace perfectamente intercambiable con un sistema físico accedido mediante el primero.

Utilizando este simulador se generó un sistema LEON 3 virtual en el cual se ejecutó el port del Firmware CIAA. Este sistema simulado tiene la ventaja de poder ser modificado con facilidad mediante parámetros que permiten introducir variantes como el tipo y la cantidad de las UARTs presentes y la frecuencia base del reloj del sistema. Esto fue extremadamente útil durante las fases de puesta en marcha y depuración del código.

IV-B. Validación

La parte más compleja de la comprobación del funcionamiento de la capa de compatibilidad es la validación del funcionamiento del sistema operativo sobre la nueva plataforma. Afortunadamente el Firmware CIAA provee una infraestructura que permite la ejecución semi-automática de una batería de más de doscientos ensayos individuales que evalúan el funcionamiento de todos los mecanismos internos del sistema operativo FreeOSEK (módulo *rtos*). Los casos de prueba incluyen la intervención de interrupciones y múltiples formas de cambio de contexto entre tareas, así como también una cantidad muy grande de variantes en la ejecución de decenas de programas de ensayo, lo que ejercita los mecanismos de gestión de ventanas de registros. Una ejecución exitosa de esta batería de ensayos completa significa por lo tanto un nivel de confianza muy elevado sobre el diseño de los algoritmos de bajo nivel del sistema operativo.

Una vez adaptada esta infraestructura para automatizar la ejecución de la batería de ensayos en la plataforma LEON 3 se ejecutó la misma sobre los dos modelos de prueba disponibles. La flexibilidad del sistema simulado en TSIM para definir el sistema simulado permitió ejecutar la batería múltiples veces haciendo variar en cada caso la frecuencia del sistema. La finalidad de esto último es la de modificar las relaciones temporales entre los eventos simulados y por lo tanto ampliar la cobertura de los ensayos realizados. Un resumen de los resultados de ejecutar la batería de ensayos en cada uno de los modelos de prueba puede encontrarse en el Cuadro II.

La verificación de la base de tiempo del temporizador del sistema operativo (parte del módulo *rtos*) y del controlador UART (módulo *drivers*) debió realizarse manualmente de forma separada. Para eso se diseñaron una serie de aplicaciones de prueba que ejercitaban diversos casos de uso del código a verificar, y se verificó en cada caso que su funcionamiento se encontrase en todo momento dentro de los parámetros de diseño del sistema.

Todos los ensayos realizados fueron satisfactorios comprobando de esta forma el correcto funcionamiento del código desarrollado para la capa de portabilidad así como también el de la estructuras auxiliares (generación de sistema operativo, compilación, carga de programas, ejecución de batería de ensayos, etc.).

Cuadro II
RESULTADOS DE LA VALIDACIÓN DEL MÓDULO *rtos*.

Plataforma	Frecuencia	Resultado
DE2-115	50 MHz	satisfactorio
TSIM	5 MHz	satisfactorio
TSIM	10 MHz	satisfactorio
TSIM	20 MHz	satisfactorio
TSIM	30 MHz	satisfactorio
TSIM	40 MHz	satisfactorio
TSIM	50 MHz	satisfactorio
TSIM	75 MHz	satisfactorio
TSIM	100 MHz	satisfactorio
TSIM	150 MHz	satisfactorio
TSIM	200 MHz	satisfactorio

V. CONCLUSIÓN

En este artículo se presentó el desarrollo de un port que permite ejecutar aplicaciones basadas en el Firmware CIAA en sistemas LEON 3 sintetizados dentro de circuitos de lógica programable de tipo FPGA. Este tipo de sistemas presenta una flexibilidad que la hace ideal como plataforma para prototipado y desarrollo de sistemas de procesamiento especializado.

La ejecución del trabajo consistió en el desarrollo de una capa de compatibilidad que contiene el código de bajo nivel necesario para ejecutar el sistema operativo FreeOSEK sobre el procesador LEON 3, y de un controlador de UART que permite utilizar dicho canal de comunicación a través de las interfaces tipo POSIX del firmware. Se implementaron también los mecanismos de generación del sistema operativo, compilación, depuración, carga de programa y validación automática para incorporar la nueva arquitectura.

El Firmware CIAA portado fue validado utilizando para ello una combinación de ensayos estandarizados y otros diseñados ad-hoc individualmente. Los primeros fueron utilizados para validar el funcionamiento del módulo *rtos*, mientras que los segundos se utilizar para verificar el correcto desempeño del temporizador de sistema y del controlador de la UART en el módulo *drivers*. En todos los casos el comportamiento de port del Firmware CIAA para LEON 3 fue satisfactorio.

REFERENCIAS

[1] Proyecto CIAA, “Computadora Industrial Abierta Argentina,” 2014, disponible: 2016-06-25. [Online]. Available: http://www.proyecto-ciaa.com.ar/
[2] Proyecto CIAA, “CIAA Firmware Project,” 2014, disponible: 2016-06-25. [Online]. Available: https://github.com/ciaa/Firmware
[3] G. L. Puga, *Desarrollo de capa de compatibilidad del Firmware CIAA para procesadores LEON3*, 2016.
[4] *OSEK/VDX - Operating System*, OSEK/VDX Std., Rev. 2.2.3, 2 2005.
[5] M. Cerdeiro. (2015, 11) Introducción a OSEK-OS - El Sistema Operativo del CIAA-Firmware. ACSE.
[6] *IEEE Standard for a 32-Bit Microprocessor Architecture*, Std., 1995.
[7] *GRLIB IP Library User’s Manual*, Cobham Gaisler, January 2016. [Online]. Available: http://www.gaisler.com/index.php/downloads/leongrlib
[8] *GRLIB IP Core User’s Manual*, Cobham Gaisler, January 2016. [Online]. Available: http://www.gaisler.com/index.php/downloads/leongrlib