Prácticas de Laboratorio de Linux Empotrado sobre Placas de Desarrollo XUPV2P

ANTONIO GARCÍA MOYA, ÁNGEL BARRIGA BARROS
Departamento de Electrónica y Electromagnetismo.
IMSE-CNM, CSIC - Universidad de Sevilla. España.
barriga@us.es

Resumen—**En esta comunicación se describe un conjunto de prácticas de desarrollo de sistemas empotrados sobre FPGA que incluye tanto el diseño de la arquitectura hardware del sistema como la configuración, adaptación e implementación de un sistema operativo empotrado.**

Claves: ***Systemas empotrados, Linux empotrado; FPGA***

I. INTRODUCCIÓN

El objetivo de esta comunicación consiste en presentar un conjunto de prácticas orientadas al entrenamiento en los sistemas operativos empotrados. En concreto el trabajo se centra en el sistema operativo Linux para plataformas empotradas sobre FPGA basadas en el procesador MicroBlaze de Xilinx.

El auge de los sistemas empotrados y la complejidad funcional que desde el mercado se impone a estos sistemas requiere disponer de profesionales entrenados en estas materias. Los sistemas empotrados se caracterizan por un fuerte acoplamiento entre el hardware y el software. Ello obliga a que los diseñadores, tanto del sistema empotrado en sí como de aplicaciones, deban aproximarse de manera conjunta tanto a los aspectos hardware como software. En concreto, dentro de los estudios de Informática relacionados con la Ingeniería de Computadores es necesario adquirir las competencias de [1]:

- Desarrollar sistemas empotrados, así como desarrollar y optimizar el software de dichos sistemas.
- Capacidad de analizar, evaluar y seleccionar las plataformas hardware y software más adecuados para el soporte de aplicaciones empotradas y de tiempo real.
- Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.

Las prácticas que se describen en esta comunicación pretenden cubrir parte de estas competencias. Para ello se aborda el desarrollo de la plataforma hardware del sistema empotrado mediante el uso de módulos IP (Intelectual Property). Dicha plataforma hardware será personalizada de acuerdo con las especificaciones que se establezcan para el sistema. A continuación se desarrolla y personaliza el software de acuerdo con los requerimientos impuestos por el hardware subyacente. Esto supone configurar los módulos del sistema operativo, compilar el kernel e implementar y programar el sistema empotrado.

El Proyecto está basado en un curso similar de Xilinx [2] que sirve de base para el entrenamiento de los usuarios de las herramientas y tecnologías de dicho fabricante. La motivación que ha dado lugar a este Proyecto ha sido adaptar dicho curso a la placa de desarrollo XUPV2P (Xilinx University Program Virtex-2 Pro). Esta placa fue desarrollada por Digilent Inc [3] en 2005 y ha sido la plataforma base de prácticas en muchas universidades del mundo.

Actualmente, debido al vertiginoso avance en las tecnologías y arquitecturas de FPGA el dispositivo Virtex-2 Pro está obsoleto y fuera de las líneas de fabricación de Xilinx. Desde la versión 10 de la herramienta ISE para el desarrollo sobre FPGAs de Xilinx dicho dispositivo no está soportado. Dicha versión fue sustituida en 2009 y actualmente nos hallamos en la versión 13.

Ante esta situación nos encontramos con una placa versátil, operativa y que contiene todos los elementos que permiten el entrenamiento de sistemas de alta complejidad y que, sin embargo, no tiene soporte por parte del fabricante. Este hecho ha motivado cubrir este vacío de manera que pueda aprovecharse al máximo la potencialidad de las placas de desarrollo que contamos en los departamentos universitarios.

La comunicación introduce los sistemas empotrados sobre FPGA tanto en los aspectos hardware (arquitecturas, plataforma hardware) como software (sistema operativo Linux empotrado). A continuación se describirá la infraestructura necesaria para realizar las prácticas (placa de desarrollo XUPV2P, procesador MicroBlaze, herramientas de desarrollo EDK y el sistema operativo Petalinux). Finalmente se describirán brevemente las prácticas propuestas.

II. SISTEMAS EMPOTRADOS

A. Arquitectura de Sistemas Empotrados

Un sistema empotrado o embebido (embedded system) puede definirse como un sistema computador de propósito especial integrado en un sistema de ingeniería más general que realiza una o varias funciones específicas, en general, cumpliendo una serie de requisitos funcionales. Básicamente la arquitectura de un sistema empotrado se basa en un elemento

de procesado y elementos de adquisición de datos y comunicación. La figura 1 ilustra dicho esquema.

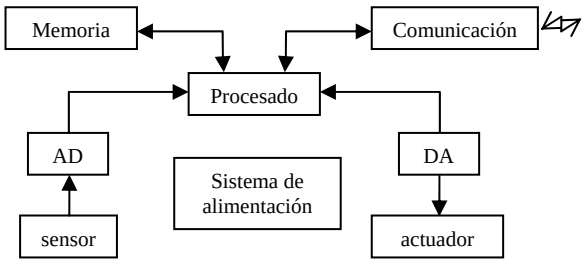

Figura 1.- Arquitectura de un sistema empotrado.

La memoria almacena los programas y datos sobre los que se realiza el procesado. Este bloque suele ser uno de los factores limitantes en un sistema empotrado. Esto da lugar a una limitación en el almacenamiento de los datos y en el tamaño de las aplicaciones software. Otro aspecto que puede dar lugar a limitaciones es la propia gestión de la memoria. En muchos sistemas empotrados los requerimientos de tamaño hacen que el elemento de procesado carezca de unidad de gestión de memoria (MMU, Memory Management Unit). Esto dificulta la gestión de memoria ya que dichos sistemas carecen de mecanismos de protección de memoria y carecen de memoria virtual [4].

El bloque de memoria puede estar constituido por diferentes tipos de memoria y requiere el uso de controladores específicos. Así es posible disponer de memoria interna on-chip y de memoria externa (ROM, DRAM o DDR, SRAM, memoria no volátil Flash, etc) [5].

El bloque de comunicación conecta el sistema con el exterior. Es posible que existan diferentes mecanismos de comunicación en un mismo sistema (Wifi, Bluetooth, GSM, etc). El bloque de comunicación deberá implementar los protocolos necesarios, deberá contener las interfaces, sistemas de modulación, antena, conectores etc. Ejemplos de controladores necesarios son MAC Ethernet, controlador USB1.1/2.0, enlaces de alta velocidad tales como LVDS (Low Voltage Differential Signaling), etc.

La adquisición y generación de datos y señales provienen de sensores y actuadores que interaccionan con el mundo externo. Dicha interacción requiere el acondicionamiento de señales adecuado en función del tipo de sensor/actuador.

El sistema empotrado dispone de un mecanismo de alimentación que suministra la energía necesaria para la operación del sistema. Dependiendo del tipo de sistema empotrado tendremos diferentes elementos de alimentación. En determinados sistemas (por ejemplo basado en batería o bien sistemas batteryless) el consumo de potencia es un factor limitante del sistema empotrado.

Finalmente, el elemento de procesado ejecuta las funciones de control y procesado del sistema empotrado. Normalmente se basa en un microcontrolador, microprocesador o en un DSP.

B. Linux Empotrado

En principio podría pensarse que Linux, al ser un sistema operativo que originariamente se diseñó para funcionar en equipos de sobremesa podría resultar demasiado pesado e inadecuado para sistemas empotrados pero realmente no es así. El núcleo de Linux presenta un alto nivel de granularidad y modularidad que hace que sea fácilmente configurable para trabajar sobre una gran variedad de hardware atendiendo a todo tipo de restricciones (de tamaño, de respuesta en tiempo real, consumo de potencia…). Su sistema de configuración permite elegir sólo aquellos elementos que sean necesarios para nuestro sistema, es decir, para un sistema en el que no se necesiten funciones red basta con deshabilitar estos componentes en la configuración del núcleo y mantener el resto. En cualquier caso existen algunas diferencias esenciales entre el Linux usado en equipos de sobremesa y el usado en sistemas empotrados entre las que cabe destacar, en primer lugar, la forma en que se configura el kernel. El sistema de archivos y los drivers son diferentes. Por ejemplo, en un sistema empotrado puede ser necesario que el sistema de archivos sea de tipo flash (CRAMFS o JFFS2) y por tanto se necesitara un driver de este tipo mientras que los sistemas ordinarios no requieren este tipo de controladores y sistema de archivos.

En segundo lugar en los sistemas empotrados se presta gran atención a las herramientas que se necesitan para el desarrollo, depuración y compilación cruzada mientras que en los sistemas de propósito general (no empotrados) el foco se centra en ofrecer al usuario paquetes que faciliten sus tareas como procesadores de texto, gestores de correo electrónico o herramientas de desarrollo web.

Por último, en tercer lugar, los sistemas de ventanas e interfaces gráficas usados en ambos sistemas son completamente distintos. Por ejemplo, sistema X Windows que se usa en Linux de sobremesa es totalmente inadecuado (debido a sus requerimientos) para entornos empotrados.

III. PLATAFORMA DEL SISTEMA

A. Plataforma de Desarrollo Hardware&Software

El desarrollo de las practicas requiere el empleo de una plataforma que permita combinar por una parte elementos y herramientas de diseño hardware y por otra herramientas de desarrollo y depuración software y de sistema operativo. La figura 2 ilustra estos elementos que constituyen la plataforma de desarrollo del sistema. Dicha plataforma se basa en el entorno de Xilinx EDK [6]. En nuestro caso el objetivo de diseño se enfoca hacia la placa de desarrollo Xilinx Virtex-II Pro (XUPV2P). Esto establece una limitación en la versión de la herramienta EDK que corresponde a la versión 10. Versiones posteriores no soportan el dispositivo Virtex II-Pro.

La herramienta XPS (Xilinx Platform Studio) permite definir y configurar la arquitectura hardware del sistema basada en el procesador MicroBlaze. Por otra parte en base a esta arquitectura se configurará un núcleo de sistema operativo Linux (usando las herramientas SDK y el entorno de desarrollo de Petalinux) ajustado a los requerimientos del sistema y que permitirá, además, incorporar nuevas aplicaciones de usuario.

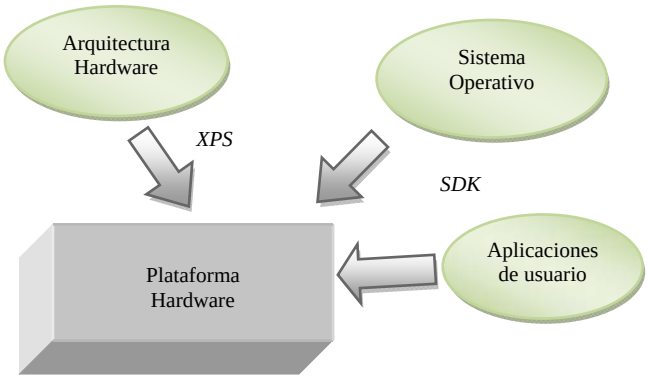

Figura 2.- Esquema de interacción de los elementos.

B. Petalinux

El sistema operativo del sistema es Petalinux, en concreto Petalinux v0.40. Dicho sistema operativo da soporte a las aplicaciones y dispositivos de hardware y proporciona una base sólida y estable al sistema.

El entorno de desarrollo para un sistema empotrado basado en Linux requiere está conformado por una serie componentes tales como herramientas de compilación cruzada (cross-compiler tool chain), Linux kernel, software GNU, depurador o librerías de C. Se necesita agrupar todos estos elementos en un solo framework que, además tiene que ser configurado para el hardware concreto antes de que pueda usarse para crear programas para el dispositivo empotrado. Este proceso se complica aun más cuando estamos ante un dispositivo reconfigurable como un FPGA puesto que se necesita separar el entorno de desarrollo usado para el hardware del proceso de creación de software empotrado[7].

PetaLinux integra todas estas características en un solo entorno de desarrollo que se integra con las herramientas de Xilinx EDK e ISE y que través de la tecnología AutoConfig, simplifica la sincronización entre el hardware y el software.

Petalinux engloba una serie de herramientas (Linux SDK) específicas para el diseño System-on-Chip sobre FPGA's. Sus características principales son las siguientes:

- Software: Código fuente del kernel Linux completo, librerías y utilidades para aplicaciones de usuario, construcción de sistema de archivos raíz del sistema.
- Hardware: Modelos de referencia para FPGA's Xilinx.
- Herramientas: Generador BSP para captar automáticamente nuevas plataformas hardware, herramientas de compilación cruzada (gcc) que incluyen librerías de C estándar, compilador cruzado GDB, generadores de módulos y aplicaciones, estructura de directorios.

Una vez instalado, pueden observarse 3 niveles principales dentro de la jerarquía de directorios de Petalinux: `tools`, `software` y `hardware`. El directorio `tools` contiene la herramientas de compilación de gcc y los scripts propios de petalinux.

El directorio `software` contiene:

- `petalinux-dist`: Es el entorno principal de construcción del sistema desde el que se invoca el script menuconfig para configurar las características de la imagen que vamos a implantar en nuestro sistema.
- `uClinux-2.4.x`: Arbol de ficheros relativo al núcleo de Linux 2.4.
- `linux-2.6.x-petalogix`: árbol de ficheros para el kernel 2.6
- Directorios contenedores de aplicaciones (`user-apps`) y módulos (`user-modules`) de usuario.

El directorio `hardware` agrupa los proyectos de EDK y las herramientas de generación de AutoConfig BSP.

La figura 3 muestra el flujo de diseño/desarrollo dentro del entorno SDK de Petalinux. La selección de la plataforma es el primer paso para la creación de una imagen del kernel personalizada para el diseño. Una configuración de plataforma es, esencialmente, un conjunto de configuraciones del núcleo asociadas a la arquitectura de una plataforma determinada. Este proceso automatizado ahorra al usuario tener que configurar individualmente cada una de las características mencionadas.

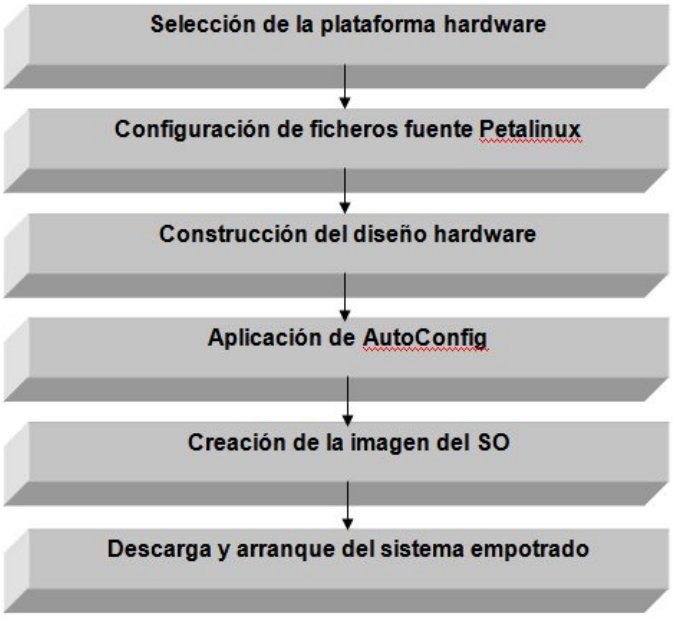

Figura 3.- Flujo SDK de Petalinux

En la fase de configuración de Petalinux se define la configuración completa de la plataforma, que puede dividirse en cuatro áreas: Opciones de proveedor y producto (Vendor/Product Settings), configuraciones y características del kernel (Kernel Settings), opciones configurables por el usuario (Vendor/User Settings) y opciones del sistema (System Settings).

PetaLinux está diseñado para completar el proceso de diseño de Xilinx EDK. Esto permite que los diseños creados desde EDK se puedan integrar fácilmente dentro de Petalinux. Una vez se ha definido la plataforma hardware es preciso generar una serie de parámetros software (del sistema operativo empotrado) basados en la configuración hardware. Petalinux incluye un paquete de soporte de plataformas (BSP) que se utiliza para activar el sistema operativo Linux y para dar soporte al framework de AutoConfig.

El framework AutoConfig de Petalinux permite propagar la configuración del hardware de la plataforma a la configuración del kernel de inux y al bootloader. Para ello, se incluyen una serie de parámetros en el archivo de especificación del microprocesador (system.mss). A continuación se muestra un ejemplo de archivo system.mss:

La creación de la imagen del SO es el proceso más largo (temporalmente) dentro del flujo de diseño. Durante el mismo se crean los archivos que componen el núcleo del sistema operativo y todos aquellos necesarios para el arranque y funcionamiento del sistema. Los mensajes de creación de archivos, de compilación y de configuración van siendo mostrados en la consola mientras dura el proceso.

Una vez generada la imagen del SO, ya solo resta descargarla a la plataforma y arrancar el sistema. Existen diferentes formas de realizar este proceso en función de las características de la plataforma. En el transcurso de las prácticas emplearemos algunas de ellas. En concreto, a través de XMD y a través de la herramienta de Petalinux petalinux-jtag-boot.

C. Entorno de Trabajo

El curso práctico de desarrollo de aplicaciones de Petalinux sobre FPGA Xilinx Virtex II-Pro utiliza la distribución de Linux CentOS 5. El sistema operativo CentOS (Community ENTerprise Operating System) es un clon a nivel binario de la distribución Linux Red Hat Enterprise Linux RHEL, compilado por voluntarios a partir del código fuente liberado por Red Hat. Se trata de un sistema operativo de libre distribución que puede obtenerse desde su sitio web [8].

IV. DESCRIPCIÓN DE LAS PRÁCTICAS

El objetivo de esta comunicación es describir un curso práctico que ilustre y cubra los aspectos fundamentales del proceso de diseño e implementación de un sistema empotrado basado en la arquitectura Microblaze y haciendo uso de Linux empotrado como sistema operativo para dar soporte a las diferentes necesidades del sistema. Para ello se propone la realización de una serie de prácticas que abarcan los aspectos fundamentales de este proceso:

- Definición, diseño y configuración de una plataforma hardware.
- Creación de una imagen del sistema operativo (Linux) a medida para la plataforma.
- Desarrollo y depuración de aplicaciones de usuario.
- Cambios de configuración de kernel para permitir nuevas funcionalidades.
- Integración con periféricos de terceros.

El sistema de prácticas está organizados en 6 sesiones de dos horas de duración cada una. El alumno recibe para cada sesión el boletín que describe la práctica correspondiente. La estructura de dicho boletín esta ornagizada en 5 apartados:

1) Introducción: describe brevemente cada práctica.
2) Objetivos: específicos de la práctica
3) Vista general: corresponde a un esquema que representa el flujo de tareas a realizar en la sesión
4) Desarrollo: describe de manera detallada cada uno de los pasos a realizar en la ejecución de la práctica.
5) Resumen final de la sesión

La figura 4 muestra la organización de las prácticas. A continuación vamos a describir cada una de las prácticas mostradas en la figura 4.

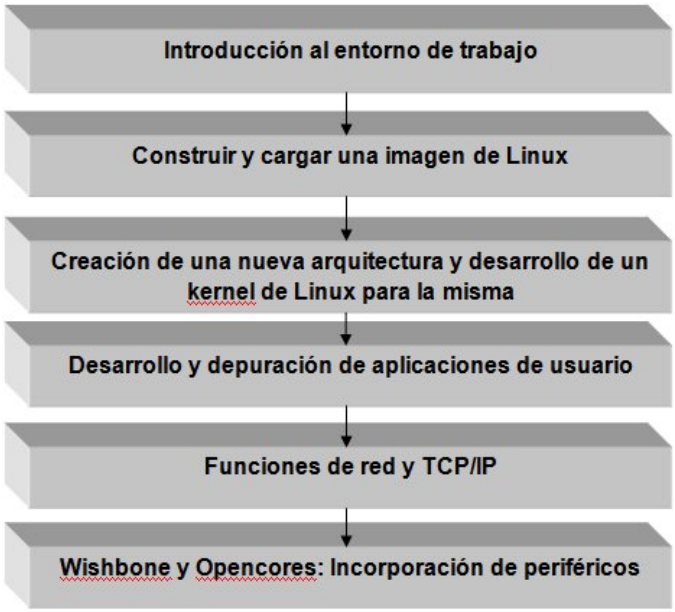

Figura 4.- Estructura de las prácticas.

A. Lab 0: Introducción al entorno de trabajo.

El objetivo de esta práctica es que el alumno se familiarice con el entorno Linux en el que se desarrolla este curso, con la herramienta de desarrollo Xilinx EDK y con la placa de desarrollo en la cual se encuentra el FPGA Virtex-II Pro con el que se trabaja en este curso.

B. Lab 1: Construcción y descarga de una imagen de Linux

En esta práctica se van a utilizar las funciones más básicas necesarias para trabajar con Linux empotrado: Construir y cargar el sistema operativo y las aplicaciones. Las aplicaciones y sistemas operativos para sistemas empotrados basados en Linux, como los desarrollados para arquitecturas basadas en MicroBlaze, se desarrollan en lo que se llama un entorno de

compilación cruzado. Esto significa que dichas aplicaciones y el kernel del SO se compilan en un equipo de desarrollo (en nuestro caso un PC con sistema operativo Linux) y después se descargan en la plataforma hardware.

La distribución estándar de uClinux contiene una serie de herramientas y opciones de configuración que automatizan la mayor parte del proceso descrito anteriormente.

C. *Lab 2: Creación de una nueva arquitectura y desarrollo de un kernel de Linux*

En esta práctica se va a combinar y a profundizar en los conceptos adquiridos en las sesiones anteriores. Por una parte se va a desarrollar una nueva arquitectura hardware basada en MicroBlaze y por otro lado construiremos y configuraremos una plataforma de Linux empotrado a medida para nuestra arquitectura. Finalmente descargaremos ambos elementos a nuestra plataforma y lanzaremos en sistema Linux empotrado.

D. *Lab 3: Desarrollo y depuración de aplicaciones de usuario*

En las prácticas anteriores hemos visto como configurar y construir un sistema Linux empotrado de propósito general. La distribución estándar de Linux empotrado contiene una gran cantidad de aplicaciones y utilidades pero en determinadas situaciones necesitaremos crear nuestros propios programas e incluirlos en la imagen para descargar a la plataforma.

Linux empotrado permite escribir aplicaciones de usuario y posteriormente incluirlas en el sistema de archivos que conforma la imagen de Linux empotrado. En la mayoría de los casos estas aplicaciones se programan en el equipo de desarrollo (y no en el sistema empotrado en sí) por ello es necesaria una compilación cruzada. Petalinux proporciona herramientas para compilar de forma cruzada las aplicaciones empotradas en el equipo de desarrollo. GDB es un depurador de software GNU que funciona en multitud de sistemas Unix y que permite depurar remotamente las aplicaciones incluidas en la plataforma empotrada.

E. *Lab 4: Funciones de red y TCP/IP*

En esta práctica vamos a utilizar las diferentes funciones de red de nuestro sistema Linux empotrado y veremos su utilidad a la hora de desarrollar y testear aplicaciones. Además construiremos una sencilla aplicación web que nos permitirá controlar algunos de los dispositivos físicos de entrada/salida de la plataforma.

F. *Lab 5: Wishbone y Opencores: inclusión de periféricos*

Hasta ahora hemos trabajado siempre con la misma arquitectura hardware en nuestra plataforma pero uno de los aspectos más interesantes y que constituye el núcleo de la versatilidad del diseño sobre FPGA's es la posibilidad de diseñar e incorporar nuevos periféricos a una arquitectura para incrementar la funcionalidad o introducir nuevas características a un diseño determinado.

El bus WishBone es un bus de código abierto estándar diseñado con el objeto de interconectar diferentes elementos dentro de un mismo chip. Este bus es el que utilizan mucho de los diseños disponibles en Opencores [9].

En esta práctica vamos a crear un nuevo proyecto hardware basado en nuestra arquitectura de referencia al que agregaremos 2 módulos IP prediseñados: PLB2WishBone bus bridge (interfaz WishBone) y WishBone simple GPIO (un controlador de entrada/salida) con el que controlaremos los LEDs de nuestra plataforma.

V. CONCLUSIONS

Se ha elaborado un curso dividido en 6 sesiones prácticas que abarca (con dificultad creciente) los procesos básicos de desarrollo de sistemas empotrados sobre FPGA. Para ello se ha desarrollado un protocolo de adaptación del entorno de trabajo para adecuarlo a las características del curso práctico que incluye desde la instalación del sistema operativo a la creación de un modelo predefinido de kernel de Linux para la plataforma de desarrollo (Virtex2-Pro-XUPV2P) pasando por otros procesos necesarios como la instalación de las herramientas Xilinx, instalación de Petalinux y adecuación del entorno "host" de desarrollo mediante scripts de configuración.

AGRADECIMIENTOS

Este trabajo ha sido soportado en parte por la Unión Europea bajo el proyecto FP7-IST-248858, por el Ministerio de Ciencia y Tecnología de España bajo el proyecto TEC2011-24319 y por la Junta de Andalucía bajo el proyecto P08-TIC-03674. Cofinanciación con fondos Feder.

REFERENCIAS

[1] Memoria de Verificación del Título de Graduado en Ingeniería Informática en Ingeniería de Computadores por la Universidad de Sevilla, BOE de 4 de agosto de 2009.

[2] PetaLogix/XUP Professors' Workshop: "Embedded Linux for the Xilinx MicroBlaze Soft Processor", PetaLogix Qld Pty Ltd., 2008.

[3] http://www.digilentinc.com/

[4] P. Radhavan, A. Lad, S. Neelakandan: "Embedded Linux System – Design and Development", Auerbach Pub. 2006.

[5] Y-L. S. Lin, editor: "Essential Issues in SOC Design. Designing Complex Systems-on-Chip" Springer, 2006.

[6] Embedded System Tools Reference Manual. Embedded Development Kit. EDK 10.1 SP3.

[7] http://www.petalogix.com/

[8] http://www.centos.org/

[9] http://www.opencores.org/