Curso Práctico de Sistemas Empotrados Basado en Placas de Desarrollo XUPV2P

Antonio García Moya y Angel Barriga Barros, *Member, IEEE*

***Title*—Practical Course of Embedded Systems based on XUPV2P Development Boards**

***Abstract*—This paper describes a lab course about embedded systems on FPGA. The proposed practices cover the main features of the design process, which includes the hardware architecture design, and the embedded operating system configuration, adaptation and implementation.**

***Index Terms*—Embedded systems, embedded Linux, FPGA, laboratories**

I. Introducción

El objetivo de este artículo 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 adiestrados 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]:

- Desarrollo de sistemas empotrados, así como diseño y optimización del software para dichos sistemas.
- Capacidad de análisis y evaluación para seleccionar las plataformas hardware y software más adecuadas 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 este trabajo 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 se personalizará de acuerdo con las especificaciones que se establezcan para el sistema. A continuación se desarrolla y personaliza el software de acuerdo con los requisitos impuestos por el hardware subyacente. Esto supone configurar los módulos del sistema operativo, compilar el *kernel* e implementar y programar el sistema empotrado.

El curso está basado en uno similar de Xilinx [2]. La motivación que ha dado lugar a este trabajo ha sido adaptar dicho curso a la placa de desarrollo XUPV2P (*Xilinx University Program Virtex-2 Pro*). Esta placa (Figura 1) 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 FPGAs, 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 se encuentra disponible la versión 14.

Ante esta situación la Virtex-2 Pro constituye una plataforma versátil, operativa y que contiene todos los elementos que permiten el entrenamiento con 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. [4].

Este trabajo introduce los sistemas empotrados sobre FPGA tanto en los aspectos hardware (arquitecturas, plataforma hardware) como software (sistema operativo Linux empotrado). En la sección III se describirá la infraestructura necesaria para realizar las prácticas (placa de desarrollo XUPV2P, procesador MicroBlaze, herramientas de desarrollo EDK y el sistema operativo Petalinux). En la sección IV se considerará el contexto docente en el que se imparte el curso. Finalmente, en la sección V, 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,

Antonio García Moya es estudiante de la Universidad de Sevilla, España. (e-mail: antgarmoy@gmail.com).Ángel Barriga Barros pertenece al Departamento de Electrónica y Electromagnetismo de la Universidad de Sevilla, España (e-mail: barriga@us.es).

DOI (Digital Object Identifier) Pendiente

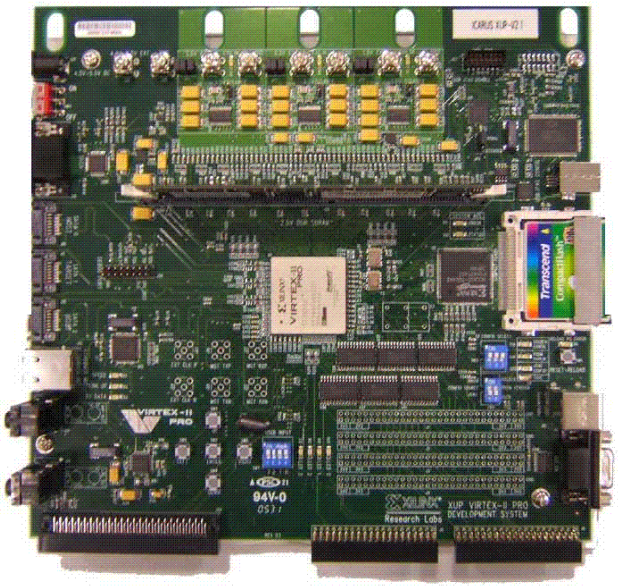

Figura 1.- Placa de desarrollo XUPV2P.

cumpliendo una serie de requisitos funcionales [5]. Muchos de estos sistemas están enfocados a realizar una única tarea o un conjunto muy limitado de tareas a las que en algunos casos se exige restricciones de funcionamiento en tiempo real, coste, tamaño, consumo, etc. Algunas características que suelen presentar estos sistemas son:

- Concurrencia. Los componentes del sistema deberán operar en paralelo.
- Fiabilidad y seguridad. El sistema debe ser fiable y seguro frente a errores. El manejo de estos errores puede realizarse vía hardware o software.
- Interacción con dispositivos físicos. Los sistemas empotrados interaccionan con el entorno a través de dispositivos entrada/salida no usuales, por lo que suele ser necesario un acondicionamiento de las señales.
- Robustez. Al sistema empotrado se le impondrá la necesidad de la máxima robustez ya que las condiciones de uso no tienen porqué ser "buenas". Por ejemplo el sistema puede estar en el interior de un vehículo con diferentes condiciones de operación.
- Consumo reducido. En los sistemas basados en baterías la reducción del consumo implica una mayor autonomía de operación.
- Dimensiones pequeñas. Las dimensiones de un sistema empotrado no dependen sólo de sí mismo sino también del espacio disponible para su ubicación.

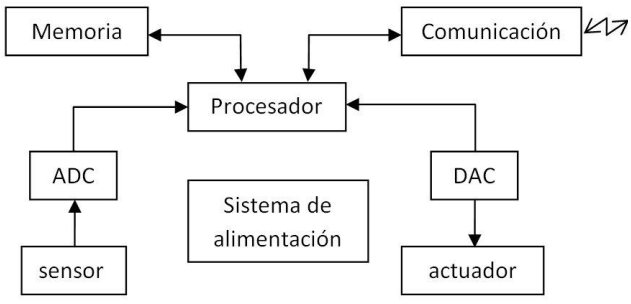

Figura 2.- Arquitectura de un sistema empotrado.

Básicamente la arquitectura de un sistema empotrado se basa en un elemento de procesado y elementos de adquisición de datos y comunicación. La Figura 2 ilustra dicho esquema.

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 las restricciones 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 [5].

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) [6].

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 USB1.1/2.0/3.0, 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 existen 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 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 el sistema particular, por ejemplo para un sistema en el que no se necesiten funciones de 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 necesita 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 el sistema *X-Windows* que se usa en Linux de sobremesa es totalmente inadecuado (debido a sus requisitos) para entornos empotrados.

III. Plataforma del Sistema

A. *Plataforma de Desarrollo Hardware&Software*

El desarrollo de las prácticas 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 3 ilustra estos elementos que constituyen la plataforma de desarrollo del sistema. Dicha plataforma se basa en el entorno de Xilinx EDK [7]. En este 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 interfaz de usuario XPS (*Xilinx Platform Studio*) del entorno Xilinx EDK 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 las características del sistema y que permitirá, además, incorporar nuevas aplicaciones de usuario.

B. *Petalinux*

El sistema operativo es Petalinux, en concreto la versión 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 estar conformado por una serie componentes tales como herramientas de compilación cruzada (*cross-compiler tool chain*), el *kernel* de Linux, software de GNU, depurador o bibliotecas de C. Se necesita agrupar todos estos elementos en un solo entorno de trabajo o *framework* que, además, tiene que configurarse para el

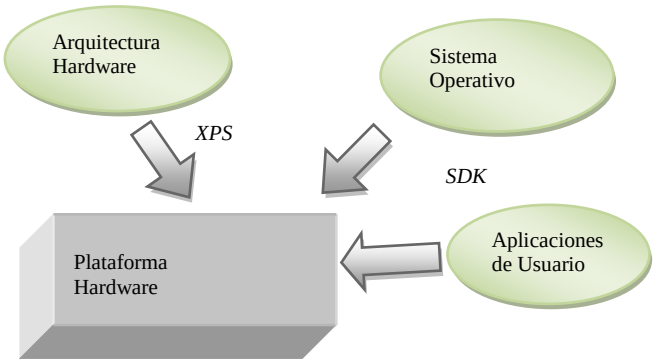

Figura 3.- Esquema de interacción de los elementos.

hardware concreto antes de que pueda usarse para crear programas para el dispositivo empotrado. Este proceso se complica aún más cuando se realiza sobre un dispositivo reconfigurable como una FPGA puesto que se necesita separar el entorno de desarrollo usado para el hardware del proceso de creación de software empotrado [8].

Petalinux integra todas estas características en un solo entorno de desarrollo que se integra con las herramientas de Xilinx EDK e ISE mediante la aplicación AutoConfig. Dicha utilidad 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 FPGAs. Sus características principales son las siguientes:

- Software: código fuente del *kernel* de Linux completo, bibliotecas y utilidades para aplicaciones de usuario y construcción del sistema de archivos raíz.
- Hardware: modelos de referencia para FPGAs de Xilinx.
- Herramientas: generador BSP para captar automáticamente nuevas plataformas hardware, herramientas de compilación cruzada (gcc) que incluyen bibliotecas de C estándar, herramienta de depuración GDB, generadores de módulos, aplicaciones y estructura de directorios.

Una vez instalado, pueden observarse tres niveles principales dentro de la jerarquía de directorios de Petalinux: `tools`, `software` y `hardware`. El directorio `tools` contiene las 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 se va a implantar en nuestro sistema.
- `uClinux-2.4.x`: árbol 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 de usuario (`user-modules`).

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

La Figura 4 muestra el flujo de diseño dentro del entorno SDK de Petalinux. La selección de la plataforma es el primer paso para la creación de una imagen del *kernel*

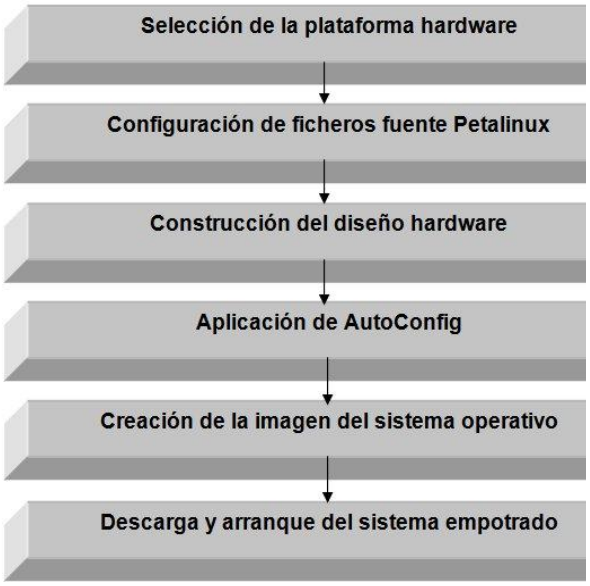

Figura 4.- Flujo SDK de Petalinux

personalizada para el diseño. Una configuración de la 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.

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*). La Figura 5 muestra la apariencia de la aplicación AutoConfig.

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 entorno de trabajo de AutoConfig.

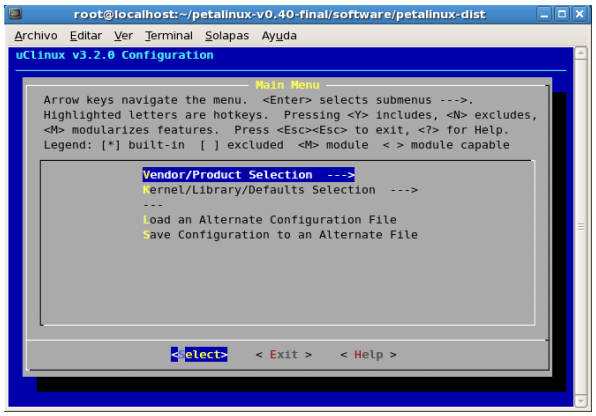

Figura 5.- Menú principal de la aplicación AutoConfig

La aplicación AutoConfig de Petalinux permite propagar la configuración del hardware de la plataforma a la configuración del *kernel* de Linux y al *bootloader*. Para ello se incluyen una serie de parámetros en el archivo de especificación del microprocesador (`system.mss`). La Figura 6 muestra un ejemplo de archivo `system.mss`.

La creación de la imagen del sistema operativo es el proceso que más tiempo consume dentro del flujo de diseño. Durante este proceso 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 sistema operativo el siguiente paso consiste en descargarla en la plataforma hardware 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 se emplean algunas de ellas. En concreto, a través de la consola XMD (que se conecta con el procesador mediante la interfaz JTAG) y a través de la herramienta de Petalinux denominada petalinux-jtag-boot.

C. *Entorno de Trabajo*

El curso práctico de desarrollo de aplicaciones de Petalinux sobre FPGA de 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 [9].

El entorno de trabajo se basa, por lo tanto, en ordenadores personales bajo el sistema operativo Linux CentOS 5. Al configurar dicho entorno se requieren las herramientas de compilación de GNU C/C++ gcc. En dicho entorno se dispone de las instalaciones de la versión Xilinx EDK 10.1 y la versión de Petalinux 0.40. También se requiere de una herramienta que permita disponer de un hiperterminal que sirva de consola del sistema. En nuestro caso se ha optado por emplear Kermit.

IV. Contexto Docente

```
BEGIN OS
 PARAMETER OS_NAME = petalinux
 PARAMETER OS_VER = 1.00.b
 PARAMETER PROC_INSTANCE = microblaze_0
 PARAMETER stdout = RS232_Uart
 PARAMETER stdin = RS232_Uart
 PARAMETER main_memory = plb_ddr_0
 PARAMETER main_memory_bank = 0
 PARAMETER main_memory_size = 0x04000000
 PARAMETER flash_memory = opb_emc_0
 PARAMETER flash_memory_bank = 1
 PARAMETER lmb_memory = lmb_bram_if_cntlr_1
```

Figura 6.- Fichero `system.mss`

El objetivo de este trabajo 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, haciendo uso de Linux empotrado como sistema operativo para dar soporte a las diferentes necesidades del sistema. Como paso previo a la descripción del curso práctico es necesario establecer los condicionantes docentes así como la metodología.

El curso se enmarca en titulaciones de grado en el ámbito de las TIC (Tecnologías de la Información y las Comunicaciones). Como requisito previo es necesario que los estudiantes hayan adquirido un conjunto de conocimientos y competencias previas:

- Programación en lenguajes C o C++
- Conocimientos básicos de sistemas operativos
- Conocimientos sobre arquitectura de ordenadores
- Diseño de sistemas empotrados *standalone*

Los dos primeros prerrequisitos son necesarios ya que el curso se enfoca en el diseño e implementación de sistemas operativos sobre sistemas empotrados y el desarrollo de aplicaciones empotradas. En concreto el sistema operativo es Linux y las aplicaciones se desarrollan en los lenguajes de programación C o C++.

Por otro lado los sistemas empotrados se basan en el procesador MicroBlaze de Xilinx. Se trata procesador RISC (*Reduced Instruction Set Computer*) que está optimizado para implementaciones sobre FPGAs de Xilinx. Tiene una arquitectura Harvard (buses separados de instrucciones y datos) de 32 bits. Las arquitecturas hardware del sistema requieren el uso de módulos IP tales como puertos serie y paralelo, controladores de memoria, controladores de interrupciones, *timers*, etc.

Al tratarse de un curso avanzado sobre diseño de sistemas empotrados es necesario que los estudiantes hayan adquirido cierta destreza en el desarrollo de dichos sistemas. En concreto en el uso de las herramientas de Xilinx ISE y EDK para el diseño de sistemas *standalone* basados en MicroBlaze. A tal efecto se ha incluido dentro del curso una práctica de introducción a dichas herramientas. En estos sistemas el control es realizado por una aplicación software centrada en la realización de una única tarea (o un conjunto reducido de tareas). Sin embargo para ciertas aplicaciones surge la necesidad de disponer de un sistema multitarea y, por lo tanto, de un sistema operativo que gestione los recursos hardware, planifique la ejecución de procesos y proporcione servicios a las aplicaciones.

El sistema de prácticas está organizado 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 está organizada en 5 apartados:

1) Introducción: describe brevemente la práctica que se va realizar.
2) Objetivos: en este apartado se detallan los objetivos de aprendizaje.
3) Vista general: es un diagrama de flujo de las actividades a realizar en la sesión (Figura 7).

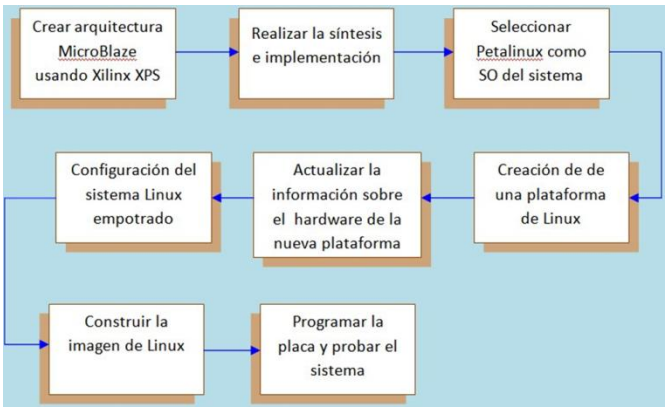

Figura 7.- Ejemplo de la vista general de una de las prácticas

4) Desarrollo: describe las instrucciones paso a paso para la realización de la práctica.
5) Resumen: sumario detallado de los conocimientos adquiridos.

V. Descripción de las Prácticas

El curso se basa en la realización de una serie de prácticas que abarcan los aspectos fundamentales del proceso de diseño e implementación de un sistema empotrado. Los aspectos que se han considerado son los siguientes:

- 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 del *kernel* para permitir nuevas funcionalidades.
- Integración con periféricos de terceros.

La Figura 8 muestra la organización de las prácticas. A continuación se describen brevemente cada una de dichas prácticas.

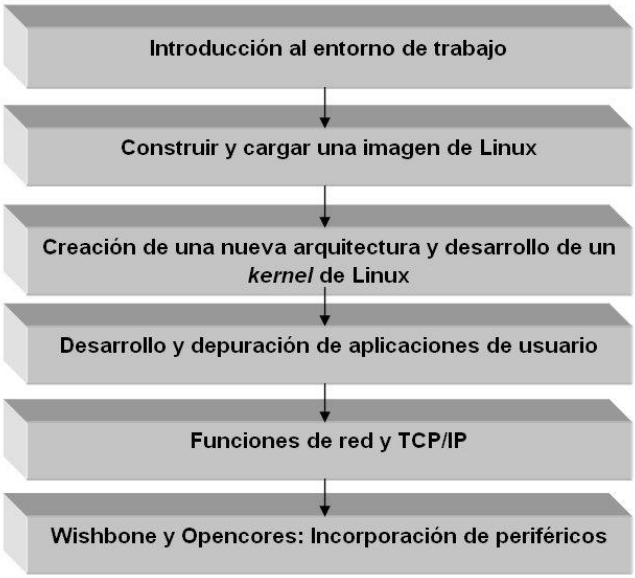

Figura 8.- Estructura de las prácticas.

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 la FPGA Virtex-II Pro con el que se trabaja en este curso.

Tras la realización de esta práctica el alumno debe haberse familiarizado con la utilización del asistente BSB (*Base System Builder wizard*) para diseñar una arquitectura basada en el microprocesador MicroBlaze, así como la revisión y modificación de un proyecto. También habrá generado un archivo de *bitstream* para programar la FPGA incluyendo la aplicación software correspondiente. Una vez programada la placa de desarrollo se realizará la conexión tipo hiperterminal usando Kermit para interactuar con el circuito y probar su funcionalidad.

B. *Lab 1: Construcción y Descarga de una Imagen de Linux*

En esta práctica los estudiantes van a utilizar las funciones 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 sistema operativo se compilan en un equipo de desarrollo (un PC con sistema operativo Linux denominado *host*) diferente del equipo en que se ejecuta (denominado objetivo o *target*).

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

Por lo tanto tras la realización de esta práctica el alumno debe haber adquirido los conocimientos necesarios sobre el uso y manejo de las herramientas de uClinux (Petalinux) para crear el *kernel* y las aplicaciones de usuario básicas de un sistema Linux. Ello le permitirá crear una imagen de Linux y descargar esa imagen en la plataforma hardware. Finalmente se comprobará el funcionamiento del sistema.

C. *Lab 2: Creación de una Nueva Arquitectura y Desarrollo de un Kernel de Linux*

En esta práctica se van a combinar y profundizar 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 se construirá y configurará una plataforma de Linux empotrado a medida para la nueva arquitectura hardware.

Se busca fijar conceptos recurrentes en el diseño de los sistemas empotrados. Para ello se repasan y practican tareas de creación y modificación de la arquitectura hardware con el uso de la herramienta BSB, se crea una nueva plataforma de Linux empotrado utilizando las herramientas proporcionadas por Petalinux y se realiza la programación y arranque del nuevo sistema Linux empotrado basado en MicroBlaze.

D. *Lab 3: Desarrollo y Depuración de Aplicaciones de Usuario*

En las prácticas anteriores se ha mostrado 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, sin embargo en determinadas situaciones es necesario crear nuevos programas específicos e incluirlos en la imagen para descargar en la plataforma hardware.

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 y depurar 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.

Como resultado de aprendizaje de esta práctica el alumno creará una aplicación de usuario sencilla (se trata del típico programa "Hola mundo") usando las herramientas de Petalinux. De esta manera se recorre el procedimiento de compilación y de agregación de dicho programa en la imagen del sistema. También se practica la depuración de aplicaciones mediante la utilidad GDB.

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

En esta práctica se utilizan diferentes funciones de red del sistema Linux empotrado. Con ello se muestra al alumno su utilidad a la hora de desarrollar y testear aplicaciones. Además se construirá una sencilla aplicación web que permitirá controlar algunos de los dispositivos físicos de entrada/salida de la placa de desarrollo.

Entre las actividades de utilización de las funciones de red en la práctica se propone acceder al sistema empotrado mediante Telnet y se realiza transferencia de ficheros usando FTP (*File Transfer Protocol*). También se plantea montar el sistema de archivos desde el equipo de desarrollo al sistema empotrado usando NFS (*Network File System*) así como la utilidad para ejecutar y probar aplicaciones Linux directamente sobre NFS sin necesidad actualizar y re-arrancar el sistema Linux de Microblaze con una nueva imagen.

La última parte de la práctica consiste en probar un servidor web empotrado. Para ello se construye una sencilla página HTML estática que permite encender y apagar diodos LED de la placa de desarrollo (Figura 9). De esta manera se muestra un mecanismo de control del hardware de manera remota.

Figura 9.- Aplicación web de control de la placa de desarrollo.

F. Lab 5: Wishbone y Opencores: Inclusión de Periféricos

Hasta ahora se ha trabajado siempre con la misma arquitectura hardware en la plataforma. Sin embargo, uno de los aspectos más interesantes y que constituye el núcleo de la versatilidad del diseño sobre FPGAs es la posibilidad de diseñar e incorporar nuevos periféricos a una arquitectura para incrementar la funcionalidad o introducir nuevas características a un diseño determinado. Por ello uno de los objetivos de la práctica es comprender el uso de módulos IP para ampliar la funcionalidad de una plataforma hardware

En esta práctica los estudiantes crearán un nuevo proyecto hardware basado en la arquitectura de referencia empleada en las prácticas anteriores. En este nuevo proyecto se agregarán dos módulos IP prediseñados: PLB2WishBone_bus_bridge (interfaz con el bus WishBone) y WishBone_simple_GPIO (un controlador de entrada/salida) con el que se controlarán los diodos LEDs de la placa de desarrollo tal y como muestra la Figura 10.

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 muchos de los diseños disponibles en OpenCores [10].

VI. CONCLUSIONES

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.

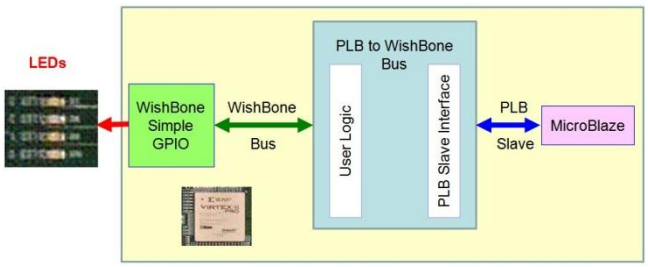

Figura 10.- Inclusión de nuevos periféricos y conexión al bus PLB.

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/, Julio de 2012.
[4] A. García Moya, A. Barriga Barros: "Prácticas de Laboratorio de Linux Empotrado sobre Placas de Desarrollo XUPV2P", X Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'2012), Vigo, Junio 2012.
[5] P. Radhavan, A. Lad, S. Neelakandan: "Embedded Linux System – Design and Development", Auerbach Pub. 2006.
[6] Y-L. S. Lin, editor: "Essential Issues in SOC Design. Designing Complex Systems-on-Chip" Springer, 2006.
[7] Embedded System Tools Reference Manual. Embedded Development Kit. EDK 10.1 SP3, Xilinx, 2008
[8] http://www.petalogix.com/, Julio de 2012.
[9] http://www.centos.org/, Julio de 2012.
[10] http://www.opencores.org/, Julio de 2012.

Antonio García Moya es estudiante de último curso de Ingeniería Informática en la Universidad de Sevilla. Realizó su Proyecto de Fin de Carrera en 2011 en el Departamento de Electrónica y Electromagnetismo de la Universidad de Sevilla. Actualmente desarrolla su labor profesional en el ámbito de la empresa privada. Su interés investigador se enfoca hacia los sistemas empotrados y el diseño de arquitecturas y aplicaciones sobre FPGAs.

Angel Barriga Barros (M'09) es catedrático de la Universidad de Sevilla. Obtuvo el título de Licenciado en Física por la Universidad de Sevilla (España) en 1984 y los grados M.S. y PhD por dicha Universidad en 1986 y 1989 respectivamente. Desde 1984 está adscrito al Departamento de Electrónica y Electromagnetismo de la Universidad de Sevilla. Actualmente también está adscrito al Instituto de Microelectrónica de Sevilla (Centro Nacional de Microelectrónica del Consejo Superior de Investigaciones Científicas) en el que es responsable del grupo de investigación "Diseño Digital y de Señal Mixta". Miembro de IEEE Computational Intelligence Society (CIS) y European Society Fuzzy Logic and Technology (EUSFLAT). Su interés investigador se centra en áreas relacionadas con las metodologías de diseño de circuitos integrados VLSI y, en particular, el diseño de circuitos digitales CMOS, implementación digital de sistemas neuro-fuzzy y herramientas de CAD para el desarrollo de sistemas basados en lógica fuzzy. Es autor de aproximadamente unas doscientas publicaciones en libros, revistas y congresos. Ha participado en numerosos proyectos de investigación (nacionales y europeos) así como en contratos de desarrollo industrial.