Revista Cubana de Ciencias Informáticas
Vol. 7, No. 1, Enero-Marzo, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 44-54
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Software libre
Recibido: 15/01/2013 | Aceptado: 1/03/2013

Linux embebido en FPGA para sistemas de monitoreo industrial

Ebedded linux on FPGA for industrial monitoring systems

Pedro E. Calleja Acosta, Miguel E. Iglesias Martínez, José F. Carmona Martínez

CDEA. Centro de Desarrollo de la Electrónica y la Automática, km 2 ½, Carretera al Aeropuerto Álvaro Barba, Pinar del Río, Cuba

* Autor para la correspondencia: pecalleja@cdea.co.cu

Resumen

Actualmente la obtención en *Field Programmable Gate Array* de módulos independientes para aplicaciones específicas de monitoreo industrial, ha hecho que su reutilización e integración sea compleja y que los tiempos de desarrollo y puesta a punto de los mismos sean considerablemente elevados. En este trabajo se propone una solución a este problema basada en embeber Linux en un *Field Programmable Gate Array*, específicamente utilizando el *kit* Spartan3AN. Las herramientas de codiseño hardware/software utilizadas para lograr este objetivo han permitido tener resultados a corto plazo que indican el enfoque para futuras investigaciones, de manera que se logró ejecutar aplicaciones de uso general para el control de ip-cores empotrados en hardware reconfigurable. La plataforma obtenida es estable y flexible a futuras implementaciones tanto de software como de hardware.

Palabras clave: FPGA, Linux, MicroBlaze, Spartan3AN, sistemas embebidos.

Abstract

Currently obtaining independent Field Programmable Gate Array modules for specific applications has made complex their reuse and integration, also, development time and tuning of these are considerably high. In this paper work we propose a solution for this problem based on embedding Linux on a Field Programmable Gate Array, specifically using the Spartan3AN kit. Hardware/software co-design tools used to achieve this goal have permitted getting results in short time that indicates which the focus for future research is. This way it was possible to run applications commonly used to control IP-cores embedded in reconfigurable hardware. The platform obtained is stable and flexible for future deployments of both software and hardware.

Keywords: *Embedded systems,* FPGA, *Linux, MicroBlaze, Spartan3AN.*

Introducción

La industria cubana se encuentra enfrascada en incrementar la eficiencia, sustituir importaciones y lograr soberanía tecnológica en las diferentes esferas nacionales de la producción y los servicios. Uno de los ejemplos más claros en este sentido se puede apreciar en la coordinación nacional para el desarrollo de tecnología de monitoreo y diagnóstico industrial.

La rápida evolución que ha tenido la electrónica digital y el impacto en el mercado de este tipo de sistemas ha hecho que CDEA preste especial interés en las nuevas características y potencialidades de los dispositivos FPGA (del inglés, *Field Programmable Gate Array*).

Actualmente CDEA ha utilizado la tecnología FPGA para:

1. Control de ADC (del inglés, *Analog to Digital Converter*) y DAC (del inglés, *Digital to Analog Converter*).
2. Procesamiento Digital de Señales.
3. Interfaces de comunicación del sistema general.

La obtención de un sistema (basado en FPGA) que integre todas estas funcionalidades, e incluso, lograr que tales desarrollos puedan servir de manera general en diversas aplicaciones, constituye una tarea que no ha podido ser materializada hasta el momento: la tendencia ha sido la creación de módulos independientes para aplicaciones específicas e interconectarlos entre sí. El problema fundamental de esto radica en que los módulos obtenidos son muy específicos, y su reutilización (sobre todo la sincronización los módulos) en aplicaciones diferentes es compleja, además de que los tiempos de desarrollo y puesta a punto de los mismos son considerablemente elevados y aún continúa siendo largo el camino a transitar para obtener el sistema que integre las funcionalidades de digitalización, procesamiento de señales y comunicación. Este problema se podría hipotéticamente resolver embebiendo en la FPGA un sistema operativo que incluya los controladores genéricos para manejar los periféricos necesarios del sistema integral que se pretende alcanzar. Luego, los objetivos trazados en esta investigación consisten en embeber Linux en una FPGA, en particular usando el kit de desarrollo Spartan3AN, y probar la ejecución de algunas aplicaciones en dicho entorno.

Entre las muchas aplicaciones de los *FPGAs* en el campo de los sistemas embebidos está el desarrollo de interfaces para el monitoreo y control de procesos remotos, una de las ideas válidas sería la utilización de un servidor web que recolecte los datos provenientes de un sistema de sensores, presentando la información en forma coherente e intuitiva, para que, de forma automática o supervisada se accionen un conjunto de actuadores según sea necesario.

Materiales y métodos

A continuación se brinda una descripción de todos los elementos que intervienen en el proceso de embeber Linux en FPGA, relacionando las diferentes unidades funcionales.

Cuando se habla de “componentes *hardware*” en el marco de esta investigación, se hace referencia, no sólo a las partes del sistema que se pueden tocar físicamente, sino además, aquellas que son descritas e implementadas por medio de un lenguaje sintetizable, típicamente algún HDL (del inglés, *Hardware Description Leanguaje*). Indistintamente se usa el término *hardware* para nombrar una parte física del sistema o un bloque de lógica reutilizable también llamado ip-core.

El kit de desarrollo Spartan3AN

El *kit* de desarrollo Spartan3AN es una plataforma de bajo costo, especialmente útil para entornos educativos. Su corazón es un FPGA Spartan3AN de 700 mil compuertas. En la Figura 1 se muestran con más detalle los componentes que forman el *kit*.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 1, Enero-Marzo, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 44-54
http://rcci.uci.cu

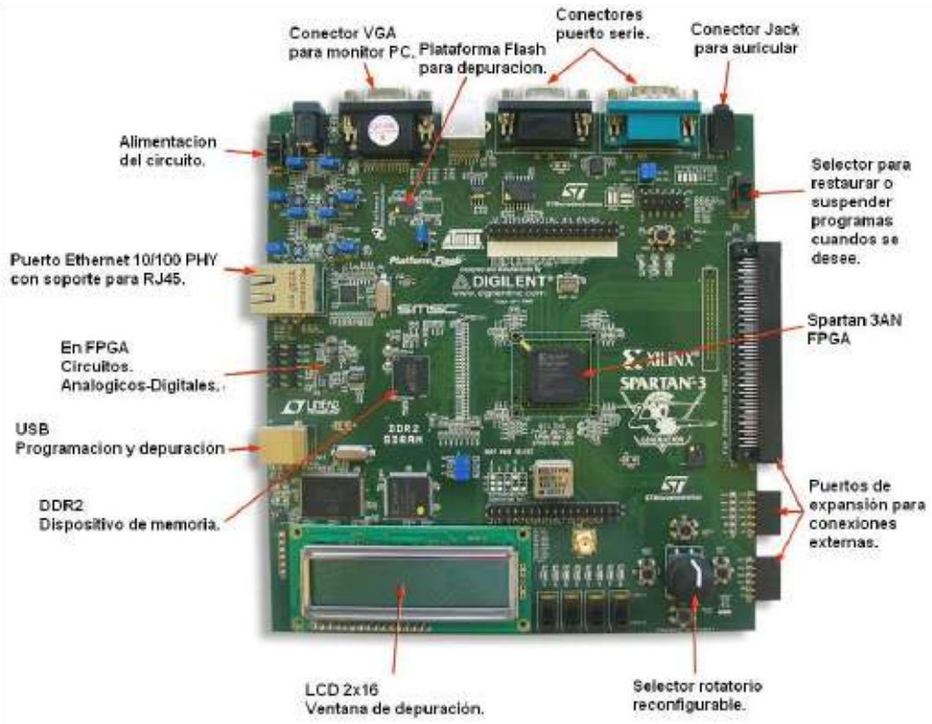

Figura 1. Elementos que conforman el *kit* Spartan3AN.

Este *kit* es totalmente compatible con las versiones de las herramientas Xilinx ISE 10 o superiores, incluida la edición WebPack. Para esta investigación se utilizó la versión 12.4 de estas herramientas (Xilinx Inc. , 2008).

El procesador embebido MicroBlaze

MicroBlaze es un procesador *soft-core* con un número reducido de instrucciones (RISC), optimizado para ser implementado sobre los FPGA de Xilinx. A pesar de no tener el código fuente disponible permite un alto grado de configuración. Sus principales características son:

- Instrucciones de 32 bits con tres operandos y dos modos de direccionamiento.
- *Bus* de direcciones de 32 bits.
- Arquitectura Harvard (*Buses* de instrucciones y datos separados).
- 32 registros de propósito general y cinco de propósito especial, ambos de 32 bits.
- Soporta pipeline.
- Soporta varios tipos de buses: OPB, PLB, LMB.
- Caché de instrucciones y datos (Xilinx Inc., 2008).

El procesador MicroBlaze tiene una arquitectura de memoria Harvard, o sea el acceso a instrucciones y datos se hace en espacios de direcciones y *buses* separados. Cada espacio de direcciones tiene un tamaño de 32 bit. Los rangos de memoria de datos e instrucciones pueden ser mapeados en la misma memoria física (Herrera, 2008).

Memorias

Un sistema embebido utiliza memoria de dos tipos: RAM y ROM. La RAM será utilizada como memoria de datos y de instrucción, en esta se ejecutan los programas, se implementa el sistema de archivos, se guardan las variables y datos de múltiples tipos. En el caso del sistema operativo, servirá para todo el manejo y planificación de procesos, manejo de interrupciones, intercambio de datos, entre otros.

(Fernández y García, 2005). La ROM se necesita para poder almacenar las imágenes del sistema operativo, del sistema de archivos y del *boot-loader* así como las variables de entorno. La principal razón para usar memoria de este tipo, en determinado sistema, es que el sistema funcione correctamente sin interacción humana de ningún tipo una vez se le aplique energía. Se requiere un tipo de memoria no volátil para esto y una determinada configuración para que el sistema direccione las imágenes a cargar.

El *kit* Spartan3AN posee memorias RAM y ROM externas al FPGA. La memoria RAM del fabricante Micron es una DDR2 SDRAM de 512 Mbit y la ROM es una Flash NOR paralela de 32Mbit.

Hardware adicional

El *kit* Spartan3AN incluye varios dispositivos de entrada/salida (E/S), que permiten ser usados en varios diseños, sin la necesidad de otro componente físico adicional. Cuatro botones y cuatro conmutadores proporcionan dispositivos de entrada para uso general y ocho LED proporcionan dispositivos de salida. Para el manejo de estos dispositivos, se usa el ip-core de Xilinx xps_gpio. El ip-core XPS GPIO proporciona una interfaz entre dispositivos genéricos de E/S y el *bus* PLB. Tiene múltiples opciones de configuración como son: ancho del canal, si es entrada, salida o bidireccional, interrupciones etc. (XILINX, 2010). El *kit* Spartan3AN posee un puerto serie de dos hilos sobre la base de un convertidor de voltaje de ST Microelectronics ST3232, que convierte los niveles de señal utilizada por RS232 a señales utilizadas por el FPGA. El puerto serie es útil para muchas aplicaciones, y, en particular para el trabajo con el procesador integrado de Xilinx MicroBlaze como dispositivo estándar de entrada y salida. Para el manejo del puerto serie, se utilizo el ip-core de Xilinx xps_uarlite. Este proporciona la interfaz de control para la transferencia asincrónica de datos a través del puerto serie y el *bus* PLB. Tiene múltiples opciones de configuración como son: velocidad, cantidad de caracteres, bit de parada, paridad, etc. y además permite transmitir y recibir de forma independiente (Xilinx, 2010).

El *kit* Spartan3AN incluye una interfaz de capa física *Ethernet Standard Microsystems* LAN8700 10/100 y un conector RJ-45. Con la implementación de un MAC (del inglés, *Ethernet Access Controller*) en el FPGA, puede conectarse opcionalmente a una red *ethernet* estándar. Todos los tiempos tienen como base el oscilador de cristal del *kit* de 50 MHz. Esta interfaz de *ethernet* está pensada principalmente para el uso en aplicaciones con el procesador *MicroBlaze*. Por lo tanto, el controlador MAC se implementa en la plataforma EDK usando el ip-core xps_ethernetlite. El IP Core Ethernet Lite está diseñado para trabajar con las normas descritas en el estándar IEEE 802.3 por medio de la interfaz MII (del inglés, *Media Independant Interface*). (Xilinx, 2009).

El boot-loader

Dentro de los componentes de software se encuentra el boot-loader o gestor de arranque. Este es un programa sencillo, encargado exclusivamente de preparar todo lo que necesita el sistema operativo para iniciar. Es común en sistemas embebidos el uso de gestores de arranque multi-etapas, en los que varios programas pequeños se van sumando, hasta que el último de ellos carga el sistema operativo.

En esta investigación se ha adoptado la solución de Petalinux que consiste en dividir el proceso de arranque en dos etapas, cada etapa es llevada a cabo por dos *software* diferentes que en conjunto completan el proceso. En la primera fase se usa FS-boot y en la segunda U-boot.

FS-boot es un gestor de arranque simple desarrollado por *PetaLogix*, destinado a servir como el mecanismo de arranque principal cuando se inicia por primera vez el procesador *MicroBlaze*. En un escenario típico de ejecución, el propósito principal de FS-boot es arrancar el *bootloader* principal desde memoria *flash* y permitir la descarga de

imágenes nuevas (Petalogix, 2009). Su compilación y depuración va ligada al desarrollo del *hardware* en la plataforma EDK. El archivo binario a descargar en el FPGA, contiene tanto la configuración del *hardware* como de este pequeño *software* inicializado en los bloques de RAM (BRAM) interna del dispositivo reprogramable.

U-boot se conoce como "*Universal boot-loader*" y originalmente fue desarrollado para la arquitectura PowerPC por Wolfgang Denx. Este ha crecido, dando soporte a un gran número de *boards* y arquitecturas de procesadores, y ahora es el gestor de arranque de facto para las implementaciones de Linux embebido (Petalogix, 2009). Dentro de las muchas utilidades de este boot-loader se encuentran, comandos: para manejo de MTD (del inglés, *Memory Technology Device*), protocolos de TCP/IP como TFTP, DHCP, NFS, BOOTP y otros, manejo de variables de entorno, soporta la ejecución de programas en formato .elf y scripts sencillos etc. La configuración se realiza a través de las directivas del preprocesador #define en una plantilla que corresponde a la arquitectura determinada. Su compilación y desarrollo van unidos al entorno *Petalinux*. El usuario decide si cuando crea las imágenes ejecutables de todo el sistema, incluye las de U-boot, o puede crear las imágenes de este de forma separada.

El Kernel µCLINUX

Un Sistema operativo (SO) o *kernel*, como también se le conoce, es un *software* que actúa de interfaz, entre los dispositivos de *hardware* y los programas usados para manejar un ordenador.

µClinux es un derivado directo del *kernel* original de Linux, adaptado para sistemas que carecen de MMU.

Sus siglas "µC" hacen referencia a "micro-controlador", para el que fue pensado en un principio. Se ha optado por él en este trabajo debido a que cada vez es mayor la comunidad de desarrolladores que lo mantienen, además de las ventajas que ofrece, como el soporte para *Microblaze*.

El hecho de que µClinux haya sido creado para soportar microprocesadores sin MMU, hace que la multitarea sea difícil de implementar. La mayoría de los archivos binarios y código fuente del *kernel* han sido reescritos para compactar y reducir el código base. Todo esto significa que el *kernel* de µClinux es mucho más pequeño que el *kernel* original de Linux 2.0, manteniendo las principales ventajas de este último, como son: estabilidad, capacidad superior en redes, y excelente soporte en el sistema de archivos (Fernández y García, 2005).

µClinux viene equipado con la torre de protocolos TCP/IP, juntos con soporte para muchos protocolos adicionales de red. La mayoría de los protocolos de red están implementados y listos para usarse. Además, es un sistema operativo apto para Internet y sistemas embebidos, donde no se necesita de una arquitectura de alto rendimiento como base. Se tiene soporte para varios sistemas de archivos tales como: NFS, Ext2, FAT32, ROMFS, JFFS y muchos más gracias al sistema de archivos virtual que desciende de Linux.

Aplicaciones

Las aplicaciones son una parte muy importante en un sistema embebido. Son, en definitiva, las que le dan el valor real al sistema. Se pueden tener aplicaciones desde una interfaz de comandos o *shell*, aplicaciones que corren en segundo plano, hasta programas de comunicación de red, conexión remota etc.

En este trabajo se han incluido una serie de aplicaciones básicas, para ilustrar las ventajas y posibilidades de usar Linux embebido sobre *hardware* reconfigurable. La mayoría de estas vienen por defecto junto con la plataforma Petalinux y por medio de un proceso de elección bastante simple se puede decidir cuál/es se incluirán en a imagen final a descargar en el *target*. Algunas de ellas son: ls, cat, df, ps, gpio-test, date, vi, printenv, sash, mkdir, mv , rm, chmod, thttpd, tftpd, y otras más.

Herramientas de desarrollo

Normalmente, en el proceso de construcción del *hardware* y el proceso de generación de *software* embebido Linux, se utilizan ambientes de desarrollo separados. Estos ambientes separados requieren una compleja sincronización, para extender las configuraciones de uno al otro. Este proceso suele ser propenso a errores si se realiza manualmente.

Para el caso del *hardware*, el ambiente de desarrollo va a depender de la plataforma de trabajo con que se cuenta, en este caso, una cuyo dispositivo FPGA es un Spartan3AN de Xilinx. Esto conduce a usar el EDK (del inglés, *Embedded Development Kit*), plataforma de la misma compañía para sistemas embebidos en sus productos. Este a su vez se integra con el ISE (del inglés, *Integrated Software Environment*) que se encarga de los procesos de síntesis e implementación. Estas herramientas fueron instaladas sobre un sistema operativo Linux Mint 11.04.

El entorno de desarrollo para un sistema basado en Linux embebido, requiere componentes de *software* como la cadena de herramientas (*toolchain*), *cross-compiler*, gestor de arranque, el *kernel* de Linux junto con el *software* GNU, bibliotecas de C y el depurador. Estos componentes deben ser incorporados en un único marco, construido y configurado para el *hardware* de destino (*target*) antes de que pueda ser utilizado para generar programas en el dispositivo de destino.

PetaLinux trae todo esto en un único entorno de desarrollo, y se integra con las herramientas ISE y Xilinx EDK. La tecnología de PetaLogix AutoConfig simplifica la sincronización entre el *hardware* y el *software*. Esta tecnología propaga los atributos del *hardware* al gestor de arranque y al *kernel* de Linux, eliminando la necesidad de la sincronización manual (Petalogix, 2009).

Petalinux “no” es un sistema operativo, es un conjunto de herramientas que integra el desarrollo del *kernel* µClinux y *software* adicional, específicamente, sobre tecnología de *hardware* reconfigurable.

Resultados y discusión

Cada plataforma de desarrollo tiene características propias, y por tanto, el proceso de embeber Linux en cada una, requiere determinadas particularidades. A continuación se describe la implementación del sistema operativo uCLinux en el *kit* Spartan3AN. Muchas de las soluciones que aquí se explican serían innecesarias para otro *target*, pero de forma general no es difícil extender la experiencia a otras plataformas de desarrollo.

Trabajo con edk y Spartan3AN

En el proceso de configuración del *hardware* embebido se usó fundamentalmente el asistente BSB (del inglés, *Base System Builder*). El BSB es un asistente que automatiza las tareas básicas de configuración del *hardware* y *software* para la mayoría de los diseños basados en las FPGA de Xilinx. Cada *kit* de desarrollo tiene sus propias configuraciones y periféricos, por lo que es necesario especificar cuál se va usar en el diseño. A continuación se listan los módulos y periféricos usados así como su configuración.

- *MicroBlaze*: Frecuencia de reloj 50 MHz, usar depuración, memoria interna BRAM 16 KB, usar *cache*, no habilitar FPU.
- XPS GPIO: Controlador para los dispositivos genéricos de E/S (*leds*, conmutadores y botones). Se dejó la configuración por defecto.
- XPS UARTLITE: Razón de baudios 115200, 8 *bits* de datos, no usar paridad, usar interrupción.
- MPMC: Este es el controlador de memoria RAM externa.
- XPS MCH EMC: Este es el controlador de memoria ROM externa.

- XPS TIMER: Tamaño de 32 *bits*, con 2 temporizadores presentes, usar interrupción.
- XPS ETHERNETLITE: Este es el controlador MAC, usar interrupción.
- El dispositivo estándar de Entrada y de Salida (STDIN y STDOUT) es el puerto RS232, la memoria de arranque es la BRAM interna (ilmb_cntlr) y no es necesario adicionar las aplicaciones de ejemplo. Una vez que se definieron estas opciones, al final del asistente se generaron automáticamente los ficheros necesarios del proyecto. Para ver con detalle todos los parámetros y opciones del proyecto se pueden editar manualmente los ficheros "system.mhs", "system.mss", "system.xmp" y "data/system.ucf".
- "system.mhs": Contiene información acerca del *hardware*. En él se declaran explícitamente todos los elementos del sistema embebido, su interconexión y los parámetros de configuración para cada uno. Cada módulo o ip-core es declarado entre las directivas BEGIN y END según una estructura predeterminada, muy similar a la programación modular descrita en VHDL. La directiva PORT define puertos de E/S en cada ip-core; cuando están fuera de los módulos hace referencia a puertos externos de la FPGA. Los parámetros de configuración se definen por la directiva PARAMETER.
- "system.mss": Contiene información acerca de *software*. En él se declaran los *drivers* asociados a cada ip-core, así como los parámetros del sistema operativo y del compilador. Cada uno de estos elementos se define en secciones limitadas por las directivas BEGIN y END. La herramienta LibGen de XPS se apoya en este fichero para la generación de todas las bibliotecas de *software* usando la tecnología BSP.
- "data/system.ucf": Se declara la asignación de pines de la FPGA para cada puerto externo del sistema embebido.
- "system.xmp": Es el fichero principal que describe todos los detalles del proyecto en desarrollo.

Configuración del software

Después que la plataforma de *hardware* se ha definido, se requiere generar una colección de parámetros o bibliotecas de *software*, basadas en la arquitectura subyacente del *target*.

En lugar de modificar el *software* manualmente para trabajar con la nueva configuración de *hardware*, este se puede configurar de forma automática rápidamente. Esto es especialmente valioso durante las primeras fases del diseño, cuando el *hardware* puede estar cambiando con frecuencia. Para esto se usa la tecnología subyacente en la plataforma XPS llamada MLD (del inglés, *Micro-processor Library Definition*). (Cameron, 2005) Petalinux incluye un BSP propio que se integra con la aplicación en desarrollo sobre la plataforma XPS.

Los ficheros BSP de Petalinux se encuentran ubicados dentro de la carpeta "hardware/edk_user_repository".

Para la configuración del *software* se editó el fichero "system.mss" en la sección OS, y además se modificó la variable ModuleSearchPath en el fichero "system.xmp". Esta es la ruta donde el EDK encontrará los ficheros ".mld" y ".tcl" necesarios para la generación de las bibliotecas de *software* usando la tecnología BSP de Petalinux.

Trabajo con Petalinux

Ya terminada la generación del *hardware*, las bibliotecas de *software* y la programación de la FPGA, se pasa al desarrollo en el entorno Petalinux. Es útil en este proceso el manual de usuario (Petalogix, 2009). Todas las operaciones en este entorno de desarrollo se ejecutan por comandos en la consola estándar del *host*. La edición del código fuente se realizó con un editor de texto común, en este caso Gedit.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 1, Enero-Marzo, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 44-54
http://rcci.uci.cu

En el próximo paso se agregó la plataforma que se ha creado al entorno Petalinux (ver Figura 2). Específicamente este comando crea una determinada estructura de directorios y plantillas de configuración con valores por defecto para la plataforma creada. Los parámetros indican el nombre del proveedor, del *kit* y la versión del *kernel* que va a usar el sistema embebido final.

```
 cd $PETALINUX/software/petalinux-dist
petalinux-new-platform -v Xilinx -p Spartan3AN -k 2.6
ew platform for Xilinx Spartan3AN successfully created
```

Figura 2. Agregar la plataforma creada al entorno Petalinux.

Posteriormente se sincronizó las bibliotecas de *software* generadas con EDK y el BSP de Petalinux, al árbol de directorios de Linux y el U-boot (ver Figura 3). El comando "petalinux-copy-autoconfig" es fundamental para la plataforma Petalinux, ya que ahorra una gran cantidad de tiempo y de posibles errores. Cada vez que se realizan cambios en el *hardware* se repite esta operación ya que el *software* no tiene otra manera de "enterarse" de los cambios y pueden surgir errores graves de ejecución. Los parámetros de este comando tienen que ser los mismos que los del paso anterior.

```
$ cd $PETALINUX/hardware/user-platform/Spartan3AN
$  petalinux-copy-autoconfig  -v  Xilinx  -p  Spartan3AN  -k  2.6  INFO:
Attempting vendor/platform auto-detect
INFO: Auto-detected Xilinx/Spartan3AN combination.
Auto-config file successfully updated for Xilinx Spartan3AN
```

Figura 3. Sincronización de las plataformas *hardware* y *software*.

Ahora sigue la selección de la plataforma que se ha agregado y sincronizado. Este es el paso inicial para construir el *kernel* del *target*. La selección consiste en asociar una colección de configuraciones del *kernel* con una plataforma en particular. Esto se realizó a través de un entorno de ventanas xWindow como el que muestra la figura 5, ejecutando el comando que aparece en la Figura 4.

```
 $ cd $PETALINUX/software/petalinux-dist
$ make xconfig
```

Figura 4. Ejecución del gestor de configuraciones.

Revista Cubana de Ciencias Informáticas
Vol. 7, No. 1, Enero-Marzo, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. 44-54
http://rcci.uci.cu

Figura 5. Selección de la plataforma.

Seguidamente se configuró el *kernel* (ver figura 6**).** Estas configuraciones que se realizan visualmente por el entorno de ventanas, se guardan automáticamente en ficheros dentro del directorio asignado para la plataforma en particular.

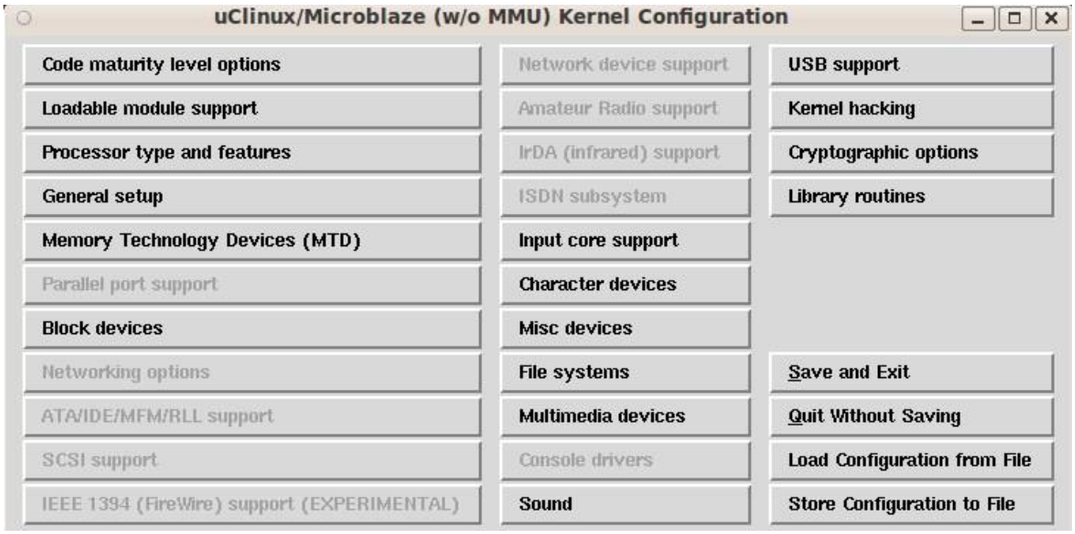

Figura 6. Configuración del *kernel*.

Por último se configuraron las aplicaciones de usuario que serán incluidas en el sistema embebido. Los parámetros del sistema tales como: contraseña del usuario root, nombre del *target*, particiones en *flash* y sistema de archivos, pueden ser cambiados en la sección "*System Settings*" (ver Figura 7).

Figura 7. Configuración de aplicaciones.

Una vez finalizada la configuración, el *kernel* está listo para ser compilado. Para ello se ejecutó el comando que aparece en la Figura 8. Este proceso se demora algún tiempo en completarse, apareciendo una serie de mensajes en la consola. Si se muestra algún mensaje de error, este debe ser corregido (editando el código fuente o cambiando la configuración) y se volvería a repetir el proceso.

```
$ cd $PETALINUX/software/petalinux-dist
$ yes "" | make oldconfig dep all
```

Figura 8. Comando para compilar el *kernel*..

Cuando el proceso de construcción ha terminado, todas las imágenes correspondientes, ya listas para descargar en el *target*, son puestas en el directorio "$PETALINUX/software/petalinux-dist/images". Estas imágenes consisten en el código del *kernel* y el U-boot debidamente compilado y en diferentes formatos, según la aplicación específica de cada uno.

El último paso es descargar y ejecutar el *kernel* de Linux. Al finalizar la inicialización del *kernel* se logró obtener el sistema operativo Linux, embebido en el *kit* Spartan3AN.

Este sistema operativo obtenido es totalmente funcional y las aplicaciones que contiene son las básicas que se pueden encontrar en cualquier ordenador con Linux con soporte para ethernet. Administración del sistema de archivos, de procesos, manejo de periféricos, intérprete de comandos, servidor web y servidor ftp son algunas de las funciones que este realiza.

Para comprobar interacción desde el sistema operativo con los periféricos de E/S gpio se utiliza la aplicación gpio-test.

Conclusiones

A través de este proyecto de investigación se ha podido constatar que se puede lograr la integración de diferentes módulos y aplicaciones generales (tanto de *hardware* como de *software*) con un alto grado de flexibilidad y funcionalidad. Para demostrar esto, se ha obtenido Linux embebido en un FPGA Spartan3AN ejecutando aplicaciones generales sobre dicho entorno; usando las herramientas de codiseño *hardware*/*software* sobre dispositivos de *hardware* reprogramable, lo cual corrobora que es posible cambiar favorablemente el enfoque de obtención en FPGA de módulos independientes para aplicaciones específicas (que hace que su reutilización en aplicaciones diferentes sea compleja y que los tiempos de desarrollo y puesta a punto de los mismos sean considerablemente elevados) hacia el de sistemas que integren soluciones generales de *hardware*/*software* con flexibilidad.

Referencias

- CAMERON, G. Configure and Build the Embedded Nucleus PLUS RTOS Using Xilinx EDK. Embedded magazine, marzo 2005, issue 1: p. 27-29.
- FERNÁNDEZ, A. M. y GARCÍA, L. G. Estudio y evaluación del sistema operativo µClinux aplicado a sistemas embebidos móviles de alta complejidad. Medellin: Universidad de Antioquia, 2005.
- HERRERA, R. *Síntesis y evaluación de un procesador DSP empotrado en FPGA.* Pinar del Río: UPR "Hermanos Saíz Montes De Oca", 2008.

- PETALOGIX Inc. Petalogix. [en línea] 2009 [Consultado: marzo de 2010]. Disponible en: [http://www.petalogix.com/resources/documentation/petalinux/userguide/Bootloaders/FSboot].
- PETALOGIX Inc. PetaLinux User Guide. [en línea] 2009 [Consultado: marzo de 2010]. Disponible en: http://petalogix.com/resources/documentation/petalinux/userguide].
- XILINX Inc. Spartan-3A/3AN FPGA Starter Kit Board User Guide. [en línea] 2008 [Consultado: septiembre de 2011]. Disponible en: [http://www.xilinx.com/support/documentation/boards_and_kits/ug334.pdf].
- XILINX Inc. MicroBlaze Processor Reference Guide. [en línea] 2008 [Consultado: febrero de 2010]. Disponible en: [http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/mb_ref_guide.pdf].
- XILINX Inc. XPS General Purpose Input/Output. Product Specification. [en línea] 2010 [Consultado: septiembre de 2010]. Disponible en: [http://www.xilinx.com/support/documentation/ip_documentation/xps_gpio.pdf].
- XILINX Inc. XPS UART Lite. Product Specification. [en línea] 2010 [Consultado: septiembre de 2010]. Disponible en: [http://www.xilinx.com/support/documentation/ip_documentation/xps_uartlite.pdf].
- XILINX Inc. XPS Ethernet Lite Media Access Controller. Product Specification. [en línea] 2009 [Consultado: septiembre de 2011]. Disponible en: [http://www.xilinx.com/support/documentation/ip_documentation/xps_ethernetlite.pdf].