Reconfiguración parcial en aplicaciones de sistemas digitales utilizando Linux como herramienta para el desarrollo

Partial reconfiguration in digital systems applications using Linux as a tool for development

Recibido: diciembre 5 de 2012
Aprobado: diciembre 10 de 2012

Carlos I. Camargo B.*, Andrés. M. Sánchez V.**, Nelson F. Rosas***

Resumen

Las características de las FPGA, tales como la reconfiguración parcial y dinámica, permiten el diseño de sistemas digitales, y en especial de sistemas embebidos, para aplicaciones con fuertes restricciones temporales. En este trabajo se presenta el desarrollo de una plataforma embebida que permite la reconfiguración de dispositivos externos. Se muestran las herramientas desarrolladas para la configuración de FPGA, así como las ventajas que trae la utilización de la configuración parcial y dinámica junto con un sistema operativo basado en Unix, en este caso uClinux.

Palabras clave

Linux Embebido, reconfiguración parcial, Sistemas Embebidos, SoC y FPGA.

* Ingeniero Electricista, Universidad Nacional de Colombia, Bogotá - Colombia. Magíster en Ingeniería Eléctrica, Universidad de los Andes, Bogotá – Colombia. Doctor en Ingeniería Eléctrica, Universidad Nacional de Colombia, Bogotá - Colombia. Profesor Asociado, Universidad Nacional de Colombia, Sede Bogotá, Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica. E-mail: cicamargoba@unal.edu.co

** Ingeniero Electrónico, Universidad Nacional de Colombia, Bogotá - Colombia. Magíster en Ingeniería Electrónica y de Computadores, Universidad de los Andes, Bogotá – Colombia. E-mail: dav18284@gmail.com

*** Ingeniero Electrónico, Universidad Nacional de Colombia, Bogotá - Colombia. Magíster en Ingeniería de Telecomunicaciones, Universidad Nacional de Colombia, Bogotá - Colombia. Profesor Asociado, Universidad de San Buenaventura, Bogotá, Facultad de Ingeniería, Director de Ingeniería Electrónica y de Ingeniería de Telecomunicaciones. E-mail: neferoji@gmail.com, nfrosasj@unal.edu.co

Abstract

The characteristics of FPGAs, such as partial and dynamic reconfiguration, allowing the design of digital systems, especially in embedded systems for applications with strong time restrictions. This paper presents the development of an embedded platform that allows the reconfiguration of external devices. The tools for configuring FPGAs are explained bringing the advantages of using partial and dynamic configuration with a Unix-based operating system, in this case uClinux.

Keywords

Embedded Linux, Embedded systems, Partial Reconfiguration, SoC and FPGAs.

I. Introducción

Los dispositivos basados en lógica reconfigurable como las FPGA fueron utilizados inicialmente para la implementación de tareas que exigían demasiado al procesador, por ejemplo, cuando se requerían muchos ciclos de máquina. Sin embargo, la creciente capacidad de integración ha permitido el desarrollo de circuitos más complejos en un solo chip (SoC). En particular, esto ha permitido que se puedan implementar procesadores y memorias en un dispositivo lógico programable. Estos dispositivos han sido dotados con características que permiten la reconfiguración parcial y dinámica soportando, por ejemplo, actualizaciones de *hardware* desde sitios remotos, algoritmos adaptativos de *hardware* y manejo más eficiente de espacio y potencia [4].

Actualmente los sistemas embebidos presentan una demanda creciente [1], y se exige de ellos la prestación de servicios tales como: interconexión en red, manejo de sistemas de archivos, interfaz adecuada con el usuario, entre otros [2]. El diseño de sistemas digitales, y en especial de sistemas reconfigurables, debe ir acompañado de un conjunto de herramientas de alto nivel apropiadas para reducir el tiempo de desarrollo, manteniendo un desempeño aceptable. Esto ha impulsado el desarrollo de herramientas de *software* que intentan hacer transparente el co-diseño *software-hardware* manteniendo las prestaciones de la tecnología.

Los diseñadores tienen entonces una gran gama de herramientas para el diseño de sistemas, por lo que se espera que estos últimos sean más eficientes en términos de espacio y capacidad de cómputo. Sin embargo, es importante el desarrollo de una plataforma genérica que permita reducir costos en el desarrollo de prototipos e implementación de sistemas reconfigurables [3].

Este trabajo está organizado de la siguiente forma: en la sección II se hace una breve descripción del *hardware* reconfigurable. En III se muestran las posibilidades que existen para implementar SoCs en las FPGA. En la sección IV se describe la importancia de contar con un sistema operativo, en este caso uClinux, para coordinar las tareas que debe realizar un sistema que use computación reconfigurable. Finalmente en V, se muestra el diseño de la plataforma y en la VI se estudia la capacidad de cómputo de la misma.

II. Materiales y métodos

Las FPGA se pueden reconfigurar, a través de un archivo llamado bitstream, para que implementen diferentes circuitos lógicos. Este archivo se almacena en una memoria de configuración (volátil) interna y describe la forma en la que se deben configurar todos los bloques del dispositivo. En los últimos años los fabricantes de FPGA han incorporado características que permiten realizar una reconfiguración parcial [4], por ejemplo, en el dispositivo se carga un *bitstream*, de menor tamaño que el que se utilizó para configurar la FPGA inicialmente, el cual contiene información de la configuración de un sector de la FPGA manteniendo las funciones implementadas en los otros sectores. El proceso de reconfiguración parcial se puede realizar de forma dinámica, el dispositivo no es enviado al estado de *Reset*, manteniendo el funcionamiento de los sectores no reconfigurados mientras ocurre el proceso, por ejemplo, una función que puede ser asignada a uno de los módulos consiste en encargarse de las tareas de reconfiguración.

Xilinx Corporation[1] ha desarrollado dos flujos de diseño para implementar la configuración parcial o parcial dinámica, en sus FPGA [4], las cuales corresponden a flujo diferencial y modular. El primero consiste en realizar cambios a un diseño implementado previamente en la FPGA (hechos con FPGA_Editor2 o modificando el código HDL) y después generar un *bitstream* que contiene las diferencias entre el diseño inicial y el modificado. Este flujo de diseño permite una rápida descarga del archivo de configuración porque solo se alteran sectores puntuales de la FPGA. Sin embargo, es un proceso que consume tiempo para generar el *bitstream* diferencial porque es necesario sintetizar nuevamente todo el diseño, así los cambios realizados sean pequeños.

El flujo de diseño modular busca disminuir el tiempo necesario para generar el *bitstream* parcial, en el cual se divide el área de la FPGA en diferentes secciones de acuerdo con ciertas restricciones impuestas por el fabricante [4], [5]. El *bitstream* describe la configuración de uno de estos sectores que ha sido instanciado como un módulo del sistema. La síntesis de este módulo, la cual corresponde a una parte del sistema total, tarda menos tiempo que el gastado en la reconfiguración diferencial.

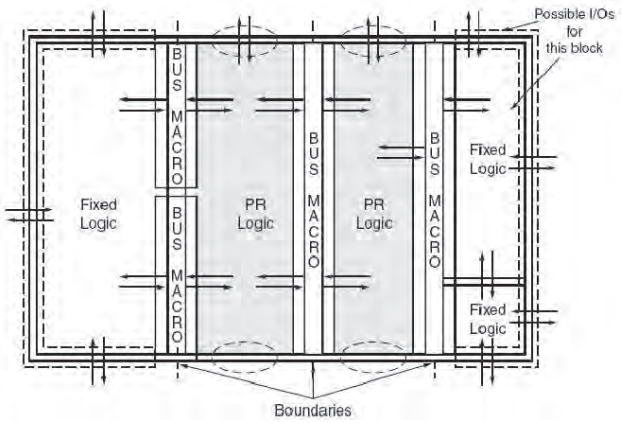

Figura 1. Disposición del área fija y reconfigurable en una FPGA.
Fuente: tomado de [4].

1 Xilinx Corporation, enlace http://www.xilinx.com/

En la figura 1 se muestra un ejemplo de la forma como se dividiría el área de una FPGA para implementar la configuración modular. Existen diferentes sectores que implementan circuitos fijos (*fixed logic*) y reconfigurables (*PR logic*), cada uno de los cuales está confinado a un área fija. Siempre que se intercambie señales entre un módulo fijo y uno programable debe existir en la frontera un bloque denominado bus macro. Este bus macro es usado para garantizar que cada vez que se presente una reconfiguración, las señales externas del bloque modificado permanezcan en el mismo lugar evitando cortos y posibles daños al dispositivo.

La figura (2) muestra el diagrama, utilizando FPGA_Editor, de un proyecto sencillo implementado con el flujo modular, el cual consiste en dos módulos: un oscilador y un registro, que componen el módulo fijo, y un subsistema que puede configurarse como sumador o restador. El oscilador en el bloque fijo tiene el propósito de evaluar la capacidad de reconfiguración dinámica del dispositivo (continúa funcionando en el proceso de reconfiguración del sumador/restador). El registro captura los datos de entrada y los pasa a través del bus macro para que el bloque reconfigurable los opere de acuerdo a la función, suma o resta, que esté implementando.

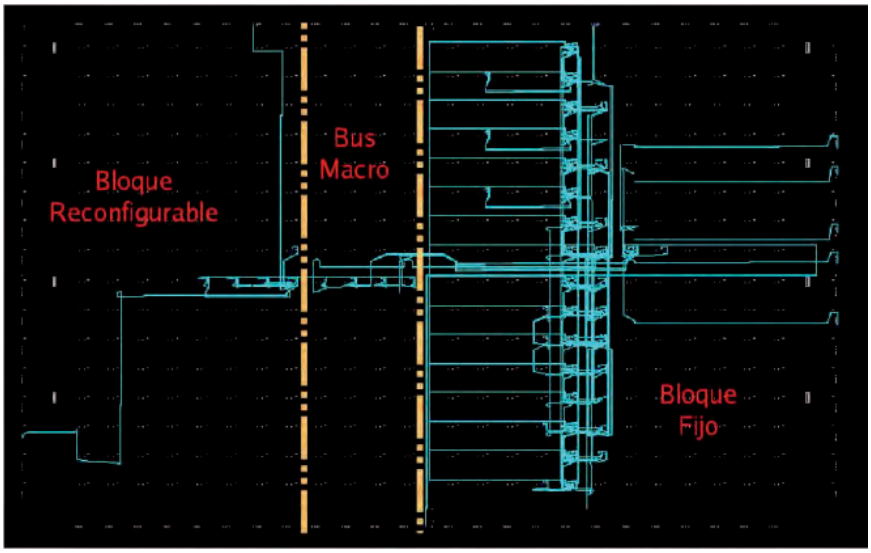

Figura 2. *Layout* del diseño visto con FPGA_Editor.

III. Sistemas embebidos en FPGA

Implementar un microcontrolador en una FPGA tiene múltiples ventajas: permite la reutilización y optimización de diseños anteriores evitando la obsolescencia de los mismos al proporcionar una plataforma flexible [3]. Por otro lado, se logra disminuir el tiempo de desarrollo, al realizar actualizaciones de Hardware y Software de forma local y remota. MicroBlaze [10] es un procesador (*soft-core*) RISC de 32 bits, arquitectura Harvard diseñado por Xilinx Corporation para sus FPGA Virtex y Spartan-II/3, es distribuido a través del Embedded Development Kit (EDK). La figura 3 presenta un diagrama de bloques del procesador, en el cual se observa la distribución de los componentes de la CPU, los periféricos embebidos en la FPGA y dos memorias externas.

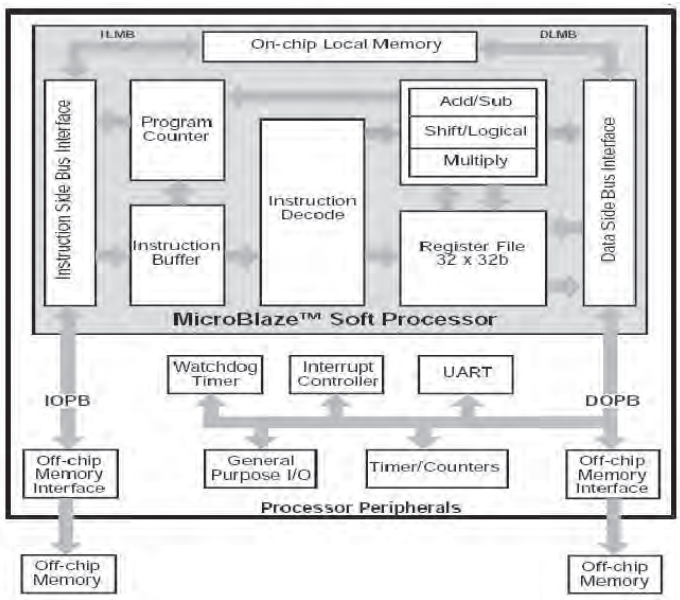

Figura 3. Diagrama de bloques del núcleo del procesador MicroBlaze.
Fuente: adaptado de [10].

MicroBlaze se comunica con los periféricos y bloques de memoria a través de 2 buses: LMB (*Local Memory Bus)* y OPB (*On-chip peripheral Bus*). El primero es un bus asíncrono de alta velocidad, el cual admite solo un maestro y garantiza el acceso en un ciclo de reloj a la memoria RAM interna. El segundo, desarrollado por IBM, es usado para conectar periféricos y memorias externas, soporta varios maestros.

IV. Sistemas operativos y computación reconfigurable

Dadas las ventajas que ofrece el diseño de sistemas basados en *hardware* reconfigurable, es necesario crear una arquitectura apropiada que permita un manejo del proceso de reconfiguración. En la actualidad este proceso se realiza con la intervención de herramientas especializadas diseñadas por el fabricante. En particular, Xilinx a través de su entorno de desarrollo EDK, proporciona herramientas para la creación de *drivers,* realización de simulación y depuración por *software* (XMD). Xilinx utiliza herramientas GNU (gcc) para la compilación del código de cada aplicación. Estas herramientas impiden la realización del proceso de reconfiguración en un ambiente embebido ya que están diseñadas para correr en computadores personales. Es por esto que es necesario diseñar herramientas *software* y *hardware* que permitan realizar el proceso de reconfiguración parcial de forma autónoma. En [6] se han discutido las características de un sistema operativo para ser usado en computación reconfigurable, y se ha argumentado que un sistema operativo embebido representa una mejor alternativa sobre sistemas basados en microkernel.

Trabajar con Linux trae importantes ventajas, por ejemplo los sistemas operativos basados en Unix proveen un conjunto de herramientas sencillas, las cuales en conjunto, utilizando la segmentación y redireccionamiento, pueden ejecutar tareas complejas. En particular, si tenemos un *bitstream* generado por las herramientas convencionales de síntesis para ser usado en la reconfiguración de cierta FPGA, se podría ejecutar el comando:

```
$ cat bitstream.bit > /dev/port
```

En donde /dev/port representa el archivo mediante el cual se accede al puerto de configuración. Es importante notar que el *bitstream* puede estar almacenado en cualquier parte (memoria interna, externa o en un sistema de archivos remoto), sin alterar el comando utilizado.

El proceso de configuración también se puede llevar a cabo desde servidores remotos [7], simplificando el proceso de reconfiguración desde el punto de vista del usuario.

La figura 4 muestra la arquitectura del kernel de Linux. El cual está compuesto por 6 bloques: el programador de tareas (Scheduler), el manejador de memoria, la comunicación entre procesos (IPC), que administran los procesos que se realizan, y por otra parte está el sistema de archivos, la interface de red y los drivers, encargados de conectar el sistema con el exterior.

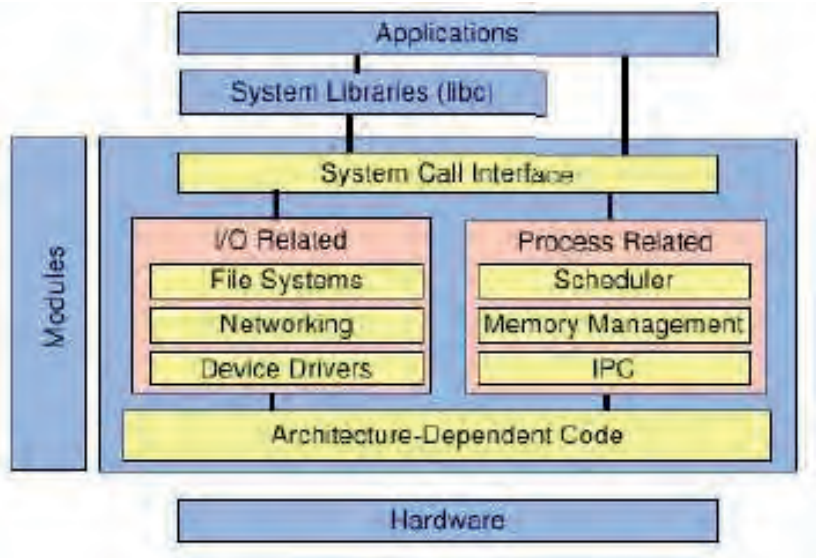

Figura 4. Representación del kernel de Linux.
Fuente: tomado de [12].

uClinux [8] es un sistema operativo basado en Linux diseñado para microprocesadores que no posee la Unidad de Manejo de Memoria (MMU), el cual ha sido portado a múltiples microporcesadores, como MicroBlaze [9]. La arquitectura del kernel uClinux es similar a la mostrada en la figura (4), salvo que no existe el bloque manejador de memoria. La falta de MMU implica cambios importantes en el sistema, por ejemplo, no existe protección de memoria ni memoria virtual, la pila no se redimensiona automáticamente y algunas llamadas al sistema se ven afectadas, haciendo que la multitarea sea compleja.

El desarrollo de uClinux se ha llevado a cabo mediante el uso de las herramientas libres del proyecto GNU tales como gdb (GNU *debugger*), gcc (GNU *Compiler Collection*) y binutils (colección de herramientas como ld (*linker*) y as (*assembler*). Las distribuciones de uClinux que se trabajaron se basan en el kernel 2.4 y se utilizaron las versiones de kernel 2.5 y 2.6.

V. Plataforma de desarrollo

En la figura 5 se presenta la plataforma utilizada, diseñada por el profesor Carlos Camargo con la compañía de la empresa Symbiosis [11], la cual tiene enormes ventajas para implementar un sistema embebido. El núcleo es una FPGA (XC3S400PQ208) fabricada por Xilinx Corporation, la cual se utiliza principalmente para implementar un MicroBlaze junto con algunos periféricos. La plataforma cuenta con una memoria SDRAM de 8MBIT, dos memorias flash: memoria NOR de 16-MBIT y memoria NAND de 1GBIT, una memoria PROM (XC04S), que hace parte de la cadena jtag, para configurar la FPGA al ser alimentada, un controlador Fast Ethernet y un reloj en tiempo real.

Figura. 5 Tarjeta de desarrollo utilizada.

La figura 6 se presenta el diagrama de bloques del sistema implementado, particularmente en la FPGA se sintetizó un microblaze. Este procesador cuenta con una memoria para almacenar datos e instrucciones (RAM_1), la cual se accede por medio de dos buses (ilmb)3 y (dlmb)4. La memoria (Ram_2) provee, a través de la interfaz OPB_bram_if_ctl, memoria caché al procesador (4kbytes para datos y 4kbytes instrucciones).

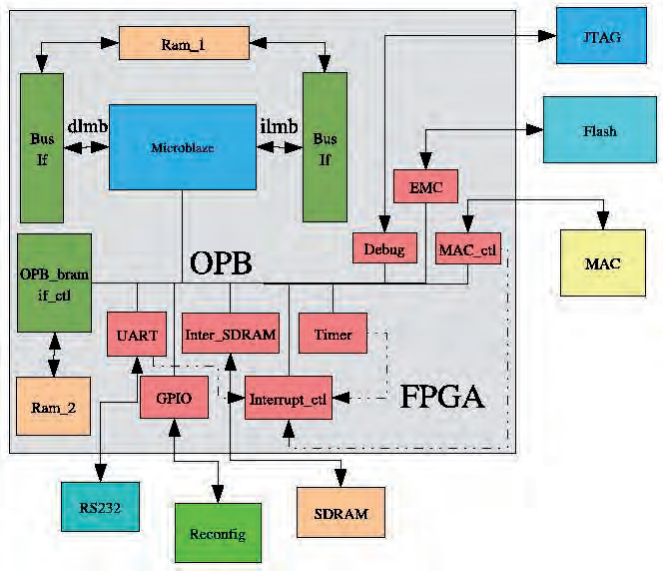

Figura 6. Diagrama de bloques de la plataforma desarrollada.

Existen 8 periféricos, que se comunican con la CPU a través del OPB (*On chip Peripheral Bus*):

- Debug: interfaz entre el sistema y un computador para realizar acciones de depuración.
- EMC (*Externa Memory Controller*): interfaz entre el microblaze y la memoria Flash.
- MAC_ctl: controlador de la Ethernet MAC.
- Timer: timer de 32 bits utilizado para manejar la multitarea por medio del sistema operativo (uClinux).
- Interrupt_ctl: controlador de interrupciones.
- Inter_SDRAM: interfaz entre la memoria SDRAM externa y el OPB.
- GPIO: puerto de 4 bits por medio del cual, mediante el protocolo JTAG, se reconfigura el dispositivo externo.
- UART: comunicación serial por medio de la cual se pueden ver los mensajes del sistema operativo.

Para compilar el kernel de Linux es necesario obtener las fuentes [9], compuestas del kernel en sí (uClinux-2.4.x) y las librerías y programas de usuario (uClinux-dist). Las fuentes se compilan para una arquitectura específica, de acuerdo al archivo que describe la arquitectura del procesador (auto-config.in), que es generado por las herramientas de Xilinx Corporation. Una vez generado este archivo, se debe ingresar a la carpeta uClinux-dist y ejecutar el comando make menuconfig. En este momento se despliega un menú en el cual se escoge el fabricante. A continuación se despliegan diferentes menús en los que se puede configurar a la media la imagen del kernel. Una vez seleccionadas las opciones del kernel se ejecutan los comandos *make dep* y *make*, de la forma usual, para iniciar la compilación del kernel, después de la cual se obtiene este en binario.

Para realizar el proceso de reconfiguración de una FPGA conectada externamente a la plataforma, se creó un módulo[2] de kernel que maneja el puerto Reconfig (figura 6), utilizando los pines de entrada/salida (GPIO). Cada dispositivo es visto por el kernel como un archivo, usualmente localizado en el directorio /dev. En la plataforma diseñada, el puerto Reconfig se accede mediante el archivo /dev/jtag, al leer este archivo el *driver* detecta si existe algún dispositivo conectado. Por ejemplo, escribir algún *bitstream* en este archivo origina la configuración de la FPGA externa.

Para la inicialización del sistema operativo es necesario almacenar su imagen en una memoria no volátil, copiarla a un medio de almacenamiento volátil (memoria RAM) y ejecutar la imagen del kernel desde allí. En este caso, la imagen del kernel de Linux se

2 El módulo es una sección de código que puede ser añadido y eliminado del kernel sin necesidad de reiniciar el sistema, de esta forma se evita compilar un kernel cada vez que se necesite utilizar estas nuevas características.

encuentra almacenada en la memoria flash. Cuando se alimenta la plataforma, el procesador MicroBlaze empieza a ejecutar un programa almacenado en la memoria local Ram_1, el cual copia el contenido de la flash en la SDRAM, después empieza la ejecución del kernel desde la SDRAM.

VI. Conclusiones

El diseño de sistemas digitales debe ir acompañado de herramientas apropiadas para aprovechar el rápido crecimiento de la capacidad de integración de los dispositivos actuales. La utilización de un sistema operativo, para administrar y controlar los recursos de un sistema, se está convirtiendo en un requisito básico, ya que permite reducir considerablemente tiempo de diseño sin afectar significativamente el desempeño.

El *hardware* Reconfigurable brinda nuevas posibilidades de diseño. Sin embargo, es importante desarrollar herramientas que permitan aprovechar el potencial que ofrece. uClinux es un sistema operativo apropiado para tal fin, porque al seguir la filosofía de los sistemas basados en Unix, permite la ejecución de tareas poderosas mediante comandos sencillos.

La utilización de un sistema operativo libre trae ventajas importantes porque disminuye los costos en la fabricación de dispositivos comerciales. Existen diversas herramientas que facilitan la labor de los diseñadores, además, al tener acceso al código fuente, se pueden crear sistemas ajustados a las necesidades particulares, imposibles de conseguir de otra forma.

Referencias

[1] J. F. Fuller, E. F. Fuchs, and K. J. Roesler, «Influence of harmonics on power distribution system protection,» *IEEE Trans. Power Delivery*, vol. 3, pp. 549-557, Apr. 1988.

[2] E. Sánchez. *El futuro de la infomática y la desaparción del computador*. Medellin, 2005.

[3] N.W. Bergmann J.A.Williams and X. Xie. Fifo communication models in operating systems for reconfigurable computing. 13th Annual *IEEE Symposium on Field-Programmable Custom Computing Machines,* 2005.

[4] J. Williams N. Bergmann and P. Waldeck. *Egret: A Flexible Platform for Real-Time Reconfigurable System on Chip. School of ITEE*, The University of Queensland. Brisbane, Australia.

[5] Xilinx Inc. *Two flows for partial reconfiguration: Module Based and difference based*, 2004. Application Note 290.

[6] G. Mermoud. *A Module-Based Dynamic Partial Reconfiguration tutorial. Logic Systems Laboratory*, École Polytechnique Fédérale de Lausanne, Noviembre 2004.

[7] J. Williams and N. Bergmann. *Reconfigurable Linux for Spaceflight Applications*. School of ITEE, The University of Queensland. Brisbane, Australia.

[8] J. Williams and N. Bergmann. *Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip*. School of ITEE, University of Queensland Brisbane, Australia.

[9] uclinux. Embedded Linux Microcontroller Project. Disponible en: http://www.uclinux.org/.

[10] uclinux. Microblaze uClinux Project. Disponible en: http://www.itee.uq.edu.au/ jwilliams/mblaze-uclinux/

[11] Xilinx Inc. *Microblaze Processor Reference Guide*, 2003.

[12] Proyecto linuxencaja URL Disponible en: http://www.linuxencaja.com

[13] An Introduction to the Linux architecture. Disponible en: http://wiki.cs.uiuc.edu/cs427/An+Introduction+to+the+Linux+Architecture