RECONFIGURACIÓN REMOTA DE FPGAS PARA DISPOSITIVOS MÓVILES

Juan Diego Echeverri Escobar
José Edinson AedoCobo
Grupo de Microelectrónica y Control
Universidad de Antioquia

juan@microe.udea.edu.co, joseaedo@udea.edu.co

ABSTRACT

En este trabajo, se utiliza una plataforma experimental para procesamiento multimedia constituida de un procesador y una FPGA y se propone una estrategia de reconfiguración para la FPGA, de tal manera que las funciones en *hardware* y su integración al software embebido se realicen de forma dinámica, con base en un conjunto de módulos previamente diseñados (IPs), almacenados en un servidor remoto, y descritos en XML. Los módulos pueden ser descargados a través de una red inalámbrica permitiendo al dispositivo adaptar el software y el hardware para desarrollar una tarea específica con eficiencia. Resultados experimentales sugieren que la estrategia implementada para la reprogramación de la FPGA es adecuada para explotar las potencialidades de la plataforma.

1. INTRODUCCIÓN

En la última década, el mercado de los dispositivos móviles ha crecido de manera exponencial. Las nuevas aplicaciones, en especial las orientadas a multimedia, requieren arquitecturas de alto desempeño y bajo consumo de potencia. El grupo de Microelectrónica y Control de la Universidad de Antioquia, viene trabajando en una plataforma para procesamiento multimedia con capacidad de reconfiguración en tiempo real de funciones en *hardware*, mediante el uso de FPGAs. Las funciones implementadas en *hardware* pueden ser integradas de manera dinámica en la funcionalidad de todo el sistema mediante la adaptación su software embebido.

Este trabajo propone una estrategia para la reconfiguración de la FPGA, en una plataforma constituida de un procesador y una FPGA, orientada a la integración de las funciones de *hardware* programadas en tiempo real en la plataforma con el software embebido, de tal manera que se pueda aprovechar, de forma dinámica, las nuevas funcionalidades del hardware. Por consiguiente, una terminal móvil puede ejecutar nuevas aplicaciones en el dispositivo reconfigurable, incluso adaptarse a la aparición o modificación de nuevas aplicaciones y estándares aprovechando la capacidad de reprogramación del *hardware.*

Este artículo ha sido organizado de la siguiente forma: En la sección 2, se realiza una breve descripción de trabajos reportados en la literatura relacionados con este tema. En la sección 3 se describe la estrategia de reconfiguración remota. En la sección 4, se muestra la metodología para describir los módulos de hardware. En la sección 5, se describe la estrategia para integrar los módulos de hardware considerando el uso del sistema operativo Linux en la plataforma. Los resultados obtenidos se describen en la sección 6 y finalmente en la sección 7, se establecen las conclusiones de este trabajo.

2. TRABAJOS RELACIONADOS

La utilización hardware reconfigurable y en especial FPGAs para dispositivos móviles se venido proponiendo y usando en los últimos años. En propuestas como DReAM[1], CHAMELEON[2], MOLEN[3] o RAP[4], se explota la eficiencia de los dispositivos reconfigurables para la ejecución de ciertos algoritmos, especialmente para procesamiento multimedia. La mayoría de estas propuestas utilizan un esquema de reconfiguración que permite adaptar la arquitectura a cambios en el ambiente o incluso a cambios dentro de una misma tarea.

En YaMoR[5] se explora, para robots móviles, la alternativa de utilizar reconfiguración parcial dinámica (DPR) con el fin de adaptar el *hardware* a una tarea especifica. Un conjunto de aplicaciones para la FPGA, se encuentran almacenadas en un servidor y pueden ser descargados al robot en cualquier momento utilizando una conexión inalámbrica.

Por otra parte, con relación a la manipulación de los archivos SVF para la programación de la FPGAs, en BSML[6] plantea un esquema de conversión de las instrucciones SVF a XML. Este trabajo busca que la información que se obtiene del *testing* de los sistemas sea más transportable y fácil de manipular.

3. ESTRATEGÍA DE RECONFIGURACIÓN REMOTA

En este trabajo se utiliza una plataforma constituida por un procesador i.MXL[7] (generalmente utilizado en dispositivos móviles) y una FPGA Spartan 3 XC3S200 [8] como se muestra en las figuras 1 y 2. Se busca en esta arquitectura, que el procesador ejecute tareas de menor exigencia computacional y poco paralelizables tales como el manejo del *stack* de protocolos TCP/IP, mientras la FPGA se encarga de procesos donde se pueda aprovechar un alto grado de paralelismo como decodificación de audio y video, algoritmos de criptografía, etc. La arquitectura utilizada permite mantener la FPGA apagada el tiempo que no sea necesario utilizarla y reconfigurarla cada vez que se requiera. Esto permite controlar el efecto del consumo estático de la FPGA que es relativamente alto [9] sobre el consumo total del sistema.

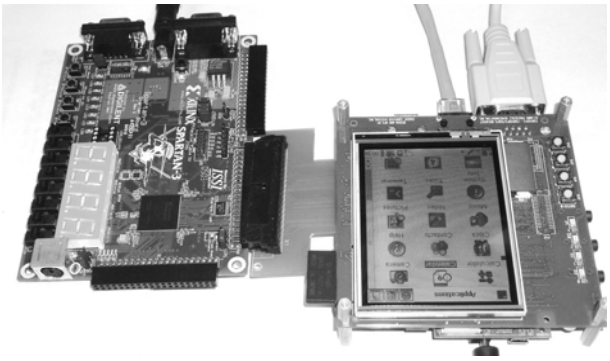

Figura 1. Plataforma multimedia Universidad de Antioquia.

Para reconfigurar de manera remota la FPGA, un conjunto de módulos de *hardware* descritos en XML, se encuentran almacenados en un servidor. De esta manera, si el dispositivo móvil requiere ejecutar una aplicación que utiliza la FPGA, busca el archivo XML que describe el módulo (la IP) inicialmente en la base de datos del dispositivo, sino la encuentra, la busca en un servidor utilizando una conexión inalámbrica (WLAN) y reprograma la FPGA. Este esquema también permite una actualización de hardware, por ejemplo en el caso de un cambio en el estándar de codificación de video, se accede al servidor y se descarga el nuevo módulo de hardware en XML a la terminal.

En la plataforma experimental usada en este trabajo se emplea una FPGA Spartan3. Para programar esta FPGA, existen varias alternativas [10]: SelectMAP, Serial y JTAG. Los métodos propietarios (SelectMAP o Serial) tienen ventaja sobre el método estándar JTAG (IEEE 1149.1) en la velocidad de programación, sin embargo, obligan a la arquitectura a trabajar con un dispositivo específico. Por este motivo, en este trabajo se decidió utilizar JTAG.

Los dispositivos que implementan el IEEE 1149.1 tienen un puerto dedicado, llamado TAP. Este conecta las señales de entrada de prueba al controlador JTAG que consiste en una máquina de 16 estados para manipular los registros internos. Las señales usadas son: *Test Data In* (TDI), *Test Data Out* (TDO), *Test Clock* (TCLK) y *Test Mode Select* (TMS), las cuales se conectan a *pines* de propósito general del procesador. Por medio de un *driver*, integrado en el sistema operacional, que controla la máquina de estados (en la sección 5 se describe con mayor detalle la estructura del software) capaz de reconfigurar la FPGA.

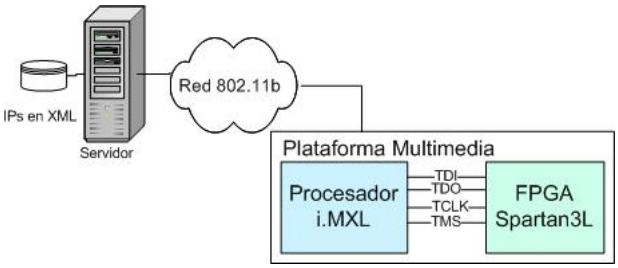

Figura 2. Arquitectura general del sistema

4. DESCRIPCIÓN DE LOS MÓDULOS DE HARDWARE

Para describir los módulos de hardware (o IPs), se utiliza el formato SVF (Serial Vector Format). SVF es una representación de operaciones de alto nivel del bus IEEE 1149 y puede ser generado por las herramientas de síntesis como el ISE de Xilinx. En este trabajo se desarrolló una aplicación, SVF2XML, que convierte el archivo SVF a XML y lo verifica considerando el *Schema* XML desarrollado.

XML define reglas para estructurar la información que conduce a crear documentos bien formados (*Well formed*). Sin embargo, debido a la libertad de XML para crear documentos, se requiere de otra herramienta que formalice las restricciones que impone la aplicación particular. Los dos métodos más conocidos son los DTD y los *XML Schema*. En este trabajo se desarrolla un *XML Schema*[11, 12] que permite validar las IPs generadas y garantizar que el dispositivo móvil sea capaz de interpretarlas.

SVF2XML es una aplicación desarrollada en JAVA capaz de convertir el archivo SVF a XML y validarlo contra el *Schema* definido anteriormente. La aplicación esta basada en el ANTLR [13], una herramienta para construir reconocedores, compiladores y traductores a partir de descripciones gramaticales. SVF2XML realiza modificaciones al archivo SVF que permite acelerar la reconfiguración y disminuir el tamaño del archivo XML como codificación base 64 para los datos y eliminación de información no utilizada por la terminal como las mascaras MASK y SMASK. En la figura 3, se muestra el procedimiento para la transformación de la descripción en VHDL de la síntesis de los módulos de hardware a una descripción en XML.

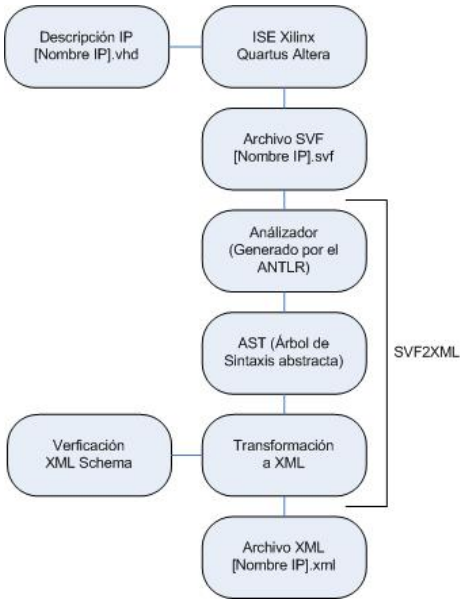

Figura 3. Transformación de módulos de hardware descritos en VHDL a XML

5. ESTRUCTURA DEL SOFTWARE E INTEGRACIÓN DE LOS MÓDULOS DE HARDWARE A LA PLATAFORMA

El sistema operativo utilizado en la plataforma experimental es un Linux 2.6.12. Como analizador XML (*parser*) se eligió Xerces-C++[14] del proyecto Apache por la facilidad de cross-compilación a la arquitectura del i.MXL,(que utiliza ARM920) y por su mayor desempeño ante alternativas de lenguajes interpretados como JAVA o Python. En este proyecto se desarrollo un *driver* de reprogramación y se integró *kerne*l del OS de la plataforma. Este *driver* es capaz de acceder a los recursos de hardware del procesador, como pines de entrada y salida, generando las señales necesarias para manipular el puerto TAP y la máquina de estados IEEE 1149-1 de la FPGA. Igualmente se desarrolló una aplicación que utiliza este *driver* para programar la FPGA a partir de una descripción en XML. En la figura 4 se muestra de forma esquemática los procedimientos desarrollados por esta aplicación.

Cuando la FPGA ha sido reconfigurada, se requiere un *driver* adicional para utilizar el módulo de *hardware*. Este lee o modifica los datos almacenados en la FPGA accediendo a la misma como un periférico conectado al bus de datos y direcciones del procesador. Este *driver* también puede ser descargado e integrado al OS para que las aplicaciones puedan usar el modulo de hardware de forma dinámica.

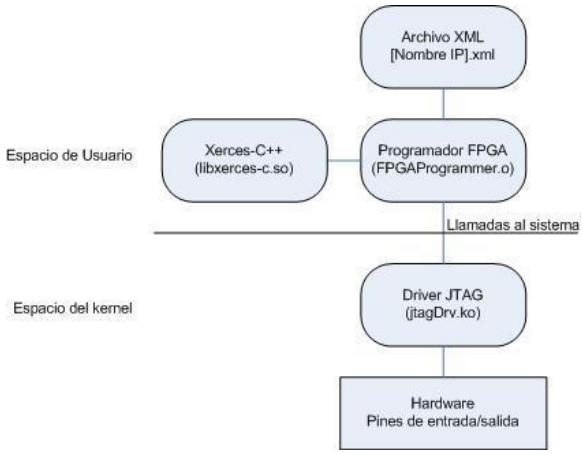

Figura 4. Aplicación en la terminal para la reconfiguración.

6. RESULTADOS

6.1 Tiempo de reconfiguración

Uno de los parámetros más importantes para la arquitectura propuesta es el tiempo de reconfiguración de la FPGA. Este tiempo debe ser limitado de tal forma que la ganancia obtenida por la paralelización del algoritmo a ser ejecutado por la FPGA, no se pierda por el tiempo utilizado en la reconfiguración de la FPGA. Existen varios factores que influyen en el tiempo de reconfiguración en la arquitectura propuesta: el *parsing* del XML, el envío de los datos al *driver* de reconfiguración y el tiempo que tarda en ser enviada la información al puerto TAP. La implementación de la aplicación y el *driver* se realizó buscando una reducción del tiempo de programación de la FPGA. La tabla 1 muestra el tiempo promedio requerido por estas tareas[1]. Es de anotar que este tiempo se puede reducir notablemente si se aumenta la frecuencia de reloj del procesador.

Tarea	Tiempo (s)
Inicialización XML	0,014
Analizador XML	0,315
Procesamiento *driver*	1,051
Señales Puerto TAP	0,93
Total	2,310

Tabla 1. Tiempo requerido para ejecutar las tareas de reconfiguración

[1] Las mediciones se tomaron con el procesador i.MXL trabajando a una frecuencia de 48 MHz.

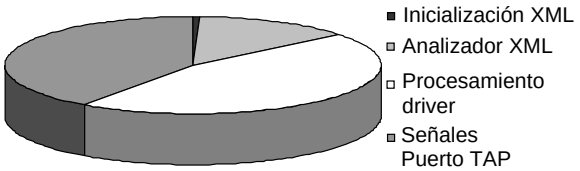

Figura 5. Efecto de cada tarea sobre el tiempo total de reconfiguración

6.2 Consumo de energía

El otro factor de peso es la energía requerida para reconfigurar la FPGA. Los dos componentes principales son el consumo por parte del procesador al ejecutar la aplicación de reconfiguración y el consumo necesario para reconfigurar la FPGA.

La plataforma multimedia utilizada para la prueba consiste de dos sistemas de desarrollo, el i.MXLiteKit de la empresa Cogent[15] y el *Spartan 3 starter board* de Digilent[16]. Para esta prueba, se caracterizó el sistema de desarrollo de Digilent con el fin de obtener el consumo de potencia cuando la FPGA no realizaba ningún procedimiento. Para el caso del i.MXLITE se tomo un promedio del consumo del sistema de desarrollo con el sistema operativo Linux en ejecución. En ambos casos, se utilizó un multímetro HP34401A y un software de captura para medir la corriente requerida por cada sistema de desarrollo durante aproximadamente 16s. Estos valores de corriente se promediaron y multiplicaron por el voltaje de alimentación para encontrar la potencia promedio consumida por los sistemas de desarrollo. Luego, se midió la corriente consumida al ejecutar la aplicación de reconfiguración en ambos sistemas de desarrollo y se obtuvo la diferencia en consumo de potencia con los promedios hallados anteriormente. Esta diferencia se multiplico por el tiempo de reconfiguración para la hallar la energía requerida. La tabla 2 muestra la energía adicional utilizada en el proceso de reconfiguración.

Consumo de energía i.MXL	152 mJoule
Consumo de energía FPGA	503 mJoule
Consumo Total reconfiguración	655 mJoule

Tabla 2. Energía requerida en el proceso de reconfiguración.

7. CONCLUSIONES

En este trabajo se propone una estrategia de reprogramación de una FPGA para una plataforma orientada a procesamiento multimedia, que permite la reprogramación a partir de la descripción del hardware en XML.

De los resultados se puede observar que el efecto del análisis XML en el tiempo de reconfiguración no es muy alto y se convierte en una estrategia promisoria para enviar la información requerida de la reconfiguración.

La estrategia de reconfiguración propuesta en este trabajo permite que un dispositivo móvil adapte el hardware en cualquier momento a aplicaciones que demandan un alto poder computacional y con alto grado de paralelismo, disminuyendo el tiempo de ejecución. Esto puede implicar una disminución del consumo de energía prolongando la vida de las baterías.

Como trabajos futuros se prevé la integración de funciones complejas como módulos de criptografía y procesamiento multimedia para evaluar conjuntamente el proceso programación y la ejecución de las aplicaciones sobre la plataforma.

8. AGRADECIMIENTOS

Los autores agradecen a la Universidad de Antioquia y Colciencias – Colombia por su apoyo en la realización de este trabajo.

9. REFERENCIAS

[1] A. Alsolaim, J Starsyk. “Dynamically Reconfigurable Solution in the Digital Baseband Processing for Future Mobile Radio Devices” *Symposium on System Theory*, 2001

[2] G. Smith, P. Heysters. “Lessons Learned from Designing the MONTIUM: a Coarse-Grained Reconfigurable Processing Tile**”** *Proceedings of the International Symposium on System-on-Chip*, pp 29-32, 2004

[3] G. Kuzmanov, G. Gaydadjiev, S. Vassiliadis. “The Molen Media Processor: Design and Evaluation”. *Proceedings of the International Workshop on Application Specific Processors*, pp. 26-33, 2005.

[4] Consultado en http://www.elixent.com/

[5] A, Upegui, et al. “An FPGA Dynamically Reconfigurable Framework for Modular Robotics”. *ARCS Workshops*, 2005.

[6] D. Rolince. BSML: “An Application of XML to Enhance Boundary Scan Test Data Transportability” *Proceedings on AUTOTESTCON*, pp. 83-91, 2001.

[7] “MC9328MXL i.MX Integrated Portable System Processor Reference Manual”, Freescale Semiconductor, Inc. 2004.

[8] “Spartan-3 FPGA Family: Complete Data Sheet”, Xilinx Inc. www.xilinx.com. 2006.

[9] K. Morris. “Power, suddenly we care”. *FPGA and structured ASIC Journal.* http://www.fpgajournal.com, 2005.

[10] C. W. Tseng. “Sparatan-3 Advanced Configuration Architecture”, Xilinx Inc. www.xilinx.com, pp. 1, 2004.

[11] E. Van Der Vlist. “XML Schema”. O’Reilly, 2002.

[12] T. Parr. “ANTLR Reference Manual”, www.antlr.org. Universidad de San Francisco.

[13] J. Echeverri. “XML Schema para la transformación de Archivos SVF a XML”. http://microe.udea.edu.co/~juan/ARCOM/XMLSchema.2006.

[14] Xerces C++ Parser. *The Apache XML Project*. http://xml.apache.org/xerces-c.

[15] Cogent i.MXL LiteKit. *Cogent Computer Systems Inc*. www.cogent.com. 2006

[16] Spartan-3 Starter Kit Board User Guide. *Xilinx. Inc.* www.digileninc.com. 2005