RIELAC, Vol.XXXII 3/2011 p.35-44 Septiembre - Diciembre ISSN: 1815-5928

RIELAC
Revista de Ingeniería Electrónica, Automática y Comunicaciones

Controlador empotrado en FPGA para Sistema Inteligente de Transporte

Alejandro Cabrera Aldaya
Alejandro J. Cabrera Sarmiento

RESUMEN / *ABSTRACT*

En el presente trabajo se expone la concepción, desarrollo e implementación de un controlador empotrado en un FPGA de Xilinx para ser utilizado en un Sistema Inteligente de Transporte (SIT). La estructura hardware del controlador está basada en la utilización de diversos módulos de propiedad intelectual del sistema de procesamiento MicroBlaze y el soporte de software está basado en la utilización del sistema operativo Petalinux. El controlador empotrado dispone de interfaces Ethernet, USB, UART, SPI e I2C para la comunicación con los diferentes niveles jerárquicos del SIT. Ha sido implementado sobre una placa de desarrollo basada en un FPGA Spartan3E de 1.200 k compuertas, ocupando un 59% de sus recursos configurables. El resto de los recursos disponibles en el FPGA permite, además de la posible actualización del controlador, la implementación hardware de algoritmos que requieren una alta velocidad de procesamiento.

Palabras claves: FPGA, MicroBlaze, Petalinux, Sistema Inteligente de Transporte

FPGA embedded controller for Intelligent Transportation System

This paper exposes the development and implementation of an embedded controller based on a Xilinx FPGA in order to be employed in an Intelligent Transportation System (ITS). The controller`s hardware configuration is based on the use of many intellectual properties modules from MicroBlaze processing system, while software support is based on Petalinux operating system. The embedded controller has Ethernet, USB, UART, SPI and I2C interfaces for communication with the different hierarchical levels of the ITS. The controller has been implemented on a development board based on a Spartan3E 1.200k gates FPGA and it consumes a 59% of the FPGA configurable resources. The rest of the FPGA available resources let, besides the controller actualization, a hardware implementation of algorithm that needs a high speed processing.

Key words: FPGA, MicroBlaze, Petalinux. Intelligent Transportation System

INTRODUCCIÓN

Desde hace algunos años se han estado explorando en el país soluciones para crear una infraestructura eficiente de gestión del transporte que satisfaga tanto las necesidades actuales como las perspectivas de desarrollo. Con este objetivo se ha estado desarrollando, entre varias instituciones, un Sistema Inteligente de Transporte (SIT) que permita gestionar toda la red de tránsito de una ciudad (semáforos, sistemas de identificación automática de vehículos, sistemas de información al viajero, etc.).

En general, los Sistemas Inteligentes de Transporte se organizan de forma jerárquica donde, desde un centro de control se puede monitorear y controlar todo el sistema, interactuando con sus correspondientes controladores locales [1-5]. Para garantizar la estructura jerárquica del SIT compuesta por un Centro de Control como nivel superior y los diferentes Controladores Locales (por ejemplo, controladores semafóricos) en el nivel inferior, debe existir una comunicación entre ambos niveles. Una alternativa sería establecer esta comunicación directamente entre el Centro de Control y cada uno de los Controladores Locales, lo cual aumentaría la complejidad y costo de estos últimos.

Otra alternativa de solución consiste en la inserción de un nivel intermedio en la jerarquía del SIT que sirva como puente de comunicación entre el Centro de Control y los Controladores Locales, tal como se muestra en la Figura 1, Este nivel intermedio lo constituye un elemento denominado Controlador Maestro, el cual constituye el objeto de la presente investigación.

Cómo se puede inferir de la Figura 1, el Controlador Maestro debe incluir múltiples interfaces de comunicaciones para posibilitar el intercambio de información hacia y desde el Centro de Control y los diferentes Controladores Locales (hasta cinco) que puede tener asociado. Así, se ha definido que la comunicación con el Centro de Control se realice de forma cableada a través de Ethernet y de forma inalámbrica mediante redes GSM/GPRS, mientras que la comunicación con los controladores locales se realice de forma serie mediante interfaces UART, I2C y SPI.

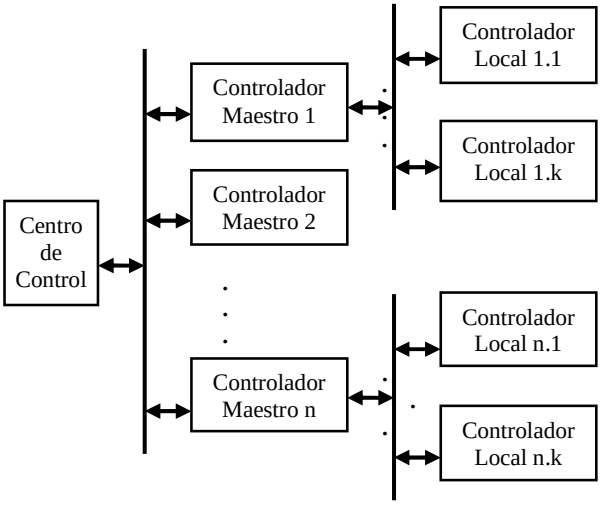

Figura 1. Estructura jerárquica del Sistema Inteligente de Transporte

Sin embargo, el servir de nodo de comunicaciones no es la única tarea del Controlador Maestro, pudiendo, en ocasiones, realizar las tareas de algunos de los Controladores Locales. Por esta razón, los Controladores Maestros deben estar basados en un potente sistema de procesamiento de información, basado en procesadores de 32 bits y con soporte para la ejecución de un sistema operativo [3-5].

Aunque pueden existir múltiples alternativas para el desarrollo del soporte de hardware del Controlador Maestro, dada la gran variedad y cantidad de interfaces de comunicaciones requeridas, así como de otros posibles recursos (temporizadores, controladores de interrupción, etc.), la alternativa de implementarlo sobre hardware reconfigurable ofrece excelentes perspectivas. Por una parte, el vertiginoso desarrollo de la industria microelectrónica ofrece dispositivos de hardware reconfigurable tipo FPGA (Field Programmable Gate Array) con múltiples recursos internos (bloques de memoria RAM, multiplicadores, diferentes tipos de interfaces de entrada/salida, procesadores, etc.) que facilitan la implementación del controlador empotrado. Por otra parte, la disponibilidad de descripciones software de componentes hardware complejas, conocidas como módulos de propiedad intelectual (Intellectual Properties, IP) requeridas para la implementación de los diferentes componentes del sistema de procesamiento, permite la reducción del tiempo de desarrollo de este controlador empotrado, incrementa la fiabilidad del mismo y, además, facilita su continua actualización (por ejemplo, versiones hardware mejoradas de las interfaces de comunicaciones o del propio procesador).

Una ventaja adicional de la utilización de hardware reconfigurable para la implementación del Controlador Maestro empotrado radica en que permite la implementación sobre hardware de algoritmos (o de parte de ellos) que requieren ser acelerados en relación con sus equivalentes realizados por software. Ejemplos de estos pueden ser los algoritmos de reconocimiento de patrones relacionados con la identificación automática de vehículos o los de cifrado/descifrado de la información a intercambiar con el Centro de Control. Esta característica equivale a dotar al sistema de procesamiento del Controlador Maestro de una mayor capacidad de procesamiento, De esta forma, el soporte de hardware reconfigurable facilita la realización híbrida hardware/software del controlador empotrado.

A continuación se expone el desarrollo e implementación del Controlador Maestro empotrado sobre un FPGA. En la sección 2 se exponen lo requerimientos de hardware del Controlador Maestro, mientras que en la sección 3 se muestra el diseño del mismo y los resultados de la implementación del controlador empotrado. En la sección 4 se exponen los elementos fundamentales relacionados con la implementación del sistema operativo Petalinux y de una aplicación básica de operación del Controlador Maestro. Finalmente, se resumen los aspectos más relevantes del trabajo desarrollado.

REQUERIMIENTOS DE HARDWARE DEL CONTROLADOR MAESTRO

Aunque el elemento fundamental del Controlador Maestro y el principal objeto de estudio durante esta investigación es el controlador basado en módulos de propiedad intelectual (procesador, buses, memorias, interfaces, etc.) empotrado en el FPGA, son necesarios un conjunto de elementos externos. En la Figura 2 se muestra la estructura general del Controlador Maestro, formada por el FPGA y diversos dispositivos externos. A continuación se exponen los requerimientos de hardware del Controlador Maestro para, a partir de este análisis, extraer los módulos IP necesarios que debe incorporar el controlador empotrado en el FPGA y proceder a su diseño e implementación en la siguiente sección.

Uno de los componentes fundamentales del Controlador Maestro son las memorias externas al FPGA. Aunque los FPGA actuales disponen internamente de cierta cantidad de bloques de memoria RAM, la capacidad de los mismos, en general no es suficiente para almacenar y permitir la ejecución de un sistema operativo y las diversas aplicaciones que debe ejecutar el sistema de procesamiento empotrado en el FPGA. Es por ello que se hace imprescindible la utilización de memoria RAM externa. La capacidad de la memoria externa necesaria está entre 16 y 64 MBytes, seleccionándose del tipo DDR SDRAM, por lo que el controlador de la misma debe ser uno de los módulos IP a empotrar en el FPGA.

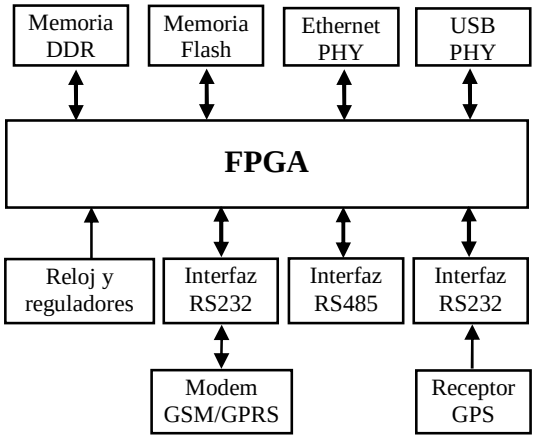

Figura 2. Estructura del Controlador Maestro

Por otra parte, dado que es necesario almacenar en una memoria no volátil la imagen del sistema operativo y de las aplicaciones a ejecutar, se requiere de memoria Flash externa. Dado que la mayoría de los FPGAs almacenan la configuración de sus interconexiones internas en una memoria RAM estática (volátil), la memoria Flash externa se utilizará también para almacenar el fichero de configuración del FPGA. De esta forma, la capacidad de esta memoria debe oscilar entre 4 y 8 MBytes.

Para establecer la comunicación cableada a través de Ethernet con el Centro de Control se requiere de una interfaz física externa al FPGA [6]. Por supuesto, en la configuración del sistema empotrado debe incluirse el controlador empotrado correspondiente.

Para lograr la comunicación inalámbrica con el centro de control el Controlador Maestro necesita interactuar con un modem GSM/GPRS. La interfaz que normalmente se utiliza para interactuar con este tipo de modem es un puerto serie asincrónico, en este caso una interfaz compatible con un UART 16550 que será empotrada en el FPGA [7-8]. El modem GSM/GPRS seleccionado ha sido el WTS GP4040 que utiliza una interfaz física RS232, por lo que se incluye el conversor correspondiente.

Con el objetivo de mantener actualizada la fecha y hora de los diferentes Controladores Locales subordinados al Controlador Maestro, así como de tener una referencia común y fiable entre los diferentes Controladores Maestros del Sistema Inteligente de Transporte, se utiliza un receptor GPS [9-10]. Dado que el receptor GPS seleccionado también se conecta a través de un puerto serie asincrónico, se incluye una interfaz física RS-232 adicional.

También se ha dotado al Controlador Maestro de una interfaz USB esclava para poder establecer comunicación local con una computadora portátil u otro dispositivo móvil que tenga capacidad de procesamiento (como PDA) de forma tal que a través de la misma se puedan realizar tareas de diagnóstico o configuración locales. Es por ello que se requiere el transceptor externo de capa física correspondiente [11].

Se incluye también una interfaz para un bus RS485 debido a que el mismo se requiere en un Sistema Inteligente de Transporte para la comunicación con dispositivos colocados a determinada distancia del Controlador Maestro, como pueden ser los contadores decádicos que muestran el tiempo restante de cada una de las luces o las pantallas para los sistemas de información al viajero.

Además, es necesario disponer de los reguladores de tensión para la alimentación de los diferentes circuitos, en especial del FPGA y de la memoria DDR, así como los circuitos generadores de reloj para estas dos componentes.

Para la comunicación entre el Controlador Maestro y los diferentes Controladores Locales asociados, se utilizan interfaces serie del tipo UART, SPI e I2C. Aunque estas interfaces no requieren componentes externos significativos, deben ser consideradas para el diseño del controlador empotrado en el FPGA que se expone en la siguiente sección.

DISEÑO E IMPLEMENTACIÓN DEL CONTROLADOR EMPOTRADO

A partir de los requerimientos de hardware del Controlador Maestro y de sus diferentes elementos componentes expuestos en la sección anterior, se procede al diseño e implementación del controlador empotrado en el FPGA, el cual estará basado en muy diversos módulos de propiedad intelectual.

El primer paso ha sido la selección del FPGA, considerando que disponga de recursos suficientes no solo para incluir los módulos IP requeridos sino también para permitir la implementación hardware, parcial o total, de determinados algoritmos que deban ser acelerados. Además, para no encarecer el Controlador Maestro, después de diferentes pruebas de implementación, se ha seleccionado un FPGA de la familia de gama media Spartan3E de Xilinx, con una capacidad de al menos un millón doscientas mil compuertas equivalentes [12].

La Figura 3 muestra la estructura general del controlador empotrado en el FPGA, en donde se observan los diversos módulos de propiedad intelectual que se requieren. Dado que esta familia de FPGA no dispone de procesadores empotrados (hardcore) es necesario utilizar los módulos IP del procesador RISC de 32 bits MicroBlaze, además de los correspondientes controladores de buses para la memoria local, la memoria cache (ambas basadas en bloques de memoria RAM) y el bus de expansión [13].

La utilización de memoria cache es importante dado que el sistema operativo y la mayor parte de las aplicaciones serán ejecutados desde la memoria externa, por lo que la utilización de la memoria cache permite reducir la latencia en el acceso a la información que requiere el procesador, incrementando así su rendimiento. Dada la arquitectura Harvard de este procesador, es muy útil la implementación de caches independientes de códigos y de datos, seleccionándose capacidades de 4 kbytes para cada una.

Un aspecto a considerar es el tipo de bus de expansión a utilizar. Xilinx proporciona dos tipos de buses para sus procesadores empotrados: OPB (On-chip Peripheral Bus) y PLB (Processor Local Bus). El bus PLB se caracteriza por requerir menos ciclos de reloj para una transferencia además de ser el estándar actual de este fabricante [14-15]. Sin embargo, el hecho de que no todos los módulos IP requeridos se encontraban disponibles de forma gratuita para este bus en el momento de esta investigación y la limitación a la conexión de hasta 16 módulos, hace necesario la utilización de ambos buses, con un puente de conexión entre el bus PLB y el bus OPB.

Los módulos IP de los controladores de memoria DDR SDRAM y Flash se requieren para el control de las respectivas memorias. Para el caso de la memoria dinámica Xilinx proporciona el controlador MPMC (Multi Port Memory Controller), diseñado para el bus PLB el cual es capaz de manejar memorias SDRAM, DDR y DDR2 [16]. Para la memoria Flash se utiliza el controlador simple de memoria externa EMC (External Memory Controller), disponible para el bus OPB [17].

Los controladores de buses I2C y SPI se requieren para implementar estas interfaces de comunicaciones con los controladores locales. Xilinx proporciona módulos IP de ambos buses disponibles para el bus PLB [18-19]. En el caso de SPI se requieren cinco salidas de selección de dispositivos correspondientes a cada uno de los cinco controladores locales posibles.

También se requieren cinco interfaces serie UART para la comunicación con los diferentes controladores locales, además de otras dos para la comunicación con el módulo GPS externo y para el bus RS485. Dado que estas interfaces UART no requieren ser programables ni requieren señales de intercambio con un MODEM, se utilizan módulos IP de interfaces UART simples (UartLite) las cuáles consumen pocos recursos en comparación con otras más complejas [20]. Sin embargo, dado que la comunicación con el módulo GSM/GPRS requiere de una interfaz serie compartible con el UART 16550, se hace necesaria la utilización de un módulo IP del mismo, disponible para el bus PLB [21].

Los puertos de entrada/salida (E/S) se necesitan para la indicación del estado del Controlador Maestro y de cada uno de los Controladores Locales asociados mediante LEDs, así como la introducción de condiciones de depuración mediante microinterruptores. Para ello se hace uso del módulo IP OPB-GPIO (General Purpose Input/Output) disponible para el bus OPB [22].

También se requieren cinco interfaces serie UART para la comunicación con los diferentes controladores locales, además de otras dos para la comunicación con el módulo GPS externo y para el bus RS485. Dado que estas interfaces UART no requieren ser programables ni requieren señales de intercambio con un MODEM, se utilizan módulos IP de interfaces UART simples (UartLite) las cuáles consumen pocos recursos en comparación con otras más complejas [20]. Sin embargo, dado que la comunicación con el módulo GSM/GPRS requiere de una interfaz serie compartible con el UART 16550, se hace necesaria la utilización de un módulo IP del mismo, disponible para el bus PLB [21].

Los puertos de entrada/salida (E/S) se necesitan para la indicación del estado del Controlador Maestro y de cada uno de los Controladores Locales asociados mediante LEDs, así como la introducción de condiciones de depuración mediante microinterruptores. Para ello se hace uso del módulo IP OPB-GPIO (General Purpose Input/Output) disponible para el bus OPB [22].

Uno de los módulos IP más complejos que presenta el diseño del Controlador Maestro es el encargado de las funcionalidades de la capa de enlace de Ethernet (Ethernet MAC). Ante la aparición de descripciones gratuitas de estos módulos (aunque con pobre documentación), Xilinx ha incorporado un módulo Ethernet MAC con algunas limitaciones, pero con abundante documentación y compatible con sus procesadores empotrados [23-24]. Este módulo (XPS EthernetLite) soporta velocidades de 10 y 100 Mbps aunque, debido a que no permite la transferencia de datos a través de acceso directo a memoria (DMA), esta velocidad se ve afectada pudiendo caer por debajo de los 20 Mbps en la capa de aplicación [23]. Después de numerosas pruebas de evaluación de diferentes módulos, se ha seleccionado el módulo IP brindado por Xilinx al menos para las pruebas de prototipo del Controlador Maestro, pudiendo ser fácilmente reemplazado por otro de mejores prestaciones al realizarse la implementación sobre hardware reconfigurable.

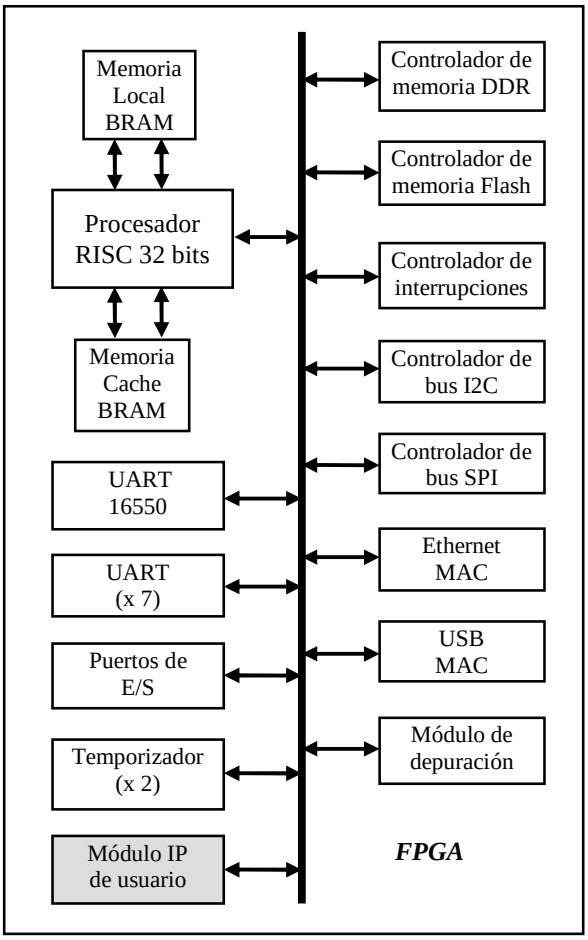

Figura 3. Estructura del controlador empotrado

Dado que la interfaz Ethernet y el sistema operativo requieren de temporizadores para su operación, es necesario incluir los correspondientes módulos IP. En este caso se hace uso del módulo IP OPB-Timer/Counter el cual incorpora dos temporizadores programables y configurables de hasta 32 bits [25].

Teniendo en cuenta que muchos de los módulos serán atendidos por interrupción, se requiere de un controlador de interrupciones. En esta investigación de ha utilizado el módulo OPB Interrupt Controller, capaz de manejar hasta 32 solicitudes de interrupción con prioridades [26].

Para facilitar la descarga y depuración de las aplicaciones que se ejecutarán por MicroBlaze, se ha incluido un módulo de depuración hardware MDM (Microprocessor Debugger Module) que facilita estas tareas [27].

Mientras todos los módulos IP expuestos hasta aquí están disponibles en el entorno de desarrollo EDK (Embedded Development Kit) de sistemas empotrados de Xilinx, para la interfaz USB sólo brinda un módulo IP de evaluación por lo que, de utilizarse, habría que adquirirlo a un costo adicional al de la licencia de EDK.

Una alternativa al uso del módulo IP de Xilinx es utilizar uno de libre distribución (código abierto) el cual es desarrollado y mantenido por los integrantes del proyecto OpenCores [28]. Este módulo solamente es compatible con la versión 1.1 de USB. La velocidad soportada por este estándar (12 Mbps) es suficiente para la aplicación del Controlador Maestro, sin embargo, la principal desventaja de esta variante radica en la pobre documentación disponible de este módulo IP además de no tener implementado el protocolo software necesario para la comunicación a través de USB.

Debido a esto, para la comunicación vía USB se ha decidido, por el momento, utilizar un dispositivo externo conversor de UART a USB [29]. De esta forma el único hardware necesario dentro del FPGA sería un módulo IP UartLite adicional para establecer comunicación con el convertidor externo a USB. Así, el módulo denominado USB MAC de la figura 3 será reemplazado en el controlador empotrado por un módulo IP UartLite. Esta solución es solamente temporal, debido, por una parte, a que la evaluación de los módulos IP de interfaces USB de OpenCores se realizará en futuras investigaciones y, por otra, que no se descarta que Xilinx libere el módulo IP del controlador USB en un futuro próximo.

En la Figura 3 se muestra un módulo de particular importancia, denominado módulo IP de usuario (resaltado en gris). Con este tipo de módulo se representa la implementación hardware de cualquier aplicación que requiera interacción con el sistema de procesamiento, como pueden ser los módulos hardware de aceleración de algoritmos. El entorno de desarrollo EDK proporciona plantillas que facilitan la interconexión del hardware específico de usuario a los diferentes buses del sistema de procesamiento, convirtiéndolo así en un módulo IP adicional el cual puede ser reutilizado cuantas veces se desee [30].

Durante el desarrollo de la investigación fueron evaluados e implementados cada uno de los módulos IP descritos. Las pruebas de evaluación se realizaron sobre una placa de desarrollo de FPGA basada en un Spartan3E de 1.200 k compuertas equivalentes, con 64 MBytes de memoria DDR SDRAM y 16 Mbytes de memoria Flash, además de disponer de dos interfaces RS232, una interfaz Ethernet PHY, diferentes dispositivos conectados a un bus SPI y conectores de expansión. Los resultados de implementación del controlador empotrado en el FPGA se muestran en la Tabla 1.

Tabla I: Resultados de implementación

Recurso	*%*
Total Number Slice Registers: 6,183 out of 17,344	35
Number of occupied Slices: 5,116 out of 8,672	59
Number of bonded IOBs: 117 out of 250	46
Number of RAMB16s: 18 out of 28	64
Number of DCMs: 2 out of 8	25
Number of BSCANs: 1 out of 1	100
Number of MULT18x18: 3 out of 28	10

Nótese que la implementación del controlador empotrado ocupa un 59 % de los slices, un 64 % de los bloques de memoria RAM y apenas un 10 % de los multiplicadores disponibles en el FPGA, con lo cual quedan recursos suficientes para la implementación hardware de algoritmos (o partes de ellos) que necesiten ser acelerados.

SOPORTE DE SISTEMA OPERATIVO Y APLICACIÓN BÁSICA

La primera versión del Controlador Maestro se desarrolló con una aplicación sin sistema operativo (stand alone) en donde fue preciso hacer un uso intensivo de los drivers de alto nivel disponibles para los diferentes módulos IP, los cuales fue necesario analizar previamente [31]. De complejidad particular resultó el trabajo con la interfaz Ethernet dado que la misma requiere de la utilización de su propio protocolo software, conocido como stack [32-33].

Con el objetivo de liberar al programador de las aplicaciones del Controlador Maestro de los detalles del hardware empotrado, se procedió a la utilización de un soporte de sistema operativo del cual se disponga el código fuente. Una de las principales ventajas de utilizar un sistema operativo basado en Linux en sistemas empotrados en FPGAs se debe a la diversidad de configuraciones hardware de los controladores diseñados, razón por la cual se hace necesaria la modificación del código fuente del núcleo (kernel) del sistema operativo para poder adaptarlo al diseño que se realice [34].

La utilización de un sistema operativo como plataforma base del software del controlador maestro brinda la posibilidad de añadir nuevos dispositivos, funcionalidad muy utilizada en los sistemas basados en FPGAs, de una manera simple y sin la necesidad de hacer grandes cambios en el software desarrollado y depurado. El manejo de los stacks de interfaces como Ethernet o USB desde un sistema operativo resulta mucho más sencillo y eficiente debido a que es el sistema operativo el que organiza y distribuye eficientemente cada una de las tareas que debe ejecutar el Controlador Maestro.

De los posibles sistemas operativos disponibles para entornos empotrados se seleccionó Petalinux, una distribución basada en uClinux para procesadores que no dispongan de unidad de manejo de memoria (MMU, Memory Management Unit), como es el caso del procesador MicroBlaze [35].

Petalinux ofrece un conjunto de ficheros de configuración (*scripts*) desarrollados para facilitar la configuración del *kernel* y adaptarla a los diferentes módulos IP del sistema empotrado. Además, incorpora drivers para los módulos IP más comunes, tales como UART, I2C, SPI, temporizadores, Ethernet, etc., lo cual facilita el trabajo con los mismos.

Así, después de analizar a profundidad el sistema operativo Petalinux y sus drivers, se procedió al desarrollo de una aplicación básica para el Controlador Maestro [36]. Esta aplicación consistió en el establecimiento de una comunicación TCP/IP a través de Ethernet entre un Centro de Control (simulado en una computadora personal, PC) y el Controlador Maestro, así como la correspondiente comunicación de éste, a través de una interfaz serie UART, con un Controlador Local, en este caso un controlador semafórico, haciendo uso de protocolos de comunicación previamente definidos en el Sistema Inteligente de Transporte.

La Figura 4 muestra una pantalla de la aplicación de prueba desarrollada en Builder C++ que se ejecuta en un PC simulando el Centro de Control.

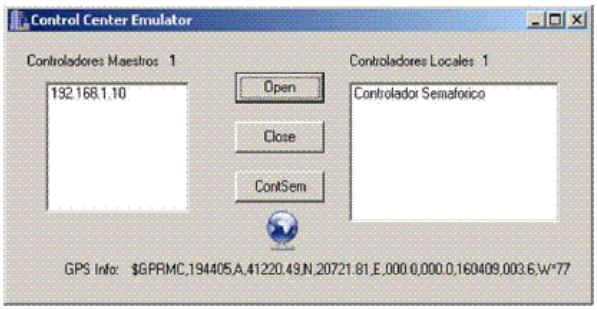

Figura 4. Aplicación de prueba del Centro de Control

En la lista de la izquierda de la aplicación se muestra la dirección IP del Controlador Maestro que ha establecido conexión con el Centro de Control, mientras que en la lista de de la derecha se muestran los Controladores Locales que ha reportado el Controlador Maestro que tiene asociados. Una vez establecida la conexión se muestra en la parte inferior la trama del receptor GPS enviada por el Controlador Maestro al Centro de Control.

El comando que ha sido implementado en esta aplicación corresponde al del cambio de la configuración de las luces de una intersección semaforizada. Una vez que el Controlador Maestro recibe el comando desde el Centro de Control, debe buscar a cual interfaz serie está conectado el Controlador Local al cual el Centro de Control le envía los datos. Esta búsqueda se hace a través de una estructura de datos creada durante la enumeración de todos los Controladores Locales físicamente conectados. Después que la aplicación básica en el Centro de Control reciba el resumen de los controladores locales, esta los reconoce y los muestra en la lista de la derecha. Con el botón ContSem se procede al envío de una configuración de luces para el controlador semafórico.

Los resultados de la configuración de un kernel operativo de Petalinux resultaron en la ocupación de algo más de 2 MByte de memoria, mientras que la aplicación de prueba desarrollada ocupaba unos pocos kbytes de memoria, con lo cual el Controlador Maestro diseñado posee recursos de memoria suficientes para soportar el sistema operativo así como muy diversas aplicaciones.

CONCLUSIONES

En el trabajo se ha expuesto el desarrollo de un controlador empotrado sobre hardware reconfigurable basado en la utilización de módulos de propiedad intelectual, para ser utilizado como parte de un Controlador Maestro en un Sistema Inteligente de Transporte.

La utilización de un soporte de hardware reconfigurable para el desarrollo de este controlador permite múltiples ventajas, entre ellas las de facilitar la actualización del hardware del controlador y de permitir la implementación hardware de algoritmos (o de partes de ellos) cuya ejecución deba ser acelerada.
Todos los módulos IP utilizados en la investigación están disponibles en el entorno de desarrollo de sistemas empotrados de Xilinx y fueron implementados y verificados en el transcurso de esta investigación.

Los resultados de implementación del controlador empotrado sobre un FPGA Spartan3E de 1.200k compuertas equivalentes muestran un 59 % de ocupación de los slices, una utilización del 64 % de los bloques de memoria RAM y apenas un 10 % de los multiplicadores utilizados.

Aunque se realizó una primera versión independiente (stand alone) de la aplicación principal del Controlador Maestro, este procedimiento no se recomienda por la complejidad que implica, sobre todo el relacionado con los stacks de algunas interfaces como USB y Ethernet.

Se ha analizado, configurado, implementado y evaluado una versión del sistema operativo Petalinux, la cual consume algo más de2 MBytes de memoria. La utilización del sistema operativo facilita en gran medida el desarrollo de las aplicaciones que debe ejecutar el controlador empotrado.

RECONOCIMIENTOS

Los autores desean expresar su agradecimiento al Complejo de Investigaciones Tecnológicas Integradas (CITI) por todo el apoyo recibido para la realización del presente trabajo.

REFERENCIAS

1. US Department of Transportation, National Intelligent Transportation System Architecture Summary, USA, 2007

2. Advanced Transportation Controller (ATC) Standard, USA, 2006

3. ATC 2070 Advanced Transportation Controller, USA, 2008

4. SITRAFFIC C900V Specifications, Siemens, 2006

5. MC68360 Quad Integrated Communications Controller User's Manual, Freescale Semiconductor, Inc., USA, 2003.

6. LAN83C180 10/100 Fast Ethernet PHY Transceiver Data Sheet, SMSC Standard Microsystems Corporation, 2007

7. Digital cellular telecommunications system: AT command set for GSM Mobile Equipment, European Telecommunications Standards Institute (ETSI), 2007

8. WTS-GP4040 Terminal GSM/GPRS User's Manual, ISE, 2007

9. Betke, K., The NMEA 0183 Protocol, 2001

10. GPS 16x Technical Specifications, Garmin International, Inc., 2008

11. STUSB03E Universal Serial Bus transceiver, ST-Microelectronics, 2007

12. Spartan-3E FPGA Family Functional Description, Xilinx, Inc., 2006

13. MicroBlaze Processor Reference Guide, Xilinx Inc., 2009

14. DS401 On-Chip Peripheral Bus V2.0 with OPB Arbiter (v1.10c), Xilinx, Inc., 2006

15. DS531 Processor Local Bus (PLB) v4.6 (v1.03a), Data Sheet, Xilinx, Inc., 2008

16. DS643 Multi-Port Memory Controller (MPMC), Xilinx, Inc., 2008

17. OPB External Memory Controller (EMC), Xilinx, Inc., 2006

18. Xilinx DS606 XPS IIC Bus Interface, (v2.00a), Data Sheet, Xilinx, Inc., 2008

19. Xilinx DS570 XPS Serial Peripheral Interface (SPI) (v2.00b), Data Sheet, Xilinx, Inc, 2008

20. Xilinx DS571 XPS UART Lite (v1.00a), Data Sheet, Xilinx, Inc., 2008

21. Xilinx DS577 XPS 16550 UART (v1.00a), Data Sheet, Xilinx, Inc., 2008

22. DS466 OPB General Purpose Input/Output (GPIO) (v3.01b), Xilinx, Inc., 2006

23. DS580 XPS Ethernet Lite Media Access Controller (v2.00b), Xilinx, Inc., 2008

24. Gao, Jon. 10_100_1000 Mbps Tri Mode Ethernet MAC Specifications, OpenCores.org, 2009

25. DS465 OPB Timer/Counter, Xilinx, Inc., 2006

26. DS473 OPB Interrup Controller, Xilinx, Inc., 2006

27. DS450 Microprocessor Debugger Module, Xilinx, Inc., 2008

28. Fieldings, S., USBHostSlave IP Core Specification, OpenCores.org, 2008

29. FT232BM USB UART (USB - Serial) Datasheet, Future Technology Devices Intl., 2002

30. Embedded System Tools Reference Manual, Xilinx, Inc, 2008

31. Cabrera, A., Controlador maestro basado en FPGA para un SIT, Trabajo de Diploma, CUJAE, Jul. 2009

32. Xilinx lwIP 1.3.0 Library (v1.00.1), Xilinx, Inc., 2008

33. Dunkels, A., Design and Implementation of the lwIP TCP/IP Stack, Swedish Institute of Computer Science, 2001

34. Raghavan, P., Lad, A., Neelakandan, S., Embedded Linux system design and development, CRC Press, 2005

35. Petalinux Kernel Description and Implementation, Petalogic Inc., 208

36. Rodríguez, J., Evaluación y desarrollo de aplicaciones para el sistema operativo Petalinux, Trabajo de Diploma, CUJAE, Jul. 2010

Autores

Alejandro Cabrera Aldaya, Ingeniero en Automática, Complejo de Investigaciones Tecnológicas Integradas (CITI), calle 114 Nº 11901, CUJAE, Marianao, La Habana,

e-mail: acabrera@udio.cujae.edu.cu

Alejandro José Cabrera Sarmiento, Ingeniero Electricista, Doctor en Ciencias Técnicas, Profesor Titular, Departamento de Automática y Computación del Instituto Superior Politécnico “José Antonio Echeverría”, calle 114 Nº 11901, CUJAE, Marianao, La Habana,

e-mail: alex@electrica.cujae.edu.cu.