Investigación

Implementación de sistemas digitales complejos utilizando sistemas embebidos

Complex DIGITAL Design Systems Using Embedded Systems

Recibido: enero de 2012
Arbitrado: abril de 2012

Carlos Camargo B.*, John Cortés R.** y Alexander Jiménez T.***

Resumen

Con el pasar del tiempo los sistemas digitales van creciendo en complejidad, así como las exigencias de desempeño y funcionamiento. La incursión de los sistemas digitales en aplicaciones vitales, en las cuales un error resultaría en la pérdida de vidas humanas, exige que estos sistemas sean muy confiables (tolerantes a fallos) y presenten un desempeño muy alto. El desarrollo de procesadores de 32, 64 bits de bajo costo unido a la disponibilidad de herramientas software como librerías, drivers y sistemas operativos de tiempo real, hace que la realización de este tipo de sistemas no sea una tarea imposible.

Palabras clave

Sistemas embebidos, diseño de sistemas digitales.

Abstract

Digital systems grow in complexity, requirements and performance. Use of digital systems in critical applications, where an error would result in the loss of human life demands faults tolerance and very high performance. Development tools like 32, 64-bit processors coupled with the availability of open software tools (libraries, drivers, and real-time operating systems) reduce thecomplexity in the implementation of such systems. This article provides the first steps in the study of embedded systems and tries to show it potential. Also addresses the steps to be carried out in the design of embedded system and is made an explanation of the tools used.

Keywords

Embedded Systems, digital design.

I. Introducción

En algunos países en vía de desarrollo, aún se utilizan procesadores de 8 o 12 bits y el lenguaje ensamblador como plataforma de desarrollo para un amplio número de aplicaciones. Debido a las limitaciones de esta plataforma (velocidad, ancho del bus de datos, periféricos, número

* Doctor en Ingeniería Eléctrica de la Universidad Nacional. Profesor del Dpto. de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia e-mail: cicamargoba@unal.edu.co.

** Doctor en Ingeniería Eléctrica CINVESTAV México. Profesor del Dpto. de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia. e-mail: jacortesr@unal.edu.co.

*** Profesor de la Facultad Tecnológica de la Universidad Distrital Francisco José de Caldas e-mail: ajimenezt2@ udistrital.edu.co.

de pines, herramientas de desarrollo de alto nivel, librerías), no es posible realizar aplicaciones complejas que requieran procesamiento con características de tiempo real o multitarea. Además, el uso de metodologías de diseño anticuadas crea una dependencia tecnológica y hace que el tiempo de desarrollo dependa fuertemente de la experiencia del diseñador.

El continuo avance de los sistemas software y de los componentes hardware presentan nuevas oportunidades para la implementación de sistemas embebidos en múltiples campos tales como: control, multimedia, aplicaciones de red e instrumentación biomédica. En la actualidad se dispone de una gran variedad de dispositivos HW (FPGAs, CPLDs, IPs, ICs) de bajo costo, esto unido al desarrollo de sistemas operativos de tiempo real (como Linux [1], eCos [2], etc) de libre distribución, librerías que permiten el control de dispositivos Hardware como LCDs, Interfaces USB, RS232, de red, librerías matemáticas, científicas, etc. y el desarrollo de compiladores que permiten generar el contenido de la memoria de programa de un gran número de procesadores a partir de un lenguaje de alto nivel como el C, C++ o Java, permite el diseño de sistemas complejos, reduciendo el costo del diseño, el tiempo de desarrollo y en algunos casos la dependencia tecnológica.

Este artículo suministra los primeros pasos en el estudio de los sistemas embebidos y su intención es mostrar la potencialidad de estos. La discusión se divide en tres partes: la primera es un resumido marco teórico donde se enumeran las características de los sistemas embebidos y las tareas que deben ser llevadas a cabo en su diseño, la segunda parte hace una descripción de las herramientas hardware y software necesarias para la implementación de sistemas embebidos, y por último se muestran algunas aplicaciones realizadas en la Universidad Nacional de Colombia.

II. Sistemas Embebidos

Un sistema embebido (SE) es una máquina computacional que emplea una combinación de Hardware y Software para realizar una función específica. Es parte de un sistema más grande que puede no ser un computador y trabaja en un ambiente reactivo con restricciones temporales. En este tipo de sistemas el software se utiliza para proporcionar flexibilidad y funcionalidad. El hardware (procesadores, ASICs, memorias...) es utilizado para proporcionar desempeño y en algunas ocasiones seguridad.

A. Características:

1. Realiza una función única o un pequeño grupo de funciones relacionadas, no es un sistema de propósito general.
2. De alto desempeño y con fuertes restricciones temporales
3. El costo, consumo de potencia y la confiabilidad son atributos que afectan el diseño.
4. Sistemas reactivos con características de tiempo real.

B. Principales tareas a desarrollar en el diseño de sistemas embebidos:

A continuación se enumeran las tareas que deben ser realizadas en el diseño de un sistema embebido, (adicionalmente se deben seguir los pasos de la metodología de diseño de sistemas digitales).

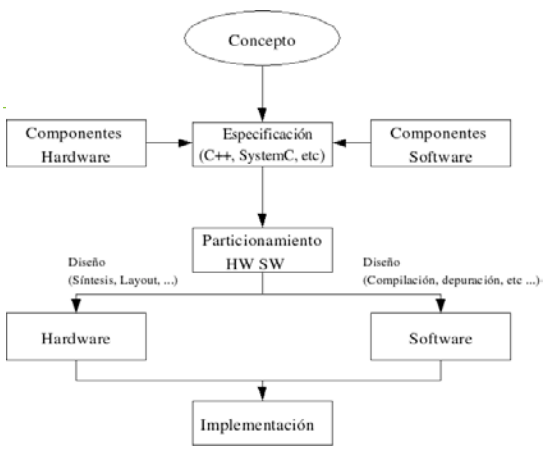

Fig. 1. Flujo de diseño de los sistemas embebidos.

A. Especificación y modelamiento

Descripción algorítmica del sistema. Por lo general estos algoritmos son valida-

dos en una descripción ejecutable, es decir, un programa escrito en un lenguaje de alto nivel[1] que modela el comportamiento del sistema, en el cual se pueden validar de forma fácil y rápida dichos algoritmos.

B. Particionamiento

División del algoritmo en componentes funcionales más pequeñas.

C. Particionamiento HW-SW

Implementación de las unidades funcionales del punto anterior en SW (Código ejecutándose en un procesador) o en HW (Implementado en un PLD). El particionamiento ideal es aquel en el que todas las funciones son implementadas en SW, (esto debido a que la implementación fue validada utilizando un lenguaje de alto nivel, el cual puede ser el mismo lenguaje utilizado por el compilador), pero debido a que los procesadores ejecutan instrucciones de forma secuencial, si la ejecución de una función compromete el cumplimiento de las restricciones temporales (es decir, la función emplea muchos ciclos de máquina) esta debe ser implementada en HW con el fin de aprovechar sus características concurrentes (de aquí la necesidad de las funciones HW en sistemas de alto desempeño). La partición óptima es aquella que cumple con las restricciones del diseño a costo mínimo[2].

Una vez se ha generado una partición óptima se debe validar la solución, este paso es algo complejo ya que involucra la simulación de componentes SW y HW. En la actualidad existen herramientas especializadas que permiten realizar esta cosimulación como: Ptolemy [3], SID [4], Seamless [5], etc.

- **Planificación.** En este paso se decide el tiempo en el que se ejecuta una determinada función. Esto es importante cuando se comparten recursos hardware. Los sistemas operativos de tiempo real (RTOS) proporcionan mecanismos que permiten administrar los recursos de forma eficiente.
- **Implementación.** En este paso se baja en el nivel de abstracción y se llevan las funciones a software que corre dentro de un procesador y a una combinación de hardware custom, semicustom o IP dentro de un PLD.

III. Requerimientos HW y SW

Debido a que los sistemas embebidos son una mezcla de componentes SW y HW es necesario contar con herramientas especializadas para cada caso. Uno de los objetivos de este artículo es mostrar las herramientas gratuitas que pueden ser utilizadas para el diseño de un SE. Por lo tanto no se hablará de las herramientas comerciales.

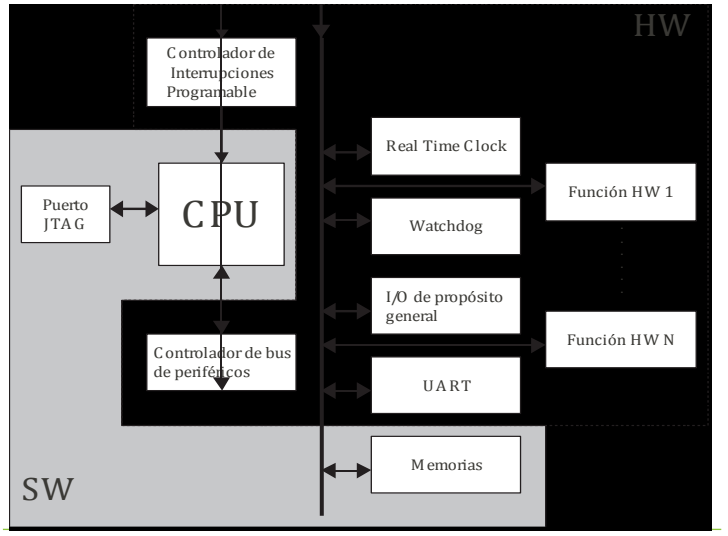

Fig. 2 . Componentes de un SE típico.

3.1 Requerimientos HW

- **Procesador.** Las tareas SW se ejecutan sobre procesadores. Sus características como ancho de los buses de datos y direcciones, arquitectura, velocidad varían

1 Ejemplos de este tipo de lenguajes son el C++, UML, System C, Handel C.

2 El costo SW se traduce en el número de ciclos de reloj que tarda una determinada función en ejecutarse, el Costo HW indica el área dentro del IC ocupada por la misma función.

dependiendo de la aplicación. Sin embargo, es indispensable que tenga la capacidad de realizar operaciones JTAG que permitan realizar funciones de depuración y proporcione una ruta de acceso a los periféricos.

- **Memorias.** Se debe disponer de dos tipos de memorias:

Memoria no volátil, que permita el almacenamiento de los programas, datos constantes y además suministre una forma de iniciar el sistema. En la actualidad se utilizan memorias flash como medio de almacenamiento debido a su bajo costo y gran capacidad de almacenamiento, además estas memorias incorporan circuitos de programación lo cual elimina el uso de programadores especiales y la necesidad de niveles de voltaje elevados, estas memorias presentan como inconveniente la baja velocidad de lectura y escritura, sin embargo, esta desventaja puede ser eliminada haciendo un arranque tipo ROM-RAM, este arranque es similar al de los computadores de escritorio, los programas residen en una unidad de almacenamiento, y en tiempo de ejecución son copiados a la memoria RAM desde donde son ejecutados.

Memoria volátil: Utilizada para almacenar las variables y estructuras de datos necesarios para la ejecución de los programas.

- **Periféricos.** Algunos sistemas operativos requieren ciertos periféricos para su funcionamiento como por ejemplo: RTC (reloj de tiempo real), PIC (controlador de interrupciones programable) y un puerto serie como herramienta de diagnóstico. En la actualidad, los procesadores incluyen ciertos periféricos que ayudan a la implementación de las aplicaciones más comunes, como por ejemplo:

  a. Controladores de LCDs.
  b. Tarjetas de red
  c. Timers
  d. USARTs
  e. Codificadores de audio
  f. Controlador de touch screen
  g. Puertos I2C, SPI, IR.

Además de los periféricos que requiere el sistema operativo se encuentran los que requiere la aplicación.

Existen tres opciones para la implementación de un SE:

1. Utilizando un procesador con puerto JTAG, una FPGA y memorias externas.
2. Utilizando un SoC [6], [7] el cual posee una gran variedad de periféricos y memorias de los dos tipos en el mismo chip. Las funciones HW propias del sistema deben ser implementadas en una FPGA o en un CPLD.
3. En la actualidad, las empresas desarrolladoras de PLDs como Xilinx [8] y Altera [9] proporcionan otra alternativa: En lugar de utilizar un procesador y una FPGA que implementa las funciones HW, se implementa el procesador dentro de la FPGA, esto reduce el tamaño de la tarjeta y aumenta el desempeño y flexibilidad del sistema. Las memorias deben ser externas.

3.2 Requerimientos SW

Comercialmente existen muchos procesadores y FPGAs que cumplen con los requerimientos anteriores, sin embargo, es indispensable contar con las herramientas adecuadas para poder realizar diseños sobre el procesador elegido. A continuación se listan las herramientas necesarias para generar el contenido de la memoria de programa.

- **Compilador:** aunque el lenguaje ensamblador ayuda a optimizar el tiempo de ejecución de funciones críticas, no es recomendable utilizarlo como herramienta de programación, debido a que el tiempo requerido para implementar un diseño complejo puede ser muy largo (dependiendo de la experiencia

del programador), además el lenguaje ensamblador crea una dependencia tecnológica ya que está unido a un determinado procesador.

La utilización de lenguajes de alto nivel como el C++ o el UML, reduce el tiempo de programación, facilita el reuso y la portabilidad del código. Al utilizar un lenguaje estándar como el C o el C++ como herramienta de programación se puede reciclar el código en diferentes proyectos que involucren diferentes procesadores, lo único con lo que se debe contar es con un compilador que permita generar el contenido de la memoria de programa para las diferentes plataformas.

El compilador GCC [8](GNU Project C and C++ compiler) de uso gratuito ha sido utilizado para el desarrollo de un gran número de aplicaciones comerciales y ha demostrado ser muy confiable. Por lo tanto uno de los puntos que deben ser tomados en cuenta a la hora de elegir el procesador es seleccionar uno de los que se encuentran soportados por gcc, en la actualidad gcc soporta los siguientes procesadores: ALPHA, ARM, MIPS, M68K, SPARC, PPC, DSP16XX, H8300, INTEL, MMIX, BLAZE y otros.

GCC genera un archivo en formato ELF (Ejecutable and Linking Format, este formato está dividido en las siguientes secciones (solo se muestran las más importantes):

→ .bss	Datos sin inicializar.
→ .data .data1	Datos inicializados.
→ .debug	Información para depuración simbólica.
→ .dynamic	Información para enlace dinámico.
→ .dynstr	Strings necesarios para enlace dinámico.
→ .rodata. rodata1	Datos de tipo solo lectura.
→ .text	Instrucciones ejecutables.

- **Binutils:** la mayoría de esta información no es necesaria para la ejecución de un programa, solo las secciones .data .rodata y .text, para «extraer» estas secciones del ejecutable se utilizan una serie de herramientas (binutils [9]) que hacen parte de la cadena de herramientas GNU.
- **Depurador:** el GDB [10] (GNU Debugger) permite depurar el código ejecutando instrucción por instrucción y mostrando el valor que toman los registros y las variables internas. Una ventaja muy importante de GDB es que permite hacer depuración dentro del chip, es decir ejecuta las instrucciones dentro del procesador, lee el valor de los registros y variables y retorna esta información, esto es importante ya que no simula el comportamiento del procesador sino que la información obtenida es real. GDB puede comunicarse con los procesadores de diversas formas siendo las más populares el puerto serie y el JTAG.

Para poder realizar este tipo de depuración, GDB permite descargar el programa a la memoria RAM del procesador. Otra herramienta que puede ser utilizada para esto es ARMTOOL [11]

- **Programación de la memoria no volátil:** la distribución oficial de GDB no puede programar las memorias no volátiles, sin embargo, proporciona un canal por el cual se puede realizar dicha programación. Se puede cargar en RAM un programa que programe esta memoria.
- **JTAGER[2],** es una herramienta que proporciona un puente entre la interfaz JTAG del procesador y el PC, permitiendo programar estas memorias vía JTAG.
- **Herramientas de síntesis:** como herramienta de síntesis puede utilizarse cualquier herramienta comercial, las cuales se encuentran en versiones gratuitas.
- **RTOS:** el sistema operativo es muy importante dentro del diseño de un SE, ya que él debe proporcionar las siguientes funciones:

- → Secuencia de arranque, parada y re-arranque
- → Control de interrupciones
- → Control de excepciones
- → Control de temporizadores
- → Programador de tareas
- → Elementos de sincronización
- → Mecanismos de arbitraje de recursos
- → Gestión de la memoria
- → Trazado de tareas y servicios
- → Manejo de dispositivos hardware
- → Herramientas de autodiagnóstico

IV. Aplicaciones desarrolladas en la Universidad Nacional de Colombia

4.1 Plataforma de desarrollo basada en el GBA

El obstáculo más grande a sortear cuando se quieren hacer diseños utilizando sistemas embebidos es el alto costo de las herramientas de desarrollo. Al realizar una búsqueda de plataformas disponibles comercialmente se encuentra al GAMEBOY ADVANCE (GBA) de Nintendo, este «Juguete» posee las siguientes características:

Procesador: ARM7TDMI 32bit RISC CPU, 16.78 MHz.
Memoria Interna: BIOS ROM 16 KBytes Work RAM 288 KBytes (32K in-chip + 256K on-board) VRAM 96 KBytes Palette RAM 1 KByte (256 BG colors, 256 OBJ colors)
Display: 240x160 pixels (TFT color LCD display)
Sonido Análogo 4 channel CGB compatible Digital 2 DMA sound channels Output Built-in speaker, or stereo headphones
Gamepad 4 Direction Keys, 6 Buttons
Puerto serie: varios modos de transferencia (Asíncrono, síncrono)
Memoria externa: 32MB ROM o flash ROM + max 64K SRAM

Como puede verse, esta plataforma cuenta con todas las funcionalidades de las plataformas comerciales y su principal ventaja es el bajo costo (USD 30). Como herramienta de enseñanza el GBA es muy útil a la hora de mostrar conceptos relacionados con los RTOS (Multitarea, mecanismos de sincronización, planificador de tareas, etc), ya que posee un display que permite imprimir resultados de la ejecución de un programa. Debido a que el procesador es un ARM7, se pueden utilizar las herramientas GNU como entorno de desarrollo. Además existen programas gratuitos [12], [13] que emulan el funcionamiento del dispositivo.

Como se sabe, el GBA funciona con «cartuchos» de juegos. El conector donde se insertan estos juegos proporciona los buses de datos, direcciones y control (Rd, Wr, IRQ). Un cartucho está formado por: una memoria no volátil, la cual almacena el «ejecutable» del juego, una memoria RAM y una interfaz con los buses. Si al momento de energizar el GBA se detecta la presencia de un cartucho, el procesador inicia la ejecución del juego. Si no, el procesador espera que se cargue un programa en la memoria RAM interna. La carga de este programa se hace a través del puerto serie del GBA utilizando el software xboo [14].

El sistema operativo utilizado en las aplicaciones es eCos [15]. En la figura 3, se muestran los componentes de este RTOS. Se eligió eCos debido a que la compañía Charmedlabs lo adaptó al GBA.

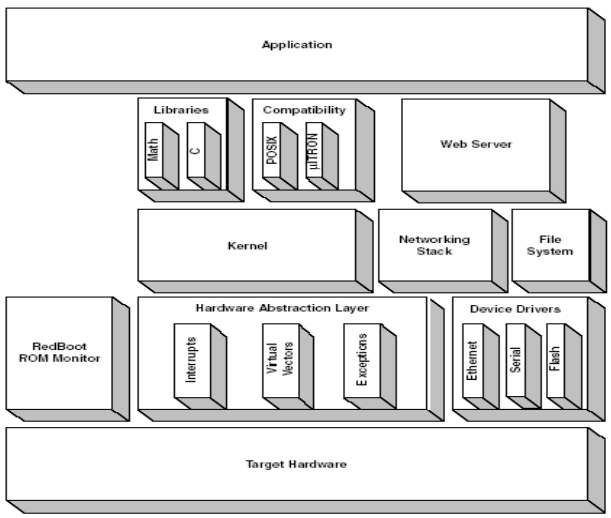

Figura 3. Componentes de eCos[16].

Debido a que las aplicaciones deben ser autónomas, se requiere que el programa esté almacenado en un medio no volátil. Además, como se dijo anteriormente un SE puede requerir funciones HW, las cuales se implementan en un PLD. Para cumplir con estos requerimientos se construyó una plataforma basada en la herramienta comercial XPORT [17]. El diagrama de bloques de nuestra tarjeta se muestra en la figura 4.

Debido a que la FPGA es volátil se dispone de un mecanismo que realiza la configuración de la misma cada vez que se energiza el sistema. Este mecanismo está formado por el puerto JTAG (puerto paralelo del computador), el CPLD (encargado de la comunicación con el PC) y la memoria flash. La memoria flash realiza tres funciones: Guardar la información de configuración de la FPGA, Almacenar el programa que debe ser ejecutado por el GBA y por último almacenar una segunda configuración de la FPGA necesaria para establecer un medio de programación de la flash (Esta configuración solo se utiliza cuando se programa la flash).

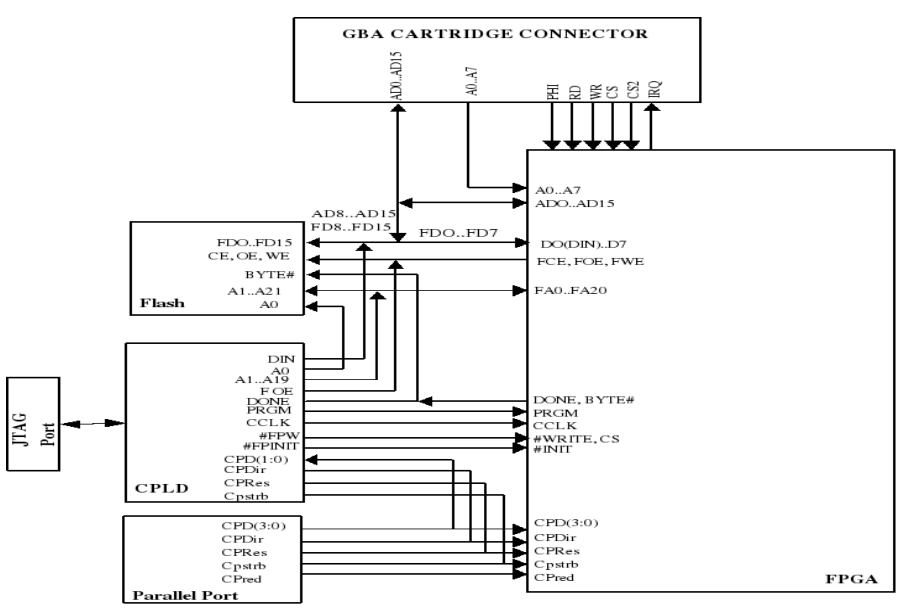

Figura 4. Diagrama de bloques de tarjeta de desarrollo.

En la figura 5 se muestra el diagrama de bloques de la implementación de una aplicación utilizando la tarjeta de desarrollo. Como puede observarse, la FPGA proporciona un camino de conexión entre el GBA y la FLASH, además proporciona los recursos HW para implementar periféricos propios de la aplicación.

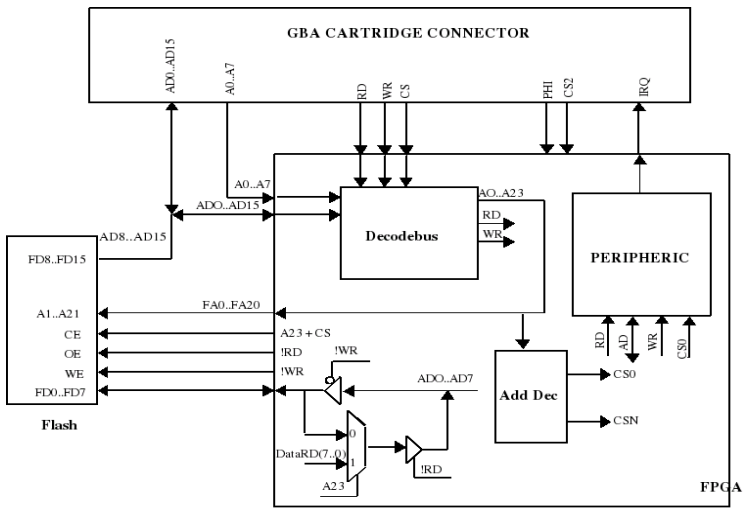

Figura 5. Diagrama de bloques de una aplicación típica utilizando la tarjeta de desarrollo.

4.2 Tarjeta de desarrollo basada en un procesador AT91

A pesar de los excelentes resultados arrojados por el sistema de desarrollo basado en el GBA, esta no es una alternativa que se pueda utilizar en ambientes industriales aislados, en los cuales no existe la intervención humana. Para poder realizar aplicaciones comerciales en las que se requiera la utilización de un SE, se diseñó una tarjeta de desarrollo basada en un procesador ARM comercial de ATMEL el AT91. Esta tarjeta de desarrollo fue diseñada de tal forma que pudiera remplazar al GBA en aplicaciones en las que no fuera necesario utilizar un dispositivo de vídeo y pudiera adaptarse perfectamente a la tarjeta de desarrollo XPORT. En la figura 6 se muestra una fotografía de la tarjeta.

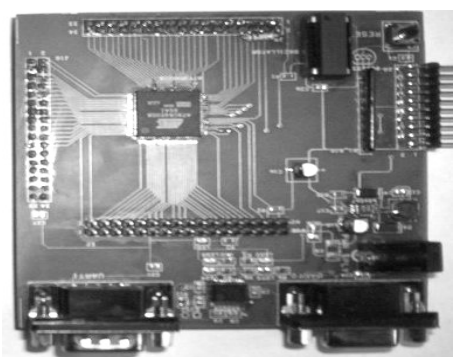

Figura 6. Tarjeta de desarrollo basada en un procesador AT91 de ATMEL.

Esta plataforma de desarrollo solo posee los componentes mínimos que permitan depurar aplicaciones, es decir, el procesador (con una memoria RAM interna de 256 kB), el puerto de programación JTAG y puertos seriales de diagnóstico y aplicación.

4.3 Sistema de vigilancia con transmisión de imágenes vía Internet

En esta aplicación existe una cámara conectada a la red Ethernet a través de un GBA, el GBA se encarga de recibir la información de la imagen y publicarla en un servidor WEB. Para la realización de este proyecto se trabajó con el circuito integrado CS8900 y se utilizó el stack lwip (lightweight TCP/IP stack) de eCos[15]. El stack lwip fue diseñado para ser usado en sistemas embebidos y posee un footprint de memoria muy bajo en comparación con el stack BSD. Se eligió este chip y este stack debido a que están soportados bajo eCos, lo cual reduce el tiempo de desarrollo, el driver de eCos para el CS8900 proporciona las siguientes funciones:

- Inicializa el controlador: (cs8900a_init)
- Comienza la interfaz (cs8900a_start)
- Detiene la interfaz (cs8900a_stop)
- Determina si es posible enviar otro paquete: (cs8900a_can_send)
- Controla la interfase (cs8900a_control)
- Envía (cs8900a_send)
- Recibe (cs8900a_receive)
- Envía un paquete a un protocolo de nivel superior (cs8900a_deliver)

La figura 7 muestra una fotografía del adaptador de red diseñado e implementado.

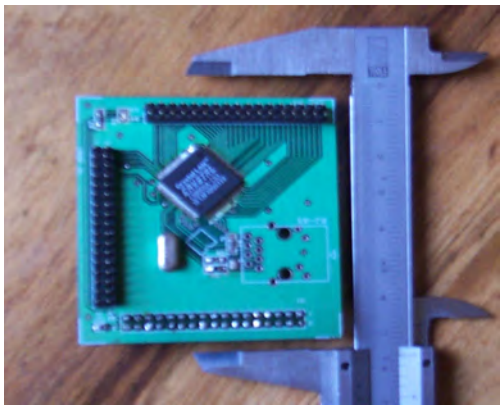

Figura 7. Tarjeta de red con el IC CS8900.

4.4 Osciloscopio de un canal

Este proyecto explora la capacidad de la plataforma para desarrollar tareas de tiempo REAL, para esto se implementaron dos aplicaciones basadas en la siguiente arquitectura:

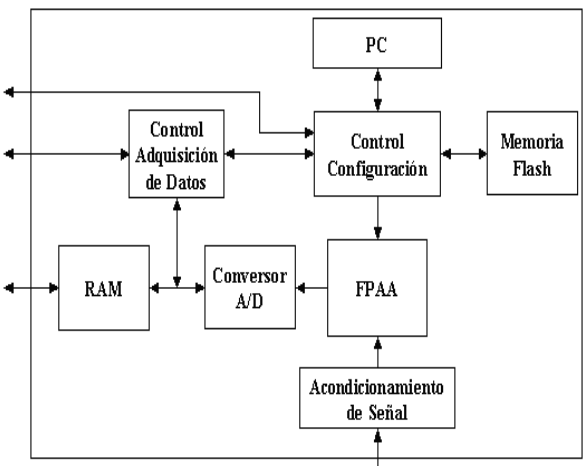

Figura 8. Diagrama de bloques de la tarjeta de adquisición de datos.

El módulo principal es un FPAA (Field Programmable Analog Arrays), dispositivo que puede ser configurado parcialmente de forma digital en tiempo de ejecución, para implementar una gran variedad de aplicaciones análogas como filtros, amplificadores, controladores, etc. Esto permitirá tener una plataforma de adquisición de datos muy flexible para dejar abierta cualquier otra aplicación que se desee desarrollar.

Debido a que la FPAA es volátil se debe proporcionar un mecanismo de configuración automática similar al de la tarjeta XPORT. Para este fin se tiene el bloque *Control configuración*. La información de la configuración es estática para el caso del osciloscopio y dinámica para el caso del PID autosintonizado.

Una vez implementado el circuito análogo dentro del FPAA, se necesita convertir estos datos a formato digital para su posterior transmisión al GBA. Para esta tarea se utiliza un conversor Análogo-Digital

Estos datos que convierte el conversor no podrán ser enviados inmediatamente al GBA, ya que la velocidad de procesamiento de este disminuirá la rápida adquisición de las señales, y obviamente disminuye el ancho de banda que se podrá graficar. Debido a esto la adquisición debe hacerse por hardware, para lo cual se cuenta con un módulo *Control Adquisición de Datos*.

Este bloque se implementó nuevamente por medio de un CPLD, el cual se encarga de llenar una memoria RAM con una cantidad definida de datos que arroja el conversor, dependiendo de los parámetros de trigger y tiempo con los que se cuente. Una vez adquiridos los datos, se enviará una interrupción al procesador del GBA, para que este los tome de la memoria RAM. De esta forma no importa la velocidad con la que el GBA procese los datos, ya que la adquisición ha ocurrido con anterioridad a alta velocidad, logrando capturar señales de mayor frecuencia. La figura 9 muestra una fotografía de la tarjeta de adquisición de datos.

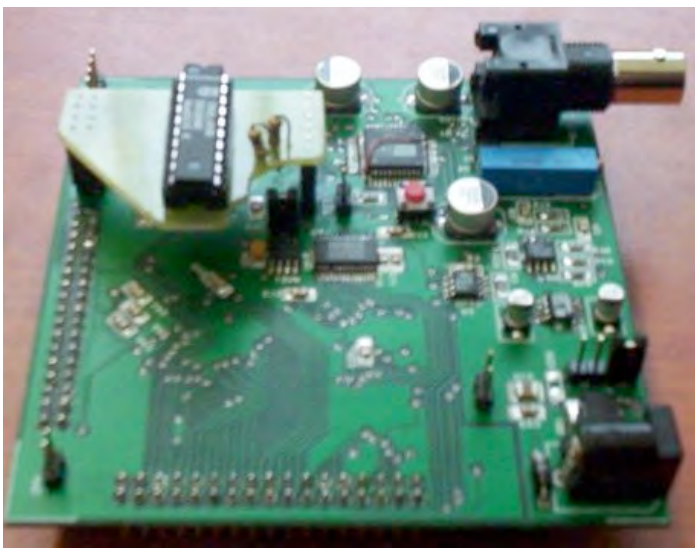

Figura 9. Tarjeta de adquisición de datos.

V. Trabajo futuro

El siguiente estudio es la implementación de aplicaciones utilizando un sistema operativo más poderoso como linux, para esto ya se desarrollaron dos tarjetas, una basada en un procesador ARM720, el cual es básicamente un ARM7 + MMU, este procesador posee los periféricos necesarios para implementar aplicaciones gráficas con lo cual se puede cargar un sistema manejador de ventanas como por ejemplo microwindows. La otra tarjeta es una plataforma de desarrollo para microblaze, posee una SPARTAN III de 400 mil compuertas y se implementarán aplicaciones realizadas con uClinux.

Conclusiones

Los SE proporcionan una herramienta muy poderosa para la implementación de sistemas digitales con alto desempeño.

Las herramientas de libre distribución proporcionan una base de desarrollo sólida y económica.

Los RTOS de libre distribución como eCos, linux y uClinux, facilitan la implementación de aplicaciones, ya que soportan una gran variedad de dispositivos hardware y poseen librerías muy desarrolladas para aplicaciones en SE.

La adopción de esta tecnología es necesaria en países como los nuestros, (en vía de desarrollo) ya que nos pone al mismo nivel que países más desarrollados.

El GBA es un excelente herramienta para la enseñanza de SE.

Referencias bibliográficas

[1] Proyecto código abierto. ARM Linux Project. Disponible en: www.arm.linux.org.uk/

[2] Red Hat, inc. Proyecto de código abierto eCos. Disponible en: http://sources.redhat.com/ecos

[3] Universidad de Berkeley. Proyecto ptolemy. Disponible en: http://ptolemy.eecs.berkeley.edu/

[4] Red Hat, inc. Proyecto de código abierto SID: http://sources.redhat.com/sid.

[5] Mentor graphics inc. Herramienta de coverificación www.mentor.com/products/fv/hwsw_coverification/seamless/index.cfm

[6] Sharp semiconductor. Sitio web: www.sharpsma.com

[7] Cirrus logic. Sitio web: www.cirrus.com

[8] Xilinx dispositivos lógicos programables. Sitio web: www.xilinx.com

[9] Red Hat, inc. Proyecto abierto binutils. Disponible en http://sources.redhat.com/binutils/

[10] Red Hat, inc. Proyecto abierto GDB. Disponible en: http://sources.redhat.com/gdb/

[11] Proyecto HRI. Armtool, herramienta para control del In Circuit Emulator en un procesador ARM. Disponible en http://hri.sourceforge.net/tools/armtool.html

[12] Nintendo. Sitio web: www.nintendo.com/systemsgba

[13] Proyecto de código abierto Visual Boy Advance. Disponible en: http://vba.ngemu.com/

[14] Proyecto de código abierto xboo. Disponible en: www.work.de/nocash/gba-xboo.htm

[15] Red Hat, inc. Sistema operativo de tiempo real eCos. Disponible en:http://sources.redhat.com/ecos

[16] A. J. Massa, *Embedded software development with eCOs*, Prentice Hall.