Servidor Web Embebido en una FPGA con Codiseño como Metodología de Diseño

Miguel Ariza Triviño[1] Alvaro Javier Buitrago[2]
[1]Universidad Sergio Arboleda, Bogotá, Colombia, miguel.ariza@usa.edu.co
[2]Universidad Sergio Arboleda, Bogotá, Colombia, alvaro.buitrago@usa.edu.co

ABSTRACT

This paper describes the process of developing a web server embedded in a FPGA device for monitoring and control applications, the system allows monitoring and control variables connected to a physical device from a web page through a commercial browser from any computer Internet connected. The design methodology used is co-design, which allows the optimal use of hardware resources as well as the reuse and scalability of both hardware and software system.

Keywords: Codesign, FPGA, Web Server, Embeded Software

RESUMEN

Este documento describe el proceso de desarrollo de un servidor web embebido en un dispostivo tipo FPGA, para aplicaciones de monitoreo y control, el sistema permite monitorear y controlar variables conectadas al dispositivo fisico desde una pagina web por medio de un navegador comercial desde cualquier computador conectado a Internet. La metodologia de diseño utilizada es codiseño, la cual permite la optimizacion en el uso de los recursos de hardware asi como la reutilizacion y escalabilidad del sistema tanto a nivel de hardware como de software.

Palabras clave: Codiseño, FPGA, Servidor Web, Software Embebido

1. INTRODUCCION

Usualmente los sistemas embebidos poseen una interfaz externa para monitorear el estado de algunas variables, hacer un diagnostico o interactuar con el sistema. Tradicionalmente esto ha sido realizado a través de una terminal serie o interfaces de texto como pantallas LCD y teclados de membrana; la creciente exigencia y sofisticación del usuario final, sumado a la cantidad de datos que se manejan en sistemas de esta naturaleza, ha llevado al desarrollo de interfaces graficas, interactivas y de fácil manejo, además en muchos casos se quiere habilitar la posibilidad de observar y controlar de forma remota. El desarrollo de servidores web embebidos permite tener la cobertura global de la Internet y la familiar interfaz de un navegador para interactuar con sistemas embebidos. Este documento describe la implementación de un servidor web embebido sobre una plataforma expandible y reutilizable para el desarrollo de sistemas electrónicos que puedan ser monitoreados y controlados mediante una página web en un navegador convencional desde cualquier computador conectado a la Web. El sistema propuesto puede usarse en la implementación de aplicaciones prácticas como mantenimiento remoto, actualización de firmware, laboratorios virtuales, casas y edificios "inteligentes" entre otros. Por las características del diseño y con el uso de FPGAs como plataforma de implementación se logra la reusabilidad y escalabilidad del sistema.

2. CONCEPTOS INICIALES

2.1 SERVIDORES WEB

El desarrollo de servidores web embebidos, permite en el acceso a sistemas distribuidos de instrumentacion y control, y provee una solucion para laboratorios, instrumentacion, industria y automatizacion de hogares, los usuarios pueden monitorear y controlar diferentes tipos de transductores desde paginas web activas (Klimchynski, 2003).

Algunas de las aplicaciones mas reconocidas en la implementación de servidores web embebidos son: mantenimiento y diagnostico remoto de equipos, automatización de hogares, monitoreo para sistemas de seguridad y control de maquinas dispensadoras entre otras. Existen diferentes tecnologias para la implementacion de servidores web embebidos, la mayoria basados en microcontroladores como el Tiny Renesas H8® (Moraleda, 2005), o el AVR460® (Atmel, 2005), estos dispositivos presentan caracteristicas particulares limitadas a cada familia, naturalmente no es posible cambiar sus caracteristicas fisicas. Por otro lado la implementacion de este tipo de sistemas sobre dispositivos reprogramables tipo FPGA (Actel, 2005), presenta la ventaja de la flexibilidad y escalabilidad de los sistemas implementados .

Una de las tecnicas mas usadas para el desarrollo de servidores web embebidos es el uso de Sistemas Operativos en Tiempo Real (RTOS), donde el servidor web provee la interface entre el navegador y la aplicación como una tarea separada dentro del RTOS y puede ejecutarse en paralelo con la aplicación o como parte integral de la aplicación. La aplicación maneja el resto del hardware externo y provee la interface a los sistemas de adquisición de datos[2]. El uso de codiseño sobre FPGAs, puede evitar la necesidad de usar RTOS, llevando funciones criticas a que se ejecuten en hardware o usando procesadores en paralelo (Sun, J. and Ryser P., 2006). En la medida que Ethernet se ha estado moviendo de las redes tradicionales a aplicacicones sobre sistemas embebidos, se ha convertido en un puerto de comunicaciones estándar para la mayoria de las FPGAs[3].

2.2 CODISEÑO

El termino codiseño hardware/software es usado para denotar la interacción entre los flujos de diseño de hardware y software en un sistema embebido, diferente al desarrollo tradicional de hardware y software al mismo tiempo o en paralelo, pero aisladamente. El flujo tradicional de diseño hardware-software, se describe en la Figura 1.

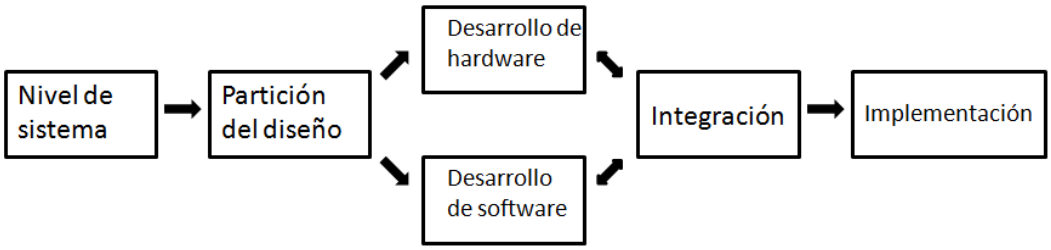

Figure 1: Flujo tradicional de diseño hardware-software

1. Nivel de Sistema: se definen las especificaciones funcionales del sistema y los parámetros de desempeño.
2. Partición del Diseño: se divide el diseño en hardware y software.
3. Desarrollo hardware/software: se desarrolla hardware y software en paralelo como tareas aisladas.
4. Integración: se unen los componentes de hardware software para formar el sistema integrado.
5. Implementación física del diseño.

Las técnicas de codiseño buscan básicamente evitar el aislamiento del diseño de hardware con el diseño de software, ya no se ven como sistemas individuales que se unen en una aplicación sino sistemas, que se desarrollen en un proceso de diseño integrado. El principal objetivo de usar técnicas de codiseño hardware-software es dar la posibilidad real al diseñador de decidir cuales funciones se ejecutan en hardware y cuales

en software buscando optimizar el desempeño de un sistema embebido, especialmente en sistemas de desempeño crítico. Mientras que los procesadores discretos tienen una arquitectura fija, los Cores de procesadores basados en FPGAs, proveen la libertad de determinar si una función se ejecuta en hardware o en software. Hoy en día, las FPGAs, están siendo usadas como la principal plataforma en muchos sistemas embebidos y están desplazando a los microprocesadores de sus mercados tradicionales (Hauff, M. 2007). El flujo de codiseño usado en este proyecto es el presentado en la Figura 2 donde la parte de software y hardware se desarrollan de manera sincronizada, y es posible mover funciones de uno a otro dominio durante el proceso de diseño.

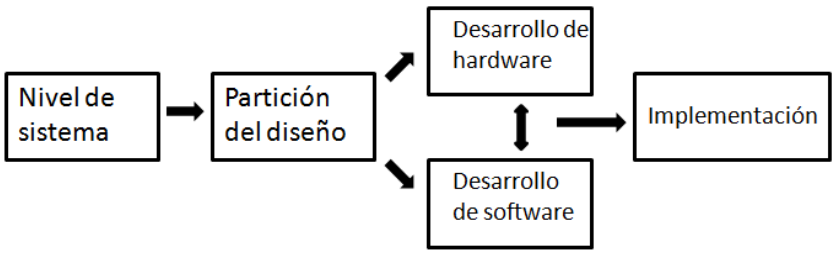

Figure 2: Flujo de codiseño utilizado

3. DISEÑO E IMPLEMENTACION

El objetivo de este proyecto es desarrollar un servidor web embebido capaz de enviar y recibir datos de tal forma que podamos actuar sobre un sistema físico de manera remota. El sistema propuesto consiste en una plataforma reconfigurable con un servidor web embebido, que permita la interacción con diferentes sensores y actuadores conectados al sistema desde cualquier navegador tradicional conectado a Internet. Integrar todo el sistema dentro de un mismo dispositivo reprogramable tipo FPGA, permite la flexibilidad de usar el sistema en diferentes tipos de aplicaciones, se habilita un puerto I2C al sistema para conexión de sensores y puertos de entrada salida digital de propósito general para actuadores.

3.1 DISEÑO DE HARDWARE

La herramienta de diseño utilizada para el desarrollo del sistema es Altium Designer®, y la plataforma de implementación física usada para validar el diseño es Nanoboard II®, sobre un dispositivo Spartan 3 de Xilinx®, la metodología de diseño utilizada parte de de las especificaciones funcionales del sistema, el sistema propuesto consta de un puerto I2C para conexión de diferentes tipos de sensores, un puerto de ocho líneas de salida de propósito general para controlar variables digitales, un puerto de ocho líneas de entradas digitales para señales de control y dos memorias SRAM, de 516MB cada una para almacenar el código del programa, las variables del sistema y la pagina web. El diseño de alto nivel es el siguiente:

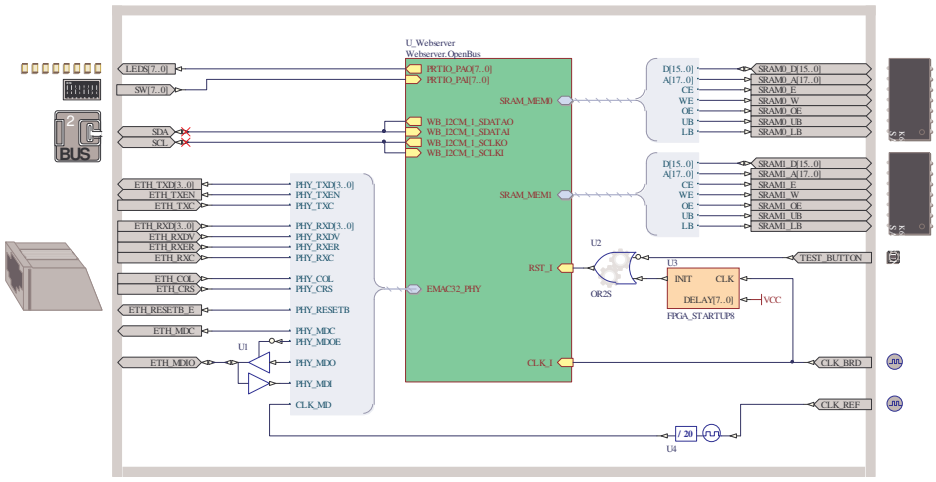

Figura 3: Nivel de Sistema

El diseño del hardware digital se desarrolla usando descripción basada en Openbus®. Openbus es un sistema de descripción de sistemas digitales basado en la arquitectura de interconexión Whishbone, que permite interconectar Cores de propiedad intelectual (IPs) dentro de una FPGA. El Core principal del sistema es un procesador TSK3000®, un Core de procesador de 32 bits, compatible Wishbone de arquitectura RISC.

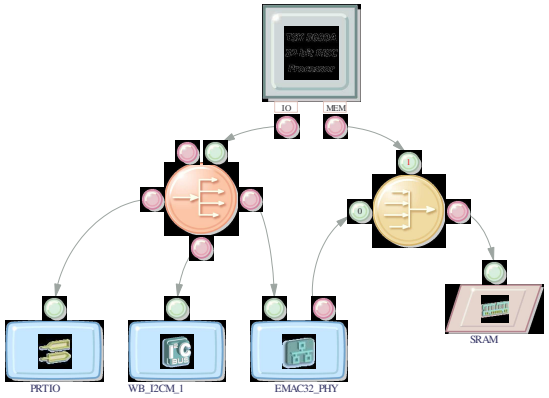

Figure 4: Descripción del hardware del diseño

El procesador TSK3000 se configura con una memoria interna de 16KB, el programa y las variables se almacenan en la memoria externa SRAM, para optimizar recursos en la FPGA. El controlador de periféricos EMAC32 (Ethernet Media Access Controller) provee una interface entre el procesador y la capa física de Ethernet (PHY).

3.2 DISEÑO DE SOFTWARE

El diseño de software se desarrolla teniendo en cuenta las restricciones de hardware en velocidad de transmisión, capacidad de procesamiento y memoria utilizada, el primer paso de diseño es desarrollar una librería adecuada que permita aprovechar de manera eficiente los recursos de hardware. Para lograr este objetivo se debe disponer de un Stack de protocolos de transmisión que deben incluir:

- ICMP (Internet Control Message Protocol): Se debe incluir debido importancia para encontrar y corregir errores en la red.
- IP (Internet Protocol): Protocolo fundamental para la transmisión de datos a través de Internet.

- TCP (Transmisión Control Protocol): Otro protocolo ampliamente utilizado para la transmisión de información, implica implementar la creación y configuración de Sockets.

El completo desarrollo de un Stack que contenga por lo menos estos tres protocolos es una tarea que puede tomar mucho tiempo, por esta razón, se decide utilizar algunos de los Stacks de protocolos ya existentes, las características principales que este Stack debe tener son:

- Estar escrito en ANSI C, esto con el fin de garantizar la reusabilidad del sistema en otros sistemas embebidos.
- Ser rápido y liviano con el fin de que cumpla con las restricciones de hardware del sistema.
- Estar desarrollado bajo una licencia de Software Libre, para poderlo implementar abiertamente en diferentes plataformas.
- Su integración con los demás módulos del sistema debe ser fácil.
- Soporte sobre la librería.

Se encontraron dos librerías con estas características, estas librerías son uIP y lwIP (Dunkels, A. 2009), las dos librerías tienen características muy parecidas, sin embargo, encontrar soporte de uIP para el procesador TSK3000 no fue posible, por lo que la decisión final fue usar lwIP.

3.2.1 Stack lwIP para el procesador TSK3000.

La principal ventaja del stack lwIP sobre el stack uIP es su soporte en la plataforma de desarrollo Altium Designer® y su fácil integración con las librerías de Ethernet brindadas por la plataforma, sin embargo es esencial entender su arquitectura y sus principales funciones para poder construir sobre este Stack una plataforma sólida para el Servidor Web.

El Stack de protocolos lwIP es una implementación del stack TCP/IP. El diseño de este Stack se enfocó en mantener el consumo de memoria bajo, al igual que el tamaño del código, con el fin de que este Stack se pudiera utilizar en sistemas embebidos, para lograr este objetivo el código de lwIP evita al máximo la copia de información entre buffers y utiliza en su lugar intercambio de apuntadores.

Los módulos más importantes del Stack lwIP son:

- Administración de Buffer y Memoria: En todo sistema de comunicación un punto crítico en el que se debe enfatizar bastante es en el manejo de memoria debido a que los buffers que se manejan en este tipo de sistemas pueden variar de tamaño frecuentemente. El protocolo TCP puede manejar buffers de algunos cientos de Bytes mientras que los otros protocolos manejan buffers de unos pocos Bytes.
- Interfaces de Red: Guardan toda la información de las diferentes interfaces de red que se estén utilizando en el sistema.
- Protocolos: Cada protocolo se implementa de forma independiente, sin embargo en algunas partes del código esta división es más bien difusa y es necesario tener en cuenta cierto protocolo para poder implementar el siguiente, aunque este punto puede dar para una discusión sobre la mejor forma de implementar los protocolos, en este caso es transparente para el desarrollo del servidor el cual está en un nivel de abstracción más alto que todo el Stack de protocolos.
- Interface: El Stack lwIP se puede utilizar de dos formas diferentes desde el servidor, mediante funciones o utilizando la API (Aplication Peripheral Interface) nativa del Stack. La API está desarrollada pensando en utilizar el Stack junto con un Sistema Operativo, el cual no es el caso para este diseño, por otro lado las funciones ofrecen control directo sobre el Stack y se pueden utilizar en sistemas sin Sistemas Operativos como es nuestro caso.

3.2.2 Servidor Web

La primera aproximación al servidor Web que se desea implementar es bastante simple, lo que se pretende es poder guardar en memoria una página web y transmitirla a través de una red a un PC, la página web enviada debe tener la posibilidad de enviar instrucciones sencillas al servidor Web por medio del método GET o del método POST.

Teniendo en cuenta estas especificaciones podemos crear un diagrama de estados general con las principales funciones que debe contener el servidor como se muestra en la Figura 5.

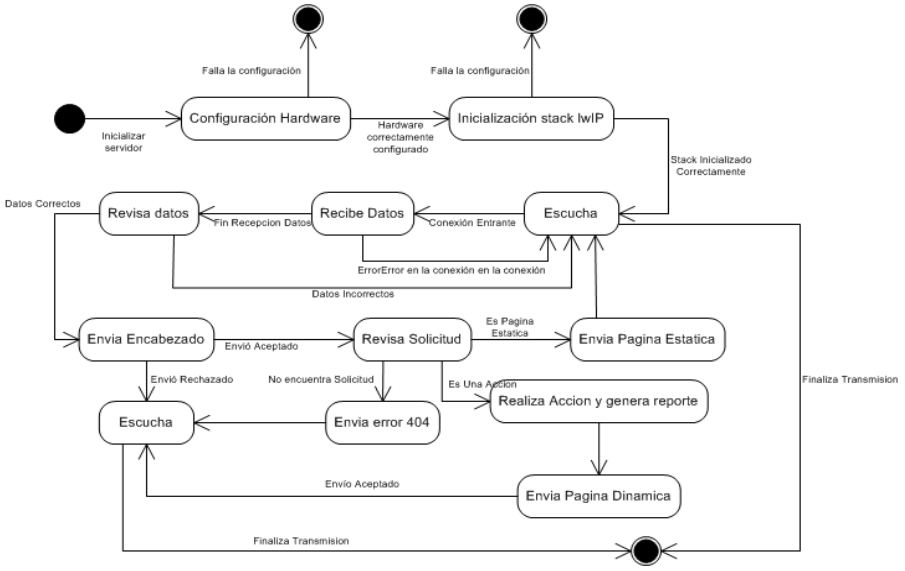

Figura 5: Funciones de software

3.2.3 Funciones del software

Las funciones representadas en la Figura 5, son implementadas en lenguaje C, para ser ejecutadas en el Procesador TSK3000 y se describen a continuación:

- **Configuración del hardware:** Es el primer paso que se realiza en todo sistema de cómputo, en este caso se refiere a inicializar el hardware de comunicación, es decir el controlador y el driver para el dispositivo EMAC32, y en este caso particular también configurar el puerto de entrada-salida para interactuar con los actuadores, como se había mencionado antes el hardware habilita la posibilidad de conectar dispositivos I2C, pero para la primera etapa de este trabajo se usan solo señales digitales provenientes de los puertos de entrada salida. La inicialización del hardware puede fallar por diferentes motivos los cuales se consideran transparentes para nuestro diseño, si el hardware no se puede inicializar correctamente, el programa se da por terminado.
- **Inicialización del Stack lwIP:** Se refiere a la configuración de cada uno de los protocolos de comunicación, esta tarea la hace lwIP de manera casi automática, sin embargo es necesario configurar la dirección IP, el Gateway y la máscara de subred manualmente, sí lwIP falla en inicializar el Stack de Protocolos el programa se termina con una advertencia de error.
- **Escucha:** Este es el estado donde permanece más tiempo el software, este estado simplemente es de espera a que llegue una conexión entrante, también es el estado desde donde se termina el programa, en ningún otro estado se debe terminar el programa porque esto implicaría que se dejaron de realizar tareas y por lo tanto se pueden generar fallas en la recepción de los datos.

- **Recibe Datos:** Una vez se recibe una conexión se utiliza el Stack LwIP para recibir los datos útiles de la transmisión, en este punto del programa se deben verificar posibles errores en la conexión que impide que se reciban todos los datos.
- **Revisa Datos:** En este estado se revisan los datos con el fin de verificar que su contenido sea correcto y que por lo tanto se puedan procesar, en esta primera aproximación al problema solo se van a recibir peticiones de páginas estáticas y peticiones con el método Get que permitan controlar un grupo de leds conectados al puerto de salidas digitales y que permita, además, enviar información sobre el estado de estos leds al navegador.

- **Envía Encabezado:** Si los datos recibidos son correctos, quiere decir que el servidor debe enviar una respuesta, ya sea esta un error o no, por lo tanto en este punto del programa se puede enviar el encabezado HTML por adelantado, en los pasos subsecuentes se enviará la demás información dependiendo de la solicitud realizada.
- **Revisa Solicitud:** Se encarga de verificar el tipo de solicitud que se hizo y de recopilar los datos necesarios para atender esta solicitud, si es una página estática se procede a verificar si la página existe, para esto se debe comparar la página recibida con las páginas que están guardadas en la memoria del servidor, si ninguna página coincide se envía el error de página no encontrada, si por otro lado hay una página que coincide se prepara la página para su envío y en un estado posterior se transmite.
- **Realiza Acción y Genera Reporte**: En este estado se debe realizar la acción solicitada por el usuario de la página Web, y sin importar el resultado se debe generar un reporte de la acción que se realizó, este reporte debe decir si la acción se pudo realizar o no y además dar información acerca del hardware sobre el cual se realizó la acción.
- **Enviar Pagina Dinámica:** Se envía la página que contiene la información del comportamiento del hardware ante la acción solicitada, se le llama página dinámica ya que depende del comportamiento del hardware.
- **Enviar Página Estática:** En este estado simplemente se envía una página guardada en la memoria del servidor que no tiene cambios, por ejemplo, con información sobre el servidor, la plataforma de desarrollo o inclusive algunas fotos.

Figura 6: Pagina web

4. RESULTADOS Y CONCLUCIONES

El uso de FPGAs, unido a la metodología de codiseño utilizada permitió una rápida implementación del sistema propuesto, la posibilidad real de poder diseñar el software y el hardware sincronizadamente permite optimizar tiempos de ejecución y recursos físicos del dispositivo. El campo del codiseño es un amplio campo de investigación, existe una amplia diversidad de aplicaciones, en la medida que los sistemas embebidos aumentan de complejidad es necesario contar con metodologías adecuadas que permitan optimizar los diseños sin incurrir en penalidades de tiempo y recursos.

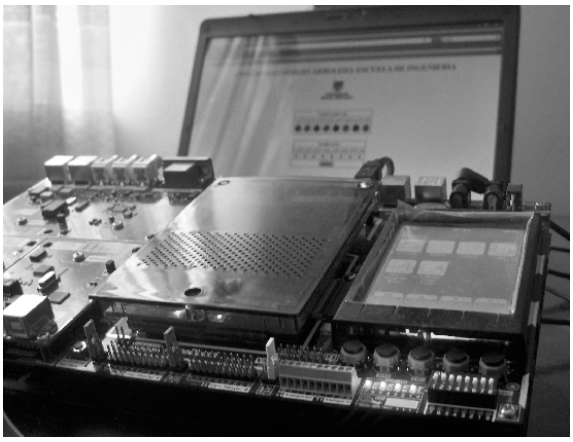

Figura 7: Implementación Final

El presente proyecto se presenta como un aporte al desarrollo de los sistemas embebidos interconectados por Internet, el campo de aplicaciones de este tipo de sistemas se ha mantenido en continuo crecimiento durante los últimos años, la posibilidad de conectar diferentes sensores y actuadores a este sistema y el hecho de que este implementado en una FPGA, hace posible su reutilización en proyectos de diferente naturaleza y complejidad incluso aplicaciones con imágenes o video.

REFERENCIAS

Actel Application Brief, (2005). “Designing a web server system using CoreMP7”

Atmel Aplication Note 3296 (2005). “AVR460: Embedded Web Server”.

Dunkels, A. (2009). http://www.sics.se/~adam/lwip/doc/lwip.pdf, 10/11/10

Dunkels, A. (2009). http://www.sics.se/~adam/uip/uip-1.0-refman, 15/11/10

Hauff, M. (2007). “Compiler directed codesign For FPGA-based embedded systems”, Ph.D. thesis RMIT University, Sydney, Australia.

Klimchynski, I. (2003). “Extensible embedded web server for Internet-based data acquisition and control”.

Sun, J. and Ryser P. (2006). “Implementing a lightweight web server using PowerPC and tri-mode Ethernet MAC in Virtex-4 FX FPAGs”. Xilinx, Inc.

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.