6º Jornadas de Desarrollo e Innovación Tecnológica | 2007
Instituto Nacional de Tecnología Industrial
ELECTRÓNICA
Desarrollo e Innovación

Interfaz de depuración para microcontrolador, descripción de hardware

Tropea, S. E.[i]

[i]INTI-Electrónica e Informática

Introducción

El desarrollo de sistemas electrónicos basados en sistemas embebidos presenta un desafío a la hora de eliminar errores de implementación. Esto se debe a que los microcontroladores utilizados para estas tareas suelen ser pequeños y no es posible que ejecuten las complejas tareas involucradas en la depuración de errores. La tarea se dificulta aún más cuando dichos dispositivos no poseen una interfaz de usuario amistosa, sin salida de video donde conectar un monitor ni entradas de teclado o similares para ingresar datos.

Para solucionar estos problemas se suele incluir funcionalidad en el microcontrolador que permite realizar las tareas de depuración en forma remota utilizando una computadora personal, donde se encuentran disponibles los recursos antes mencionados.

En este trabajo presentamos el desarrollo de una interfaz de depuración especialmente creada para ser usada en conjunto con nuestro microcontrolador [1] compatible con el *PIC* 16C84[2] de Microchip[3]. Cabe destacar que el microcontrolador original no dispone de estas funcionalidades y que sólo microcontroladores más avanzados de su familia poseen limitadas capacidades de este tipo. Este trabajo complementa el desarrollo de nuestro microcontrolador, también presentado en estas jornadas[4], por lo que recomendamos la lectura de dicho trabajo con anterioridad a la de este.

El objetivo de este trabajo fue el de obtener un mecanismo que permitiera eliminar errores de implementación en proyectos basados en nuestro microcontrolador.

Metodología

Las herramientas de desarrollo utilizadas fueron las recomendadas por el proyecto FPGALibre[5][6][7] impulsado por nuestro laboratorio. Para este desarrollo se utilizaron estaciones de trabajo que corren Debian[8] GNU[9]/Linux.

Se procedió a agregar dos puertos especiales al microcontrolador, uno de entrada y otro de salida. Debido a que dichos puertos poseerían numerosas señales, a que las mismas serían usadas opcionalmente en el caso de que se conectara este accesorio y a que las mismas podrían cambiar en el futuro, se utilizaron los llamados *records* de *VHDL* que permiten agrupar señales, tal como una estructura en C permite agrupar variables.

Se agregó funcionalidad al microcontrolador para poder detener su ejecución, inspeccionar sus registros y monitorear la actividad del bus de datos. Esta funcionalidad se exportó a través de los puertos antes mencionados.

Se diseñó un periférico compatible con WISHBONE[10] que implementara la funcionalidad necesaria para controlar las señales antes mencionadas. Se seleccionó el estándar de interconexión WISHBONE con el objetivo de que el periférico pudiera ser accedido utilizando distintos mecanismos de comunicación entre la computadora y la *FPGA*. Para el diseño inicial se seleccionó un puente que permite interconectar el puerto paralelo de una PC con un bus WISHBONE[11][12], este periférico fue desarrollado por nuestro laboratorio y es una versión mejorada del anteriormente publicado. De esta manera el nuevo periférico es controlable desde el puerto paralelo de una PC.

Se creó un programa escrito en C++ que permite controlar este periférico utilizando el puerto paralelo de una PC. Debido al hecho de que crear la interfaz de usuario para la depuración de un programa es una tarea muy compleja se optó por aprovechar una interfaz de usuario ya existente. De esta manera sólo fue necesario concentrarnos en la parte técnica del problema. Para comunicar nuestro programa con una interfaz de usuario ya existente se optó por implementar el protocolo denominado GDB/MI (*GNU DeBugger Machine Interface*). Este protocolo es parte del depurador del proyecto GNU, el GNU *debugger*, y permite

controlarlo desde una interfaz de usuario amigable. De esta manera fue posible utilizar una interfaz de usuario que originalmente fue diseñada para controlar al GNU *debugger*.

Para dicha interfaz de usuario se seleccionó el programa SETEdit[13]. El mismo fue desarrollado por el autor de este trabajo y es el entorno de trabajo recomendado por el proyecto FPGALibre. SETEdit implementa el protocolo GDB/MI para la depuración de programas en C/C++ y *assembler* por lo que no fueron necesarios cambios importantes para lograr que el mismo se adaptara a la depuración de programas escritos en lenguaje de ensamblador corriendo en nuestro microcontrolador.

Nuestro microcontrolador no implementa la funcionalidad de reprogramación incluida en el original. Esto no es un problema importante debido a que las *FPGAs* son reconfigurables y por lo tanto basta con volver a sintetizar el diseño para modificar el programa ejecutado por el microcontrolador. Es común que los depuradores remotos puedan modificar el programa ejecutado por el sistema embebido por lo que para complementar este desarrollo se diseñó un periférico WISHBONE capaz de acceder a la memoria de programa del microcontrolador. Esto permitió reconfigurar dicha memoria sin necesidad de reconfigurar toda la *FPGA*.

La *Fig. 1* ilustra la interconexión entre los distintos componentes de hardware antes mencionados, los bloques ilustrados en color verde corresponden a los desarrollados para este trabajo. En la *Fig. 2* se muestra el flujo de datos dentro de la computadora, los datos ingresan a través del puerto paralelo utilizando el protocolo EPP, son tomados por el sistema operativo (Linux) utilizando operaciones básicas de entrada/salida y enviados al espacio de usuario utilizando el dispositivo *ppdev*, estos datos son procesados por el programa depurador (*icepic*) y enviados a la interfaz de usuario (SETEdit) utilizando el protocolo GDB/MI.

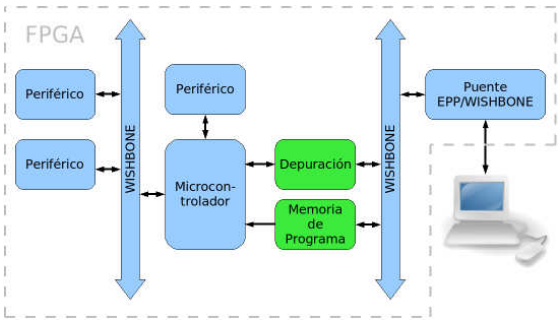

Fig. 1: Diagrama de conexiones de los bloques de hardware involucrados.

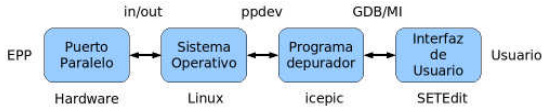

Fig. 2: Flujo de datos dentro de la computadora.

Resultados

Nuestro desarrollo permite:

—Detener/Reanudar la ejecución del microcontrolador en cualquier momento.

—Ejecutar su programa paso a paso.

—Detener la ejecución cuando se alcanzó una posición de memoria determinada, punto de parada o *breakpoint*. La cantidad de *breakpoints* es configurable entre 1 y 256.

—Reinicializar el microcontrolador.

—Acceder a registros especiales tales como el acumulador, el puntero de la pila y el contador de programa.

—Acceder a todos los registros de la memoria de datos, incluyendo registros especiales tales como el estado del microcontrolador.

—Modificar cualquier registro, inclusive modificar el contador de programa forzando un salto.

—Inspeccionar la pila de llamadas (*calling stack*).

—Detener la ejecución cuando se accede a una posición de memoria de datos, *watchpoint*. Los accesos pueden seleccionarse para detenerse por lectura, escritura o ambos. El número de *watchpoints* es configurable entre 1 y 256.

—Alterar la memoria de programa.

El área ocupada depende de la cantidad de *breakpoints* y *watchpoints* configurados, para el caso de tres *breakpoints* y tres *watchpoints* se observó un uso de 186 *slices*. Esto incluye el periférico utilizado para modificar la memoria de programa del microcontrolador y el puente que conecta el puerto paralelo de la PC con el bus WISHBONE antes mencionado. Esto corresponde al 9,7 % de una *Spartan* 3[14] 200 o bien sólo un 1,3 % de una *Spartan* 3 1500.

Conclusiones

Se obtuvo una herramienta poderosa, capaz de realizar la mayor parte de las operaciones realizadas por depuradores utilizados en computadoras personales, y que es de gran ayuda a la hora de buscar errores en sistemas funcionando en tiempo real.

El uso de *FPGAs* posee como ventaja el hecho de que este agregado puede removerse en la versión

definitiva del diseño con lo que el mismo no ocupa recursos en el dispositivo final. En el caso en que el diseño ocupe prácticamente el total de la *FPGA* basta con usar una *FPGA* más grande durante la etapa de desarrollo, a los fines de incluir unidades de depuración como esta.

Otra ventaja inherente al uso de *FPGAs* es que este periférico puede ser configurado. De esta manera si se necesita un número mayor de *breakpoints* y/o *watchpoints* basta con reconfigurarlo y volver a sintetizar el diseño.

Debido a que la depuración se realiza *in-circuit* (dentro del circuito) es posible utilizar este desarrollo como base para el diseño de un emulador del tipo *in-circuit*. Este tipo de herramientas son comunes cuando se trabaja con microcontroladores como el 16C84 que no posee funcionalidad para depuración. Se trata de herramientas costosas. Por lo que este diseño no sólo puede utilizarse para el caso en que se usen *FPGAs* sino también para desarrollos que utilicen el microcontrolador original.

La selección del estándar de interconexión WISHBONE permitió el reuso de nuestro puente de puerto paralelo y abre la posibilidad a la implementación de otro tipo de mecanismos de comunicación, como podrían ser RS-232 o USB.

La utilización del protocolo GDB/MI permitió acelerar notablemente el desarrollo y reusar interfaces de usuario ya existentes y con las cuales nuestro equipo ya se encontraba familiarizado.

La utilización de las herramientas propuestas por el proyecto FPGALibre mostró ser adecuada para este desarrollo.

Referencias

[1] S. E. Tropea, J. P. D. Borgna, "Microcontrolador compatible con PIC16C84, bus Wishbone y video", *FPGA* Based Systems, ISBN 84-609-8998-4, 2006.
[2] Microchip Technology Inc., "8 bits CMOS EEPROM Microcontroller", p 8 (http://ww1.microchip.com/downloads/en/devicedoc/30445c.pdf, verificado 27/06/2007).
[3] Microchip Technology Inc., http://www.microchip.com/
[4] S. E. Tropea, "Microntrolador compatible con *PIC* 16C84, descripción de hardware", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.
[5] S. E. Tropea, D. J. Brengi, J. P. D. Borgna, "FPGALibre: Herramientas de Software Libre para diseño con *FPGAs*", *FPGA* Based Systems, ISBN 84-609-8998-4, pp 173-180, 2006.
[6] S. E. Tropea, D. J. Brengi, J. P. D. Borgna, S. N. Gwirc, "FPGALibre: Herramientas de Software Libre para diseño con *FPGAs*", 6° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2007.
[7] Proyecto *FPGA* Libre, http://fpgalibre.sourceforge.net/
[8] Debian project, http://www.debian.org/
[9] GNU project, http://www.gnu.org/
[10] Silicore and OpenCores.Org, "WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores", http://prdownloads.sourceforge.net/fpgalibre/wbspec_b3-2.pdf?download
[11] A. Trapanotto, D. J. Brengi, S. E. Tropea, "Puente IEEE1284 en modo EPP a bus Wishbone", *FPGA* Based Systems, ISBN 84-609-8998-4, 2006.
[12] A. Trapanotto, D. J. Brengi, S. E. Tropea, "Descripción de hardware para interfase con puerto paralelo tipo IEEE 1284", 5° Jornadas de Desarrollo e Innovación Tecnológica, Instituto Nacional de Tecnología Industrial, Argentina, 2004.
[13] S. E. Tropea y otros, "SETEdit, un editor de texto amigable", http://setedit.sourceforge.net.
[14] *Xilinx Spartan* 3 *FPGA*, http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/

Para mayor información contactarse con:

Ing. Salvador E. Tropea - salvador@inti.gov.ar