INTERFAZ DE DEPURACIÓN PARA MICROCONTROLADOR

Tropea, S. E.

Electrónica e Informática
Instituto Nacional de Tecnología Industrial
Buenos Aires, Argentina
email: salvador@inti.gov.ar

ABSTRACT

En este trabajo presentamos el desarrollo de una interfaz de depuración desarrollada para ser usada con un microcontrolador compatible con el PIC 16C84.

La misma permite controlar la ejecución del microcontrolador permitiendo detenerla y/o reanudarla. Para detener la ejecución se proveen mecanismos directos así como puntos de parada en el código (*breakpoints*) o por acceso y/o modificación de datos (*watchpoints*). Esta interfaz permite acceder a registros internos del controlador e inspeccionar el *stack*.

Este core se complementa con el desarrollo de un programa que permite la utilización de interfaces de usuario de alto nivel ya existentes. Esto se logró utilizando el protocolo GDB/MI del GNU *debugger*.

1. 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 un microcontrolador compatible con el PIC 16C84 de Microchip [1] implementado en la misma FPGA. Dicho microcontrolador [2] [3] fue desarrollado por nuestro equipo de trabajo. 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.

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

2. METODOLOGÍA

Las herramientas de desarrollo utilizadas fueron las recomendadas por el proyecto FPGALibre [4] [5]. Para este desarrollo se utilizaron estaciones de trabajo que corren Debian [6] GNU [7] /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 [8] 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 [9] que permite interconectar el puerto paralelo de una PC con un bus WISHBONE. De esta manera el nuevo periférico pudo ser controlado utilizando el puerto paralelo de una PC.

Para interpretar los comandos provenientes de la

PC y detener o reanudar la ejecución de la CPU se utilizó una máquina de estados. La misma permite detener la ejecución sólo en los ciclos de reloj adecuados, decodificación y lectura de operandos, y manejar la ejecución paso a paso.

Para permitir que el número de *breakpoints*y *watchpoints*fuera configurable sin alterar la cantidad de registros de entrada/salida del periférico, se implementó un registro selector de *breakpoints*y otro de *watchpoints*. Esto permite al periférico implementar 1 a 255 elementos manteniendo la estructura de registros.

3. SOFTWARE COMPLEMENTARIO

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 [10]. El mismo 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 dicho microcontrolador.

4. HARDWARE COMPLEMENTARIO

El microcontrolador en cuestión 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

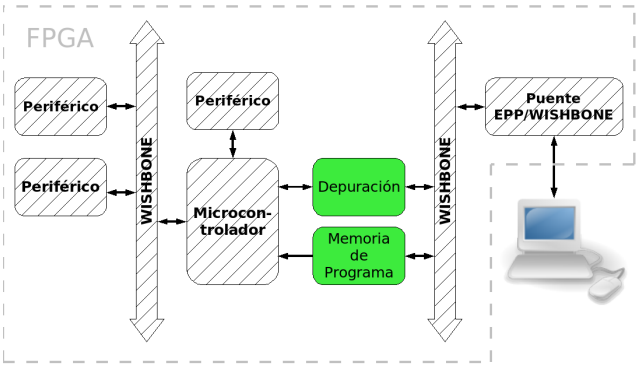

Fig. 1. Diagrama de conexiones de los bloques de hardware involucrados.

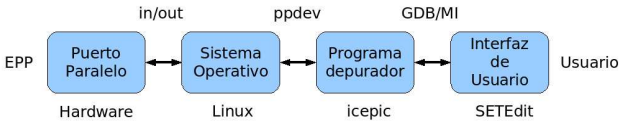

Fig. 2. Flujo de datos dentro de la computadora.

la FPGA.

La Fig. 1 ilustra la interconexión entre los distintos componentes de hardware antes mencionados, los bloques ilustrados con relleno sólido corresponden a los desarrollos descriptos en 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.

5. 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.
- Detectar desbordes en la pila y detener la ejecución cuando esto sucede.

El periférico insumió 16 direcciones de entrada salida:

- 2 para selección de comando y lectura de resultados y/o estado.
- 4 para los *breakpoints*: 1 de selección, 2 de dirección y otro de estado.
- 3 para acceder a los registros de memoria de datos: dirección, banco y valor.
- 4 para inspeccionar y/o modificar el *stack*y el *stackpointer*.
- 3 para los *watchpoints*: 1 de selección, 1 de habilitación y modo y otro de dirección.

Se requirió implementar 8 comandos para el control del periférico:

- STOP: detener la ejecución.
- RUN: reanudar la ejecución.
- STEP: ejecutar sólo un paso de programa.
- RESET: reiniciar el microcontrolador.
- FRESET: reiniciar todo el sistema.
- GPCL: leer la parta baja del contador de programa.
- GPCH: leer la parta alta del contador de programa.

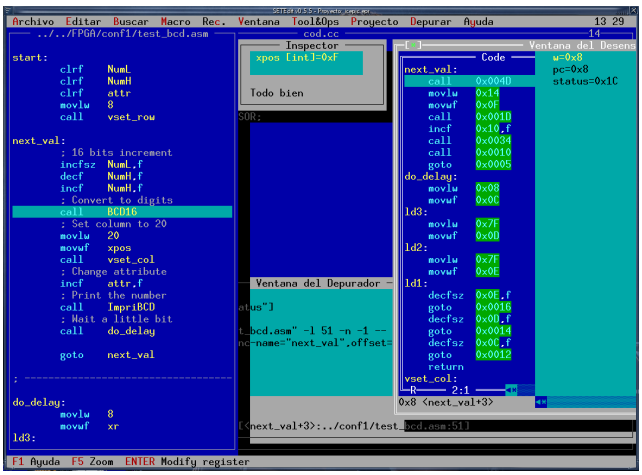

Fig. 3. Sesión de depuración.

- CGW: leer el registro acumulador (W).

La Fig. 3 muestra una sesión de depuración utilizando SETEdit. En la misma se observa el código fuente del programa, el código desensamblado y una ventana utilizada para monitorear el valor de una variable.

Este desarrollo fue verificado en hardware utilizando FPGAs *Spartan* 3 de Xilinx. Para la síntesis se utilizó el programa XST 8.2.02i (I.33).

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, de aproximadamente 32 *slices*, y el puente que conecta el puerto paralelo de la PC con el bus WISHBONE, de aproximadamente 31 *slices*. Esto corresponde al 9,7 % de una *Spartan* 3 200 o bien sólo un 1,3 % de una *Spartan* 3 1500.

6. 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 un 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.

7. REFERENCIAS

[1] "8 bits cmos eeprom microcontroller," http://-ww1.microchip.com/downloads/en/devicedoc/-30445c.pdf, Microchip Technology Inc.

[2] J. Salvador E. Tropea, "Microcontrolador compatible con pic16c84, bus wishbone y video," in *FPGA Based Systems*. Mar del Plata: Surlabs Project, II SPL, 2006, pp. 117–122.

[3] S. E. Tropea, "Microcontrolador compatible con pic16c84, descripción de hardware," http://-utic.inti.gov.ar/publicaciones/jornadasINTI2007/-pic.pdf, 6tas Jornadas de Innovación y Desarrollo - INTI, 2007.

[4] Salvador Eduardo Tropea, Diego Javier Brengi, and Juan Pablo Daniel Borgna, "FPGAlibre: Herramientas de software libre para diseño con FPGAs," in *FPGA Based Systems*. Mar del Plata: Surlabs Project, II SPL, 2006, pp. 173–180.

[5] INTI Electrónica e Informática *et al.*, "Proyecto FPGA Libre," http://fpgalibre.sourceforge.net/.

[6] Debian, "Sistema operativo Debian GNU/Linux," http://www.debian.org.

[7] "GNU project," http://www.gnu.org/.

[8] Silicore and OpenCores.Org, "Wishbone system-on-chip (soc) interconnection architecture for portable ip cores," http://prdownloads.sf.net/fpgalibre/-wbspec_b3-2.pdf?download.

[9] T. B. Tropea S., "Microcontrolador compatible con pic16c84, bus wishbone y video," in *FPGA Based Systems*. Mar del Plata: Surlabs Project, II SPL, 2006, pp. 257–264.

[10] Salvador E. Tropea *et al.*, "SETEdit, un editor de texto amigable," http://setedit.sourceforge.net.