Entorno de desarrollo y depuración de aplicaciones basadas en el microcontrolador soft-core PicoBlaze™

Garrigós-Guerrero J., Fernández-Conesa, J., Martínez-Álvarez J., Toledo-Moreo J.

Universidad Politécnica de Cartagena, Murcia, España,
javier.garrigos@upct.es, jjavier.martinez@upct.es, javier.toledo@upct.es
http://www.upct.es

Resumen. En este trabajo se presenta PicoIDE, un entorno de desarrollo y depuración de aplicaciones para el core de microcontrolador de 8 bits PicoBlaze™, distribuido de forma gratuita por Xilinx Inc., que completa el conjunto de herramientas proporcionado por esta empresa. Además, PicoIDE ha sido estructurado de forma modular para que sea fácilmente adaptable a las nuevas versiones del microcontrolador modificadas por el usuario, utilizando los procedimientos descritos. Para ello, está previsto que su código fuente se distribuya bajo la Licencia Pública General GNU. La funcionalidad de PicoIDE es verificada desarrollando diferentes aplicaciones sobre PicoBlaze™ y adaptándolo a arquitecturas modificadas del microcontrolador.

1 Introducción

La utilización de microprocesadores y/o microcontroladores incrustados en dispositivos programables, junto con la lógica específica de la aplicación, es una estrategia común en un gran número de aplicaciones. Estos sistemas en un chip (SoC, *System on a Chip*) [1] permiten integrar la flexibilidad y la facilidad de uso de un procesador con las grandes prestaciones obtenidas al implementar un algoritmo con lógica específica.

El número de arquitecturas de microcontroladores y microprocesadores disponibles para dispositivos programables, en forma de macros tanto *soft* como *hard* es numeroso. En aplicaciones con grandes requisitos de cómputo pueden utilizarse FPGAs que soportan cores de los procesadores PowerPC™, MicroBlaze™ o ARM™, entre otros. Sin embargo, en aplicaciones menos exigentes es posible recurrir a pequeños microcontroladores que, a pesar de sus limitadas prestaciones, proporcionan varias ventajas. Entre ellas destacan, principalmente, el reducido consumo de área en la FPGA, la sencillez y facilidad de uso, y el aprovechamiento de la experiencia del diseñador, puesto que con frecuencia soportan juegos de instrucciones compatibles con otros microcontroladores de amplia utilización en la industria. Cores de microcontroladores PIC, 8051, Z80, o 6502 entre otros, existen tanto en versiones comerciales como gratuitas.

Entre los procesadores más modestos, una de las arquitecturas más interesantes es el microcontrolador PicoBlaze™, que ha sido desarrollado por la compañía Xilinx Inc. para una implementación óptima sobre diferentes tipos de FPGAs. En este trabajo se desarrolla un entorno de desarrollo y depuración de aplicaciones para el PicoBlaze versión CPLD, especialmente desarrollada para que sea fácilmente adaptable a otras versiones del microprocesador existentes o elaboradas por el usuario. Nuestro depurador se ha desarrollado utilizan-

do el lenguaje Java™, lo que aumentará su portabilidad a distintos sistemas operativos.

El resto de este artículo se organiza como sigue. En la sección 2 se expone la arquitectura del microcontrolador PicoBlaze para CoolRunner que constituye la base para el desarrollo de nuestro depurador. En la sección 3 se aborda la descripción de la herramienta PicoIDE, su funcionalidad y el procedimiento para modificarla. En la sección 4 detallan los resultados obtenidos al implementar sistemas basados en diversas versiones de PicoBlaze sobre diferentes FPGAs. Finalmente, la sección 5 resume las conclusiones y principales aportaciones.

2 El core de microcontrolador PicoBlaze™

El PicoBlaze es un core de microcontrolador de 8 bits que Xilinx suministra de forma gratuita en forma de código VHDL sintetizable, estando su utilización libre de derechos de propiedad intelectual sobre FPGAs de esta empresa [2]. Actualmente existen cuatro versiones, cada una de ellas optimizada para una arquitectura específica de FPGA [2-5]. En la Tabla 1 se resumen las principales características de cada una de ellas. Aparte del código VHDL del microcontrolador, cada distribución incluye algunas utilidades para modificar el core y una pequeña UART, además del necesario programa ensamblador, ejemplos de utilización y plantillas.

Característica	PicoBlaze para: Spartan-3, Virtex-II, Virtex-II Pro	PicoBlaze para: Virtex-II, Virtex-II Pro	PicoBlaze para: Virtex,VirtexE, Spartan-II, Spartan-IIE	PicoBlaze para: CoolRunner-II
Memoria de programa	1024	1024	256	256
Tamaño instrucción	18-bits	18-bits	16-bits	16-bits
Juego de instrucciones	57	49	49	49
Nº de registros 8-bits	16	32	16	8
Profundidad de pila	31	31	15	4
Ensamblador	KCPSM3	KCPSM2	KCPSM	ASM
Área	96 (S-3 slices)	84	76 (S-IIE slices)	212 (XC2C256)
Prestaciones (MIPS)	44 (S3) 100 (V-IIPro)	40-70	37 (S-IIE)	21
Memoria *Scratch Pad*	64 bytes	-	-	-

Tabla 1. Comparación de características y prestaciones (Fuente: Xilinx Inc).

Todas las versiones de PicoBlaze proporcionadas por Xilinx poseen una arquitectura de 8 bits, mientras que el número de instrucciones es de 49 en todas salvo en la más reciente adaptación, para la familia Spartan-3™ de FPGAs, que incluye un banco de memoria auxiliar denominado *Scratch Pad* (ver resumen de características en Tabla 1) y 8 nuevas instrucciones. La Tabla 2 muestra de forma resumida el juego de instrucciones de este microcontrolador.

De todas las versiones, únicamente el PicoBlaze para la familia de CPLDs CoolRunner™ ha sido específicamente diseñado para ser fácilmente modificable por el usuario [5]. Para conseguirlo, Xilinx distribuye este microcontrolador en forma de código VHDL de alto nivel (descripción *comportamental*) independiente del dispositivo, mientras que el resto de versiones utilizan primitivas de bajo nivel que lo hacen fuertemente dependiente de

la arquitectura de la FPGA. Asimismo, el programa ensamblador, una aplicación para DOS denominada asm.exe, se distribuye también en forma de código fuente (asm.cpp), a diferencia de las diferentes versiones del ensamblador KCPSM utilizado en las otras implementaciones del microcontrolador, que sólo se distribuyen como ejecutable.

Control de Programa		Lógicas		Aritméticas
JUMP dir		LOAD sX,cte		ADD sX,cte
JUMP Z,dir		AND sX,cte		ADDCY sX,cte
JUMP NZ,dir		OR sX,cte		SUB sX,cte
JUMP C,dir		XOR sX,cte		SUBCY sX,cte
JUMP NC,dir	*	TEST sX,cte	*	COMPARE sX,cte
CALL dir		LOAD sX,sY		ADD sX,sY
CALL Z,dir		AND sX, sY		ADDCY sX, sY
CALL NZ,dir		OR sX, sY		SUB sX, sY
CALL C,dir		XOR sX, sY		SUBCY sX, sY
CALL NC,dir	*	TEST sX, sY	*	COMPARE sX, sY
RETURN				
RETURN Z		Desplazamiento/Rotación		Almacenamiento
RETURN NZ		SR0 sX	*	FETCH sX,sdir
RETURN C		SR1 sX	*	FETCH sX,(sY)
RETURN NC		SRX sX	*	STORE sX,sdir
		SRA sX	*	STORE sX,(sY)
		RR sX		
Entrada/Salida		SL0 sX		Interrupciones
INPUT sX,puerto		SL1 sX		RETURNI ENABLE
INPUT sX,(sY)		SLX sX		RETURNI DISABLE
OUTPUT sX,puerto		SLA sX		ENABLE INTERRUPT
OUTPUT sX,(sY)		RL sX		DISABLE INTERRUPT

Tabla 2. Juego de instrucciones del PicoBlaze. Todas las instrucciones se ejecutan en dos ciclos de reloj. Las instrucciones con (*) sólo están disponible en la versión KCPSM3 del microcontrolador.

El PicoBlaze para CoolRunner CoolRunner (que denominaremos a partir de ahora PicoBlaze-CR) es por lo tanto la adaptación más portable de este microcontrolador sobre diferentes arquitecturas de FPGA, como veremos más adelante. Además, dado que el microcontrolador propiamente dicho y el programa ensamblador correspondiente pueden ser configurados incrementando o limitando su funcionalidad (juego de instrucciones, memoria de programa, etc.), ambos pueden adaptarse de forma óptima a diferentes problemas. Por todo ello, hemos seleccionado esta versión del PicoBlaze para utilizarla como base de varias aplicaciones actualmente en desarrollo.

Sin embargo, hemos de tener en cuenta que, al ser independiente de la tecnología de implementación debido a que por no utilizar primitivas de bajo nivel de una arquitectura de FPGA concreta, esta versión del PicoBlaze es la menos optimizada, y por tanto la que presenta menores prestaciones. En la Tabla 3 se muestra una comparativa de la síntesis del microcontrolador sobre diferentes dispositivos. Mientras que las diferencias en el tamaño del circuito son poco significativas, las variaciones en la frecuencia máxima de funcionamiento son sustanciales, debido principalmente a la tecnología de integración del dispositivo, por lo que las prestaciones obtenidas van desde aproximadamente 25 MIPS hasta los 57,5 MIPS.

Cuando comparamos implementaciones de este microcontrolador con las versiones del mismo optimizadas para cada arquitectura (Tabla 1), la mayor penalización se obtiene en el

área ocupada por el circuito, que es en todas las arquitecturas alrededor de un tercio del área ocupada por la versión independiente de la tecnología.

Característica	Virtex-II,	Virtex	Spartan-IIE	CoolRunner-II
Dispositivo	2v40cs144-6	xcv50-bg256-6	xc2s200pq208-7	XC2C384-7-FT256
Área	254 slices	268 slices	265 slices	250 macroceldas
Velocidad	115,493 MHz	50,906 MHz	60,082 MHz	65,7 Mhz

Tabla 3. Implementación del PicoBlaze para CPLDs (PicoBlaze-CR) sobre diferentes arquitecturas.

A pesar de la desventaja en área y prestaciones, el PicoBlaze-CR resulta útil en ciertas aplicaciones en las que la velocidad de procesamiento de 25 a 50 MIPS es suficiente. Ello es debido a que esta versión admite la posibilidad de modificar el microcontrolador con gran facilidad para eliminar lógica que no utilicemos en una aplicación, como las interrupciones, determinadas instrucciones, la pila, etc., con lo que el área final puede reducirse considerablemente. Por supuesto, usualmente también resulta conveniente añadir instrucciones nuevas, como operaciones de multiplicación por hardware, o incrementar los recursos (registros, pila, etc.).

3 El entorno integrado de desarrollo PicoIDE

La carencia de una herramienta de depuración adecuada complica el proceso de desarrollo de aplicaciones y limita la aplicabilidad del PicoBlaze. Hasta donde conocemos, el único entorno de desarrollo basado en GUI existente para este microcontrolador es el distribuido por la compañía Mediatronix BV, denominado pBlazeIDE [6]. A pesar de ser gratuita, esta aplicación tiene el inconveniente de que su código fuente no está a disposición del usuario, por lo que éste no puede adaptar el software a versiones modificadas del PicoBlaze para tareas específicas, una de las grandes ventajas del microcontrolador.

Por el contrario, PicoIDE es un entorno integrado de desarrollo de aplicaciones para el PicoBlaze-CR, que además ha sido específicamente diseñado para ser fácilmente adaptable a versiones modificadas de este microcontrolador. Desarrollado en Java™ para aumentar su portabilidad, está previsto que su código fuente sea distribuido bajo la Licencia Pública General GNU [7] una vez finalice la fase de pruebas.

La herramienta tiene principalmente dos utilidades integradas en una misma interfaz gráfica de usuario o GUI (del inglés *Graphical User Interface*), la primera, para el desarrollo de la aplicación, y la segunda, para su depuración, cuya funcionalidad se describe en los siguientes epígrafes.

3.1 El modo de Desarrollo

El modo Desarrollo proporciona un *front-end* gráfico para el programa ensamblador de MS-DOS que Xilinx distribuye. Incluye un editor de textos para escribir y formatear el código fuente y una ventana de texto para mostrar mensajes (ver figura 1). Una vez escrito el programa, se selecciona el botón (o comando de menú) adecuado para ensamblarlo, lo que se traduce en una llamada al ensamblador asm.exe de Xilinx. La aplicación entonces redirige la salida estándar del asm.exe a la consola de mensajes, lo que permite visualizar

más cómodamente los errores o alarmas generados durante el proceso de ensamblado.

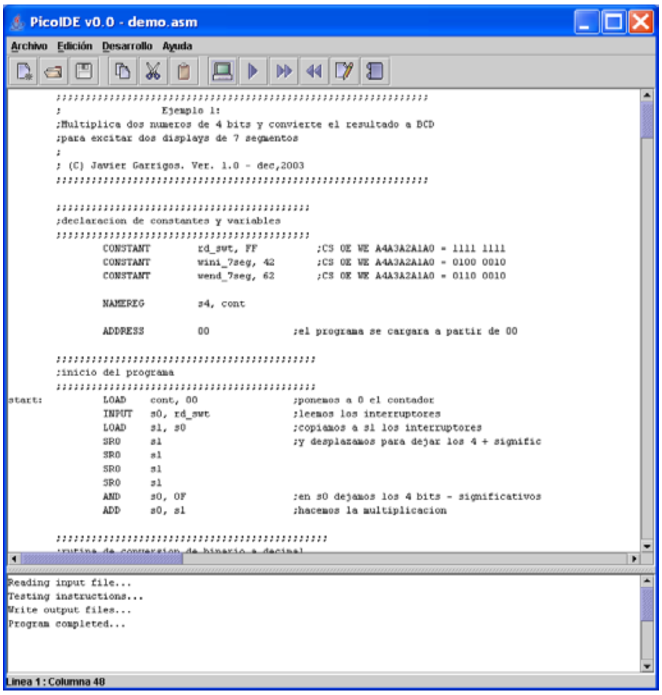

Fig. 1. Ventana principal del PicoIDE en Modo Desarrollo.

En un principio nos planteamos la posibilidad de desarrollar nuestro propio programa ensamblador, lo que nos permitiría adaptarlo convenientemente para cada arquitectura modificada del PicoBlaze-CR. Sin embargo, a diferencia de las otras versiones, Xilinx proporciona el código fuente del ensamblador para el PicoBlaze-CR, e incluso proporciona instrucciones sobre cómo modificar el código para un nuevo juego de instrucciones y recompilar el ensamblador. Por tanto, finalmente se optó por incorporar llamadas a este programa dentro de nuestro entorno, en lugar de producir una aplicación similar.

3.2 El modo de Depuración

El modo Depuración permite ejecutar cualquier programa línea a línea, ejecutar un grupo de líneas determinado, o en forma continua. Simultáneamente se muestra la línea en ejecución, el contenido de los registros internos y el de los puertos de entrada/salida. Para una visualización más ordenada, en este modo de funcionamiento, al que se pasa tras seleccionar cualquiera de las opciones de ejecución del programa, la interfaz gráfica se ha dividido en tres paneles, como se muestra en la Figura 2. El panel principal muestra el código desensamblado resaltando la instrucción que se ejecuta en cada instante; el panel situado a su derecha muestra los contenidos de los registros de usuario, así como el contador de programa y los *flags* de estado del microcontrolador; por último, el panel inferior muestra el contenido de los 256 puertos de entrada/salida (todos los valores se muestran en hexadecimal).

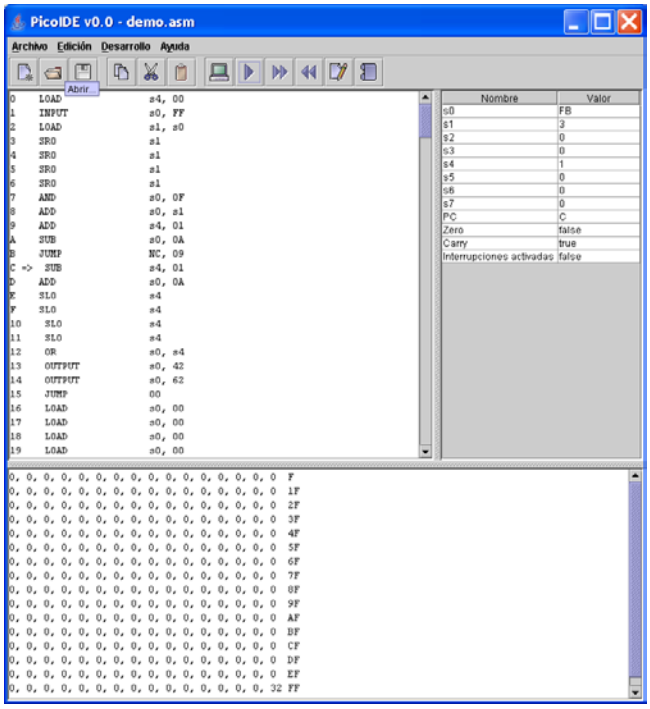

Fig. 2. Ventana principal del PicoIDE en Modo Depuración.

3.3 Estructura de la aplicación

La aplicación PicoIDE se ha desarrollado utilizando el lenguaje Java™, habiéndose estructurado en seis clases, tal y como se muestra en la Figura 3. En la clase principal, *PicoIDE*, se define la interfaz gráfica de usuario (ventana principal, barra de menús, etc.). La gestión de los eventos de usuario dentro del GUI la realiza la clase *Eventos*, que se encarga de llamar a los métodos adecuados de las clases *PanelDesarrollo* o *PanelDepuración* según el modo de funcionamiento en el que estemos trabajando.

Por su parte, *PanelDesarrollo* incluye los métodos que implementan el editor de textos y controla la gestión de eventos propios (posición del cursor, número de línea y columna, banderas de modificación de fichero, etc.). Por otro lado, *PanelDepuración* gestiona la parte gráfica del motor de depuración. Se encarga por ejemplo de mostrar los cambios en el contenido de los registros o en la E/S como consecuencia de la ejecución de una nueva instrucción.

El motor de depuración propiamente dicho lo constituyen las clases *MotorDebugger* y *PicoInstrucción*. La primera lee el código máquina correspondiente al ensamblado de un programa y genera un objeto *PicoInstrucción* para cada una de las instrucciones. Cada *PicoInstrucción* realiza el proceso de desensamblado, obteniendo el mnemónico y los argumentos correspondientes a una instrucción en hexadecimal. Con estos datos, *MotorDebugger* procede a la ejecución propiamente dicha de la instrucción. Finalmente, *PanelDepuración* actualizará la interfaz gráfica para mostrar los nuevos resultados al usuario.

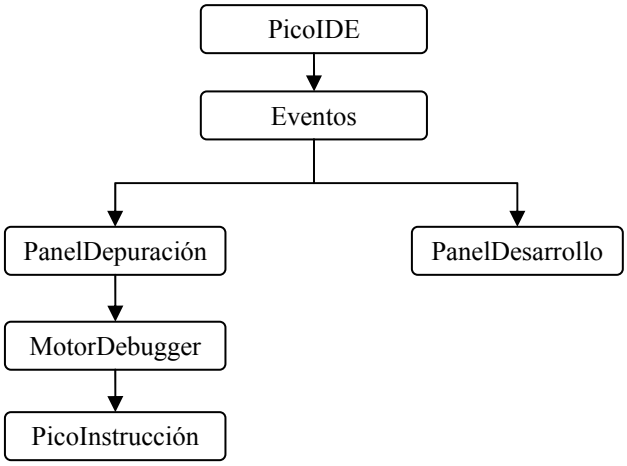

Fig. 3. Diagrama de flujo de la aplicación PicoIDE.

4 Aplicaciones y resultados obtenidos

El entorno de desarrollo PicoIDE está siendo verificado en la actualidad en varios campos. En primer lugar, ha sido utilizado para el desarrollo de diferentes aplicaciones dentro de los programas de prácticas de algunas asignaturas impartidas en las titulaciones de Ingeniero en Telecomunicación e Ingeniero en Automática y Electrónica Industrial en la Universidad Politécnica de Cartagena.

Por otro lado, un PicoBlaze y un PicoIDE modificados (con mayor memoria de programa y nuevas instrucciones aritméticas como la multiplicación) están siendo utilizados para implementar el mecanismo de aprendizaje por *backpropagation* de una red neuronal para reconocimiento de hablantes. Debido a la gran capacidad de cálculo necesaria, la red será implementada directamente a nivel RTL en una FPGA. Este hecho, sin embargo, le resta parte de su flexibilidad, por lo que se ha optado por incluir un procesador incrustado (un PicoBlaze modificado) en la FPGA que realizará ciertas tareas de supervisión, principalmente a nivel de comunicaciones, control y adaptación/modificación de la red neuronal en caso de ser necesario.

5 Conclusiones

En este trabajo se han revisado algunas de las soluciones más difundidas para la realización de microcontroladores en circuitos programables, relacionando cada alternativa con su ámbito de aplicación. En particular, nos hemos centrado en las soluciones basadas en el PicoBlaze™ de Xilinx Inc. Hemos ampliado el conjunto de herramientas de desarrollo para este microcontrolador mediante PicoIDE, un entorno de diseño y depuración de aplicaciones altamente configurable que simplifica el proceso de desarrollo con este procesador. La funcionalidad del entorno de desarrollo ha sido verificada con el diseño de varias aplicaciones por parte de nuestros alumnos durante las prácticas de laboratorio.

Por último destacamos que PicoIDE está desarrollado en Java™, lo que aumenta su portabilidad entre diferentes plataformas. Además, superada la fase de pruebas, está previsto que sea distribuido libremente bajo la Licencia Pública General GNU [7]. La distribución del código fuente permitirá que un usuario pueda modificar el depurador para adaptarlo a cualquier versión modificada del PicoBlaze desarrollada para una aplicación específica, lo que convierte a PicoIDE en una herramienta valiosa para el desarrollo de sistemas basados en este microcontrolador.

Agradecimientos

Este trabajo ha sido financiado en parte por el Ministerio de Ciencia y Tecnología, TIC 2003-09400-C04-02.

Referencias

[1] Keating, M., Bricaud, P.: Reuse Methodology Manual, Second Edition. Design Reuse Partnership - Synopsys Inc. and Mentor Graphics Corp. Kluwer Academic Publishers (1999).
[2] Xilinx, Inc.: PicoBlaze 8-bit Embedded Microcontroller User Guide (UG129). http://www.xilinx.com (2004).
[3] Xilinx, Inc.: PicoBlaze 8-Bit Microcontroller for Virtex-II Series Devices (XAPP627). http://www.xilinx.com (2004).
[4] Xilinx, Inc.: PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices (XAPP213). http://www.xilinx.com (2004).
[5] Xilinx, Inc.: PicoBlaze 8-bit Microcontroller for CPLD Devices (XAPP387). http://www.xilinx.com (2004).
[6] Mediatronix BV: pBlazeIDE Documentation and code. http://www.mediatronix.com /pBlazeIDE.htm (2004).
[7] Free Software Foundation, Inc.: GNU General Public License. http://www.gnu.org /licenses/gpl.html