Tutorial Xilinx MicroBlaze

Aguayo E, González I. y Boemo E.

Escuela Politécnica Superior, Universidad Autónoma de Madrid, España
{estanislao.aguayo, ivan.gonzalez, eduardo.boemo}@uam.es
http://www.eps.uam.es

Resumen. Este artículo tiene como objetivo servir de guía para diseños en FPGAs donde se requiera capacidad de procesamiento, sin recurrir a un procesador externo. Esta tarea es la principal aplicación para el *soft-processor* Xilinx MicroBlaze. Este bloque está optimizado para ejecutar código C. Además del núcleo procesador, existen una gran variedad de periféricos; adicionalmente, pueden utilizarse los recursos de la FPGA para agregar bloques *custom* definidos en VHDL. Este tutorial abarca desde los detalles arquitecturales hasta las herramientas de diseño.

1. Introducción

Este artículo se centra sobre el procesador embebido MicroBlaze, creado por Xilinx para sus FPGAs Spartan y Virtex. Aún siendo un *soft-processor*, el usuario no tiene acceso a los detalles VHDL del circuito. Sólo puede analizarse a partir de la documentación del fabricante y de los resultados experimentales. Aún así, es una herramienta muy potente para desarrollar proyectos relacionados con arquitecturas paralelas, codiseño y control; y en general, en toda investigación sobre software, puesto que permite una comparación inmediata con otras arquitecturas y metodologías de desarrollo.

Este *tutorial* se divide en tres partes, la primera analiza la arquitectura que el fabricante ha utilizado para la realización del diseño, una segunda parte se encarga de la descripción del procedimiento para el desarrollo y utilización de código para este procesador y por último, el desarrollo de periféricos propios para este procesador.

2. Arquitectura del procesador

El procesador MicroBlaze de 32 bits ha sido desarrollado con unos requisitos muy rígidos de ocupación y prestaciones [3], debido a la limitación impuesta por los recursos disponibles en una FPGA. Por ello, resulta algo limitado, si se lo compara con otros micros *hardwired* equivalentes. Soluciones arquitectónicas, como la supeescalaridad o la supersegmentación no resultan adecuadas en un entorno lógico reprogramable, cuando el objetivo es un diseño compacto. Del mismo modo, la frecuencia de operación final se ve limitada por los altos retardo de interconexión de la FPGA.

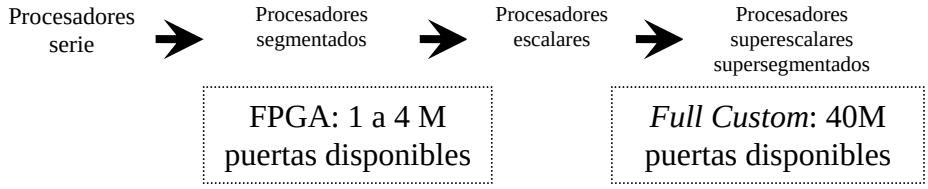

Figura 1: Situación del desarrollo de arquitecturas de procesadores.

a. Repertorio de Instrucciones

La restricción de espacio fuerza a un diseño sencillo, que encaja perfectamente con la idea de las arquitecturas tipo RISC (*Reduced Instruction Set Computer*), donde el limitado número de instrucciones permite simplificar la unidad de descodificación. En el caso de MicroBlaze, el número total de instrucciones que soporta son 87, si se consideran diferentes las instrucciones que operan con valores inmediatos de aquellas que realizan la misma operación con registros. Adicionalmente, cada instrucción se ha elegido para que el tamaño de la ALU también sea reducido. Las instrucciones que requieran un procesamiento complejo habrán de realizarse en un hardware específico, diseñado utilizando los restantes recursos de la FPGA. El mecanismo de conexión de estos co-procesadores con Microblaze se especifica en el protocolo FSL (*Fast Simple Link*) [10].

b. Pipeline

Una arquitectura RISC aumenta fácilmente su rendimiento por medio de la segmentación (*Pipelining*) [9]. En el caso de MicroBlaze, el número de etapas de *pipeline* [Referencia] es 3, ejecutando una instrucción por ciclo de reloj. Como contrapartida, la segmentación debe incorporar mecanismos para evitar los problemas relacionados con los saltos de programa. En un salto, la pila está llena de instrucciones que no corresponden con el flujo de ejecución. Estos riesgos están tratados por *hardware* en MicroBlaze: cada vez que se produce un salto, el *pipeline* se vacía cuando el salto se hace efectivo. En el repertorio se incluyen un cierto número de instrucciones de salto que permiten la ejecución de la instrucción que les precede, con el objetivo de reducir la penalización de vaciado. Esta técnica es conocida como *Delay Slots* [Referencia].

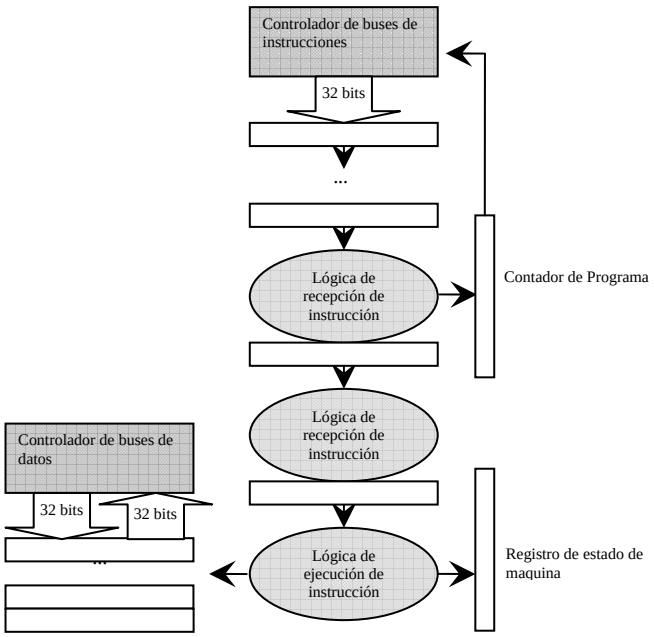

Figura 2: Esquema de bloques de MicroBlaze

c. Registros Internos y Caché

Las FPGAs actuales disponen de memoria distribuida con un tiempo de acceso corto, cuando son utilizadas por la lógica cercana. Este tipo de memoria es utilizada por MicroBlaze para materializar sus 32 registros internos, el contador de programa y el registro de estado. Este último sólo contiene el bit de acarreo, habilitación de cachés, indicador de estado de parada (*break*), error en el FSL y bit de excepción producida por división por cero. Además de estos registros internos, MicroBlaze utiliza un *buffer* de 16 instrucciones mapeado en los registros de desplazamiento SRL de los *slices*. Este recurso es fundamental para un buen rendimiento del procesador. Por ejemplo, en caso de que no disponer de multiplicador y/o divisor hardware, se aprovecha el tiempo que tarda en ejecutarse esta instrucción (32 ciclos) para tener preparadas las siguientes instrucciones.

La utilización de cachés es una práctica habitual en arquitecturas modernas; el diseño de MicroBlaze no es una excepción. Pero en este caso, la utilización y configuración de los tamaños de caché pueden ser fijados por el usuario. Para ello, existe el siguiente conjunto de opciones a la hora de configurar el procesador: habilitación de caché de instrucciones y/o datos, tamaño, rango de direcciones y tamaño de palabra, aunque esta ultima opción solo es valida para la caché de datos. Opciones no configurables por el usuario son tamaño de los bloques de caché, la política de sustitución de bloques o el grado de asociatividad de la caché.

Todas estas opciones se especifican antes de sintetizar el diseño. Las cachés son de tipo asociativo, por lo cual es necesario el calcular el número de bits de la dirección (*tag bits*) que especifican bloques contenidos en caché, mediante la ecuación (1).

Numero *tag* bits = $\log_2$(rango de memoria cacheable) - $\log_2$(tamaño de la caché) **(1)**

Las cachés utilizan los bloques de RAM fijos (BRAM) que contiene la FPGA. Sin embargo, códigos compactos también pueden mapearse en BRAM. Por lo tanto, la utilización de cachés será necesaria en aquellos casos donde el código o los datos utilizado por MicroBlaze residan fuera de la FPGA.

d. *Buses* del Sistema

MicroBlaze sigue el modelo de arquitectura Harvard [Referencia], donde datos e instrucciones son almacenados en memorias diferentes. De nuevo, gracias a la capacidad de reconfiguracion de las FPGAs, es posible configurar el sistema con diferentes opciones sobre los *buses*, pudiéndose reducir de este modo el tamaño final del sistema. MicroBlaze utiliza el estándar *CoreConnect* [5] creado por IBM, para conectar diferentes elementos en un circuito integrado. Un aspecto interesante es que *CoreConnect* permite reducir la carga capacitiva del *bus*, repartiéndola entre varios *buses*. Así, se consiguen mayores rendimientos, dado que los retardos de pistas globales son muy importantes en FPGAs. El estándar define tres tipos de *buses*, cada uno con unas características de velocidad y conectividad:

LMB: (*Local Memory Bus*): *Bus* síncrono de alta velocidad, utilizado para conectar periféricos y los bloques de memoria interna de la FPGA. Solo admite un maestro en su implementación para MicroBlaze y puede ser utilizado tanto para instrucciones como para datos. Este *bus* es compatible con el PLB (*Processor Local Bus*) incluido en el estándar *CoreConnect*, pero a diferencia de éste, el LMB no admite varios maestros ni tamaños de palabra diferentes a 32 bits.

OPB: (*On-chip Peripheral Bus*): *Bus* síncrono utilizado para conectar periféricos con tiempos de acceso variables. Soporta varios maestros y la conectividad de muchos periféricos es sencilla gracias a su identificación por multiplexación distribuida. Soporta transferencias de tamaño de palabra dinámico.

DCR: (*Device Control Register*): *Bus* síncrono diseñado para una conexión tipo anillo de periféricos con ancho de banda muy bajo. Solo soporta un maestro y está diseñado para minimizar el uso de lógica interna.

	LMB	OPB	DCR	FSL
Ancho de banda máximo	500 MB/s	167 MB/s	-	800 MB/s
Número de periféricos	1	Limitado por espacio en la FPGA	Limitado por espacio en la FPGA	16

Tabla 3: Comparación de buses de MicroBlaze

Además de estos tres *buses* existe otra alternativa para comunicarse con el procesador MicroBlaze. Se trata de un protocolo de comunicación denominado FSL (*Fast Simple Link*) mencionado anteriormente, que permite la comunicación con el procesador a través de sus

propios registros internos. El periférico actuaría a modo de co-procesador y la conexión se realiza de manera sencilla a través de unos registros de desplazamiento de 32 bits de ancho. La comunicación con estas FIFOs se realiza con dos instrucciones específicas del repertorio, que realizan las funciones de *push* y *pop* de estas memorias de desplazamiento. MicroBlaze soporta hasta 16 dispositivos conectados con este protocolo.

e. Interrupciones y excepciones

El procesador MicroBlaze contiene una línea de interrupciones, la cual al ser activada hace que el procesador ejecute una rutina de manejo de interrupciones que ha de ser especificada al compilador. Las excepciones se tratan de forma similar. Cuando una de ellas ocurre, se paraliza el procesamiento de instrucciones y se ejecuta una rutina de manejo de excepciones. En el caso de que el sistema necesite manejar más de una interrupción, será necesario la utilización de un periférico específico (*OPB Interrupt Controller*), que se encarga de multiplexar e identificar las diferentes fuentes de interrupción.

3. Utilización de la Herramienta EDK 6.1

La descripción de hardware en VHDL o Verilog no esta disponible para el usuario de FPGAs. Por ello, debe integrarse en el flujo estándar de diseño mediante una herramienta específica proporcionada por el fabricante. Esta herramienta, llamada EDK (*Embedded Design Kit*), incluye ficheros con descripciones en el formato .ngc de Xilinx, que se utilizan para describir diseños después de la etapa de síntesis [1].

EDK está compuesta por una serie de aplicaciones para la configuración hardware, la codificación y el desarrollo de periféricos específicos. Sin embargo, EDK por si sola, no es suficiente para construir y cargar un diseño en una FPGA [8]. Todas las tareas de emplazamiento y rutado, así la descarga a la FPGA deben ser realizadas dentro del ISE, la herramienta propia de Xilinx. Así, ambos paquetes son necesarios para el diseño de un sistema MicroBlaze.

La configuración de un sistema MicroBlaze con la herramienta EDK requiere el manejo de varios archivos diferentes.

a. **Archivo de configuración de hardware**

Este archivo con extensión .mhs (*microprocessor hardware specification*) contiene las especificaciones de los puertos del sistema [4]. Cada puerto ha de ser especificado con un nombre, tipo (entrada o salida) y el número de bits que componen el puerto. Así, por ejemplo, si configuramos el MicroBlaze para almacenar código en una memoria externa a la FPGA, se especificaran en este archivo las diferentes señales que desde el controlador han de salir de la FPGA al exterior para controlar esta memoria. También se especifican las opciones de configuración del procesador, tales como los *buses* que componen el sistema. Otros parámetros a especificar son la utilización de recursos *hardware* de la FPGA, como por ejemplo divisores hardware (en caso de querer utilizar recursos de la FPGA para esta tarea), habilitación de cachés y tamaño de las mismas, y por último, habilitación y opciones del modo de solución de errores (*debug*).

Este archivo contiene las instanciaciones de los periféricos del sistema, así como la configuración de los mismos. Entre estas opciones figura el rango de memoria en el cual

están mapeados estos periféricos. Estas pueden asignarse en un archivo de opciones hardware .mpd (*microprocessor peripheral description*) y descargar de contenido al archivo de configuración de hardware. La herramienta EDK contiene un editor para este archivo, de manejo muy sencillo.

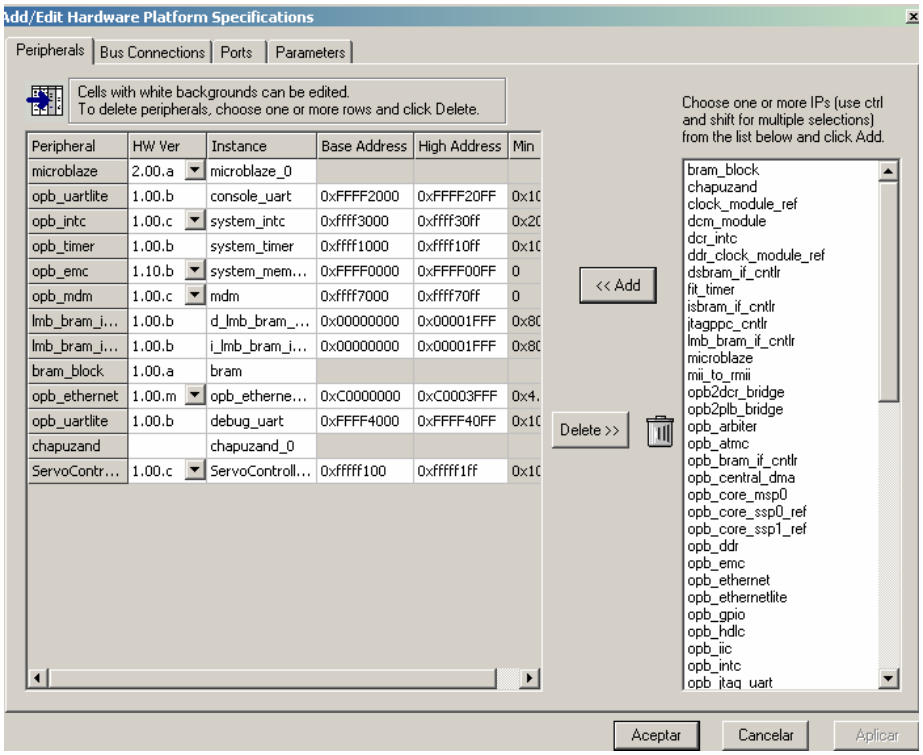

Figura 4: Captura de pantalla del editor del fichero de descripción de hardware en la herramienta EDK 6.1.

b. Archivo de Descripción del Software

Este archivo con extensión .mss (*microprocessor software specification*) contiene las opciones de compilación del software del sistema [4]. Incluye la especificación del modo de compilación del código, asignación de librerías a periféricos, especificación del método de solución de errores, y otras opciones del compilador. Este archivo depende explícitamente del archivo de descripción de hardware, ya que debe de especificarse el controlador para cada uno de los periféricos instanciados. De igual forma, si el dispositivo donde se va a cargar el diseño utiliza recursos hardware específicos, por ejemplo un multiplicador, es en este archivo donde se debe indicar al compilador que utilice este recurso hardware.

Los controladores de los periféricos se encuentran en otros archivos específicos. Los .mdd (*microprocessor driver definition*) contienen el código necesario para el control de cada periférico. En la instanciación del controlador se debe especificar la versión del

mismo, en caso de que existan varias y el nivel de abstracción con el que se va a utilizar el periférico en el código.

c. **Código**

Los dos archivos anteriores son de una sintaxis muy simple, puesto que se trata de especificar opciones e instanciar periféricos y sus controladores. Una vez compilados e introducidos los periféricos a utilizar en el sistema MicroBlaze, solo queda agregar el código del programa. Estos archivos pueden estar en C, C++, o ensamblador. El compilador para MicroBlaze está basado en el *gcc* desarrollado por el proyecto GNU [Referencia]. No existe ninguna restricción a la hora de escribir código para un sistema MicroBlaze. Al ser un procesador de 32 bits soporta todos los tipos de datos.

En el caso que el código del sistema esté diseñado para ser almacenado en diferentes zonas del mapa de memoria, por ejemplo código repartido entre la memoria interna y externa de la FPGA, es necesario escribir un *Linker Script* conteniendo las instrucciones necesarias para que se compile cada parte del código en el rango de direcciones correspondiente.

4. Construcción de Periféricos

La creación de un periférico se desarrolla en varias fases bien definidas, y para ello es necesario utilizar un sintetizador VHDL junto con otras herramientas incluidas en la herramienta EDK [2].

1. Descripción Lógica del Periférico

Esta primera fase consiste en describir el periférico en un lenguaje de descripción de hardware. Dependiendo de la forma de comunicación y la velocidad, se elegirá un tipo el *bus* al que se conectará el periférico. O bien, puede ser conectado directamente a los registros internos de MicroBlaze con el protocolo FSL. Para algunas de estas conexiones a los buses, *Xilinx* ofrece un *soft-macro* que maneja el protocolo necesario para la comunicación con el procesador. Estos bloques, denominados IPIF, ahorran al diseñador el estudio detallado de los diferentes protocolos.

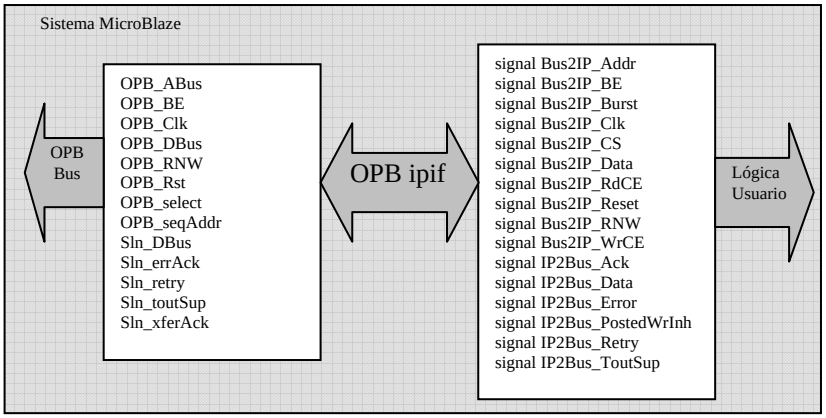

Figura 5: Conexión de la lógica de un periférico con el bus OPB utilizando un módulo OPB ipif.

2. Compilación del Periférico

La inclusión de un periférico en la librería se realiza a través de una aplicación, denominada *Import Peripheral Wizard*, incluida en EDK, que se encarga de realizar una compilación del diseño. Las librerías que se han utilizado en el diseño deben ser especificadas a esta herramienta. Esta operación se puede realizar seleccionando directamente las librerías en el menú de esta aplicación o bien mediante un archivo con extensión .pao (*peripheral analyze order*). Una vez compilado el diseño satisfactoriamente, se asignan las señales de los *buses* con las del periférico y posteriormente se especifican las diferentes opciones de configuración del periférico. Para almacenar los valores por defecto de estas opciones se requiere la creación de un archivo .mpd (*microprocessor perepheral definition*) [4]. Una vez compilado, EDK incluye al periférico dentro de la lista de posibles bloques a utilizar por MicroBlaze y solo habrá que instanciarlo para incluirlo en el diseño.

5. Problemas más comunes

A continuación se enumera una serie de puntos que deben verificarse en caso de una implementación fallida:

- **5.1** Comprobar que todas las señales de reloj (*buses*, periféricos y procesador) están conectadas al mismo puerto.
- **5.2** Comprobar el nivel de *reset* activo y cerciorarse de que coincide con el que proporciona la placa utilizada.
- **5.3** Comprobar la correspondencia entre los puertos del archivo .mhs y la asignación de *pines* en el archivo .ucf.

5.4 Asegurar la selección del nivel de abstracción (*inteface level*) con que se va a utilizar cada periférico. Además utilizar en el código las librerías correspondientes a este nivel de abstracción.

5.5 Comprobar las opciones de generación de la cadena de bits de configuración (*bitstream*) en el archivo *bitgen.ut* dentro del directorio del proyecto */etc* y asegurarse que estas opciones son las requeridas por la FPGA en la placa utilizada.

5.6 Para primeras implementaciones es fundamental el uso de la herramienta de debug, XMD. Esta proporcionara información del estado del sistema, sobre todo si está cargado correctamente en la FPGA.

5.7 Al utilizar UART y conectarla al PC a través de *pines* de la FPGA, asegurarse de la utilización de un conversor de niveles para protocolo RS-232.

5.8 A veces, al compilar el código, ocupa más espacio de memoria que el disponible. Aunque no se genera un mensaje de error en el flujo de implementación de la herramienta, el sistema no arranca.

5.9 A la hora de compilar un periférico, incluir las librerías utilizadas en el diseño en el orden de compilación correcto, en caso de utilizar la herramienta para asistir a esta compilación incluida en EDK, *Import Peripheral Wizard*.

6. Conclusiones

En este artículo se han introducido los conceptos básicos de la arquitectura y metodología de diseño con el procesador "blando" *MicroBlaze* de Xilinx. Con esta herramienta queda abierta una nueva frontera en la implementación de aplicaciones basadas en FPGAs.

Agradecimientos

Este trabajo ha sido financiado por los proyectos 07T/0052/2003-3 de la Consejería de Educación de la Comunidad de Madrid y TIC2001-2688-C03-03 del Ministerio de Ciencia y Tecnología de España.

Referencias

1. Xilinx Inc., "Embedded System Tool Guide". (Septiembre 2003)
2. Xilinx Inc., "User Core Templates Reference Guide". (Agosto 2003)
3. Xilinx Inc., "MicroBlaze Hardware Reference Guide". (Marzo 2002)
4. Xilinx Inc., "MicroBlaze Software Reference Guide". (Marzo 2002)
5. IBM, "The CoreConnect Bus Architecture". (1999).
6. Eduardo Sanchez, "*Reconfigurable computing*". Laboratoire de Systèmes Logiques, Ecole Polytechnique Fédérale de Lausanne.
7. Mary Jane Irwin, "*Embedded Systems Design*". Pennsylvania State University.
8. Sergio Lopez-Buedo, "*Configuración de FPGAs Xilinx*". Seminarios Euroform, Escuela Politécnica Superior, UAM, 2004.
9. Eduardo Boemo, "*Pipelining*". Seminarios Euroform, Escuela Politécnica Superior, UAM, 2004.
10. XAPP529 (v1.3) "*Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel*". (Mayo 2004)