Emulación de Multiprocesadores en la Plataforma Xilinx ACP

Javier Castillo
ETS Ingeniería Informática
Univ. Rey Juan Carlos
28933 Móstoles
javier.castillo@urjc.es

José Renau
Department of Computer Engineering
University of California Santa Cruz
renau@soe.ucsc.edu

Tom Golubev
Department of Computer Engineering
University of California Santa Cruz
golubev@soe.ucsc.edu

José Ignacio Martínez
ETS Ingeniería Informática
Univ. Rey Juan Carlos
28933 Móstoles
joseignacio.martinez@urjc.es

Resumen

Dentro del campo de la arquitectura de computadores la forma habitual de validar sistemas es mediante el uso de simuladores. Estos simuladores a pesar de ser atractivos debido a su bajo coste son extremadamente lentos a la hora de simular arquitecturas de forma detallada.
En este trabajo se presenta la implementación del microprocesador de código abierto LEON3 sobre la plataforma Xilinx ACP, como paso previo a la implementación de un sistema de emulación de multiprocesadores. El uso de una plataforma como la utilizada permite validar diferentes arquitecturas con un rendimiento varias veces superior a la de los tradicionales simuladores.

1. Motivación

La utilización de FPGAs como sistemas de prototipado es una de sus primeras y más importantes aplicaciones. A día de hoy es impensable mandar a fabricar un circuito integrado sin previamente validar su funcionamiento mediante una implementación sobre FPGA.

Por otra parte los arquitectos de computadores llevan décadas utilizando complejos simuladores para evaluar sus arquitecturas antes de realizar su implementación física. Dichos simuladores permiten obtener datos sobre el funcionamiento de la arquitectura e ir ajustando parámetros hasta obtener el rendimiento deseado.

A medida que los sistemas a desarrollar han crecido en complejidad, los tiempos de simulación han crecido con ellos debido a la gran cantidad de parámetros que deben simular. En la era de los multiprocesadores dicho problema ha crecido aun más si cabe, debido a la necesidad de modelar complejos sistemas de memoria para mantener la coherencia e intrincadas redes de interconexión que permiten comunicar la gran cantidad de cores integrados en una arquitectura.

Ante este desafío ha surgido la idea de utilizar los dispositivos FPGA para la emulación de arquitecturas multiprocesadores en lugar de la tradicional simulación.

En esta línea destaca el proyecto RAMP[1] (Research Accelerator for Multiple Processors). RAMP es un consorcio de empresas y universidades americanas cuyo objetivo es desarrollar modelos y herramientas de sistemas multiprocesadores utilizando para ello plataformas basadas en FPGA.

Dentro de RAMP se han desarrollado diferentes proyectos entre los que podemos destacar el prototipo RAMP-Blue[2] que consiste en 768-1008 Microblazes en 64-84 Virtex-II Pro. Para ello utilizan la placa de prototipado BEE2 desarrollada a medida para este tipo de aplicaciones por la Universidad de Berkeley, uno de los principales promotores de RAMP.

Otro de los prototipos desarrollados es RAMP-Gold[3] que modela un procesador con una arquitectura MTA (Multi-Threaded Architecture) capaz de multiplexar 64 threads en un solo procesador.

En el ámbito del proyecto RAMP ha surgido la necesidad de homogeneizar las plataformas con las que se trabaja, además de abaratar su coste para que los resultados del proyecto puedan ser reproducidos por otras instituciones. En este contexto las universidades participantes han comenzado a trabajar con la plataforma Xilinx ACP[4][5] comercializada por Nallatech.

El uso de esta nueva plataforma a la hora de construir multiprocesadores presenta numerosas dificultades y retos que serán discutidos a lo largo de este artículo.

En el capítulo 2 se presentará la plataforma Xilinx ACP, discutiendo las ventajas y desventajas que ofrece a la hora de diseñar multiprocesadores. En el capitulo 3 se mostrará el

port del procesador LEON3 para trabajar sobre dicha plataforma. Posteriormente se comentará el rendimiento obtenido y las diferentes posibilidades que existen para mejorarlo. Por último se discutirá una posible implementación de un multiprocesador sobre dicha plataforma.

2. La plataforma Xilinx ACP

La plataforma Xilinx ACP está compuesta por un procesador XEON conectado a un stack de FPGAs a través del FSB (Front Side Bus) del sistema. De esta forma la interconexión entre FPGA y CPU se realiza mediante un canal de un elevado ancho de banda (hasta 6.4Gbps) y muy baja latencia, (alrededor de 4 us).

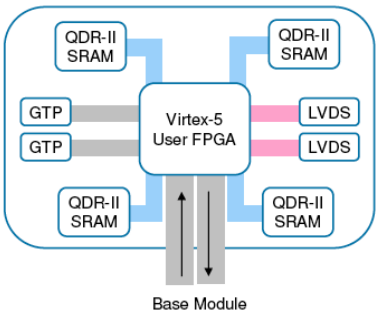

Figura 1. Módulos de Cómputo

Siguiendo esta arquitectura es posible apilar varias de los elementos mostrados en la figura 1 para disponer de más recursos de computación.

Cada uno de los módulos de computación se compone de dos FPGAs Virtex5 330LXT. Estas FPGAs son actualmente las más grandes de la familia Virtex5 proporcionando una gran cantidad de recursos lógicos al diseñador.

Cada uno de los módulos dispone también de 4 memorias SRAM DDR-II de 4Mbytes, siendo por tanto el total de memoria disponible en el módulo de 16MBytes. Esto representa un gran contratiempo a la hora de desarrollar prototipos de multiprocesadores, ya para poder ejecutar un sistema operativo como Linux es necesaria más memoria de la disponible en el módulo. La forma de solucionar este problema será discutida más adelante.

Las conexiones de los módulos con la CPU a través del FSB, así como las conexiones entre las FPGAs dentro del módulo se realizan a través de los interfaces RocketIO de las FPGAs. Para manejar estas conexiones Nallatech (el distribuidor del sistema) proporciona unos *cores* en forma de *netlists* que permiten manejar las comunicaciones. También está disponible un API en C que permite tanto programar las FPGAs desde la CPU, como mandar y recibir datos a través del FSB.

3. LEON3 sobre Xilinx ACP

El objetivo de este trabajo es construir un sistema multiprocesador capaz de ejecutar Linux sobre la plataforma Xilinx ACP.

Para construir el sistema multiprocesador se ha seleccionado el procesador LEON3[6] de Gaisler ya que a partir de una implementación uniprocesador es posible construir un multiprocesador de manera sencilla, sin más que conectar más procesadores como maestros del bus AHB.

El procesador LEON3 es un RISC de 5 etapas, con arquitectura SPARC v8. Dispone de cachés completamente configurables así como soporte de memoria virtual y multitud de periféricos como controladores de memoria SDRAM y DDR, además de un interface de depuración a través de JTAG.

A la hora de portar el sistema a la plataforma Xilinx ACP es necesario solventar las carencias de ésta. La plataforma proporciona gran cantidad de lógica y comunicaciones muy rápidas, pero el único interface de entrada/salida es el FSB. Además como ya se ha comentado los 16Mbytes de memoria SRAM disponibles en el módulo son insuficientes para ejecutar un sistema operativo completo.

Para solucionar estos problemas se ha optado por realizar una modificación en el procesador mediante la cual una parte de los recursos del sistema son llevados al host XEON. Estos recursos son inicialmente la memoria RAM, la memoria ROM y el interface serie RS232, aunque en un futuro se plantean otros periféricos como controladores de red o VGA.

Con esta solución es posible utilizar la memoria del sistema como memoria del LEON3, disponiendo de esta forma de una memoria

prácticamente ilimitada. Además es posible también mostrar una consola de usuario a través de la terminal del XEON.

En las figura 2 y 3 se muestra como sería inicialmente el sistema y cual es la estructura propuesta:

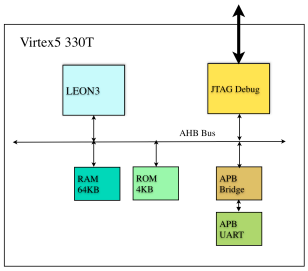

Figura 2. Sistema inicial

El esquema propuesto tiene 2 partes bien diferenciadas:

- Un módulo hardware denominado AHB2FSB que recoge todas las peticiones al bus AHB dirigidas a la ROM, RAM y UART, y las reenvía a través del FSB.
- Un demonio que corre sobre el host que atiende estas peticiones y las mapea sobre la memoria del sistema o en su caso sobre un modelo de puerto serie.

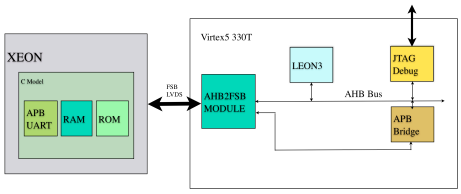

Figura 3. Sístema propuesto.

3.1. Comunicaciones

Nallatech proporciona un API en C que permite realizar comunicaciones a través del FSB entre las FPGAs de los módulos de computación y el host XEON. En el momento de escribir este artículo, este API solo proporciona una primitiva de comunicación para enviar y recibir datos. Dicha primitiva es *ACP_memcopy()*. Utilizando este mecanismo un número entero de líneas de caché del XEON son copiadas a la FPGA a través del FSB. La FPGA debe leer estos datos y enviar de vuelta otro número entero de líneas de caché que son copiados en la caché del host.

Para el objetivo de este trabajo dicho mecanismo es ineficiente ya que es necesario enviar al menos 64bytes de ida y vuelta por cada acceso al bus AHB, cuando únicamente 32 bits son realmente datos útiles. En futuras versiones se pretende utilizar una caché de nivel 2 para el procesador LEON3 del mismo tamaño de línea que la del procesador XEON que mitigue esta ineficiencia.

3.2. Módulo AHB2FSB

Para poder enviar las peticiones que genera el LEON3 al host XEON y poder atenderlas allí se ha desarrollado un módulo VHDL que hace de bridge entre el bus AHB y el bus FSB.

Dicho módulo se muestra en la figura 4.

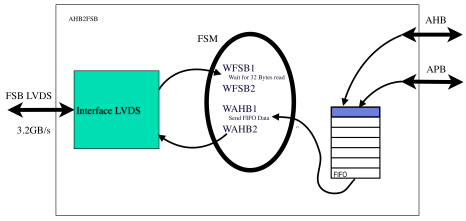

Figura 4. Elemento AHB2FSB.

Este módulo se identifica ante el procesador LEON3 como los tres módulos a los que reemplaza. De esta forma todas las peticiones que deberían ir hacia dichos módulos son reenviadas al bridge.

El bridge está compuesto por tres elementos:

- Una FIFO que recibe todas las peticiones y las almacena en espera de poder ser enviadas, debido a la gran diferencia de velocidad entre el bus FSB y el AHB. Esta FIFO tiene una anchura de 256 bits que es el tamaño de datos que se debe enviar a través del interface proporcionado por Nallatech para interactuar con el bus FSB. En los 256 bits se almacena la dirección del acceso, el dato a escribir en caso de una escritura, así como todos los campos

de control del bus necesarios para realizar correctamente la transferencia solicitada.

- El módulo que permite interactuar con los LVDS. Este módulo es proporcionado en forma de *netlist* por Nallatech y permite intercambiar datos con el procesador XEON utilizando la primitiva *ACP_memcopy()* como se comento anteriormente.
- Una máquina de estados que controla todo el proceso.

Inicialmente el sistema está esperando una petición por parte del bus FSB ya que todas las transferencias son iniciadas por parte del host a través de la llamada a la función *ACP_memcopy()*.

Una vez realizada esa llamada, la máquina de estados debe leer los 64 bytes de la línea de caché transferida que son copiados a una FIFO contenida dentro del módulo de interconexión de Nallatech. Como se transfieren 64 bytes y la FIFO es de 256 bits es necesario realizar dos lecturas para leer la línea completa. En esta línea se manda una confirmación de recepción de la petición anterior junto con el dato leído si la petición era de lectura. La máquina de estados toma los datos y los envía hacia el bus AHB. A continuación comprueba la FIFO que almacena las peticiones pendientes del bus AHB y en caso de haber alguna, esta es enviada hacia el procesador XEON. De nuevo es necesario realizar dos escrituras en la FIFO de 256 bits para enviar al menos los 64 bytes de la línea de caché.

3.3. Modelo C de memoria

Como ya se ha explicado, en el procesador XEON se ejecuta un programa que se encarga de recoger las peticiones y modelar una memoria RAM, una ROM y un puerto serie.

Tanto la memoria RAM como ROM se modelan utilizando un array. El array que modela la ROM es cargado con el binario del programa a ejecutar cuando el programa es lanzado. Es posible de esta manera ejecutar tanto aplicaciones sin sistema operativo como un sistema operativo como uClinux. Esto tiene dos ventajas, la primera es que como ya se ha comentado disponemos de memoria prácticamente ilimitada, la segunda es que no es necesario descargar la imagen del sistema operativo a la memoria de la placa a través de un *bootloader*, ahorrando mucho tiempo durante el proceso de depurado.

Una vez cargados los binarios en los modelos de memoria comienza la ejecución enviando un paquete de datos vacío y leyendo la primera petición que realiza el bus AHB. Esto se ejecuta dentro de un bucle infinito hasta que el usuario pulsa CTRL-C en cuyo caso se cierran todas las conexiones y se finaliza la ejecución.

4. Rendimiento

Una de las mayores preocupaciones durante el desarrollo del sistema es el rendimiento del bus FSB. Aunque el bus frontal trabaja a una gran velocidad existen grandes ineficiencias debido a la restricción de tener que enviar líneas completas de caché.

Para evaluar el rendimiento del bus FSB se han diseñado varios experimentos para medir tanto el ancho de banda como la latencia.

En la figura 5 y 6 se muestran los resultados de latencia y ancho de banda frente a la cantidad de datos trasmitidos en bytes.

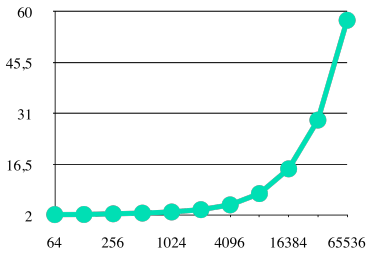

Figura 5. Latencia (us/transferencia)

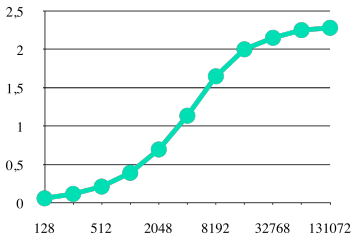

Figura 6. Ancho de Banda (GB/s)

Se puede observar como la latencia es prácticamente constante con tamaños de datos hasta 1KB, en el orden de 2 us, a partir de ahí esta sube de forma exponencial. Esta latencia traducida a transferencias por segundo nos da un resultado de 500.000. Si tenemos en cuenta que el LEON3 trabaja a 100Mhz sobre la FPGA, tenemos que el procesador lanza 100 millones de transferencias por segundo y por tanto el rendimiento es claramente insuficiente.

Sobre el ancho de banda podemos ver como es necesario transmitir una gran cantidad de datos para aprovechar todo el caudal que ofrecen los interfaces LVDS. Esto nos indica que la solución de utilizar una caché intermedia que agrupe las peticiones y las envíe en un paquete mas grande hacia el servidor aceleraría mucho el rendimiento.

En las siguientes gráficas se puede observar el rendimiento global del sistema sin cachés, con caches de instrucciones, con caché de datos y con ambas cachés para diferentes tamaños y asociatividades de las mismas.

Las columnas muestran cuantas veces es más lento el sistema respecto a la versión que corre en placa utilizando como memoria Block RAMs con un ciclo de tiempo de acceso.

Las diferentes columnas para un mismo tamaño indican si la caché era de mapeo directo o asociativa de 2 vías.

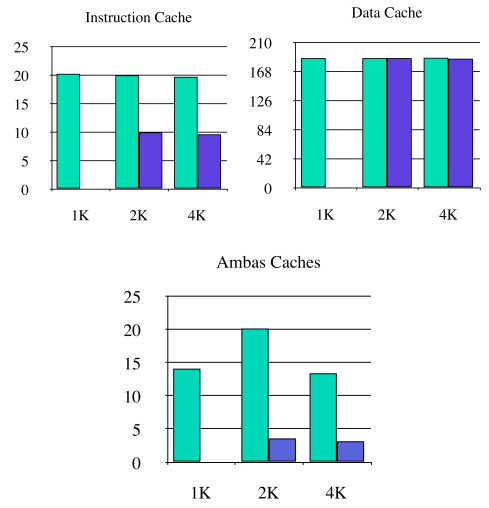

Figura 7. Rendimiento con diferentes caches

Dichas medidas han sido tomadas utilizando un conjunto de tests y promediando sus resultados. Estos tests incluyen programas como *pattern matching*, multiplicación de matrices, tests de acceso a memoria, etc.

La primera medida que se realizó fue sin caches lo cual dio como resultado que el sistema erá 210 veces más lento que el ideal, lo cual es coherente con las medias realizadas anteriormente.

Para este conjunto de tests se nota un gran impacto en el rendimiento al utilizar caché de instrucciones, frente a un menor impacto de la caché de datos. También se percibe que el uso de cachés asociativas de dos vías representa una gran mejora frente a las de mapeo directo, en el caso de utilizar ambas cachés la mejora es de 5 veces.

El tamaño de la caché no es demasiado relevante ya que para poder llevar a cabo las mediciones con respecto al sistema ideal se han utilizado programas no muy grandes para poder ser cargados en la Block RAM de la FPGA.

5. Sistema Multiprocesador

Utilizando la arquitectura propuesta es posible diseñar un multiprocesador que contenga varios procesadores LEON3 compartiendo el bus AHB.

Para comenzar debemos conocer cuantos procesadores podemos introducir en cada una de las dos FPGAs que contiene cada módulo.

En la tabla 1 se muestran resultados de síntesis para el procesador LEON3 sobre la Virtex5 330 LXT con diferentes configuraciones.

	Slices	**RAM**	**#procesadores**
No caches	6%	8%	12
IC 1K	6%	9%	11
IC 2K	6%	11%	9
IC+DC 1K	6%	12%	8
IC+DC 2K	6%	13%	7
IC+DC 2K 2 Sets	7%	14%	7
IC+DC 4K 2 Sets	6%	18%	5

Tabla 1. Rendimiento con diferentes caches

Combinando la información de la figura 7 con la de la tabla 1, llegamos a la conclusión de que para obtener un rendimiento óptimo debemos usar al menos caches de 2 vías y 2K con lo cual pondremos poner un máximo de 7 procesadores en cada FPGA.

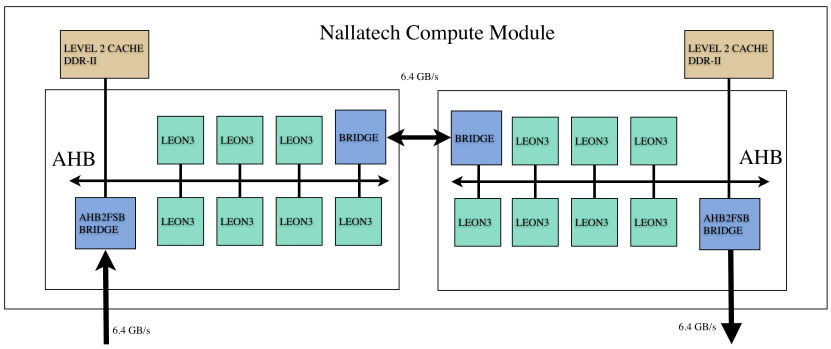

Figura 8. Sistema Multiprocesador

En la figura 8 se muestra la arquitectura que se propone.

Lo primero que llama la atención es el modo de conectar los bridges AHB-FSB. La primera FPGA recibe los datos del FSB mientras que la segunda es la encargada de enviar las peticiones hacia el FSB. Este esquema no es arbitrario y responde a las restricciones de conexiones impuestas por el fabricante de los módulos.

Lo segundo a destacar es la utilización de las memorias SRAM DDR-II como cachés de nivel 2. De esta forma cuando uno de los procesadores sufre un fallo de caché irá a buscar los datos a la caché de nivel 2, en caso de tampoco estar el dato allí se realizaría la petición de una línea completa de caché al procesador XEON a través del FSB utilizando los bridges. Estas caches tienen un ancho de línea de 32 bytes. De esta forma utilizamos la mitad de los 64 bytes de la unidad mínima de transmisión para los datos y lo otros 32 bytes para información de control, reduciendo de esta forma la ineficiencia.

En la figura se puede ver como los 14 procesadores comparten el bus del sistema y disponen de un espacio único de memoria siguiendo por tanto el multiprocesador una arquitectura SMP (*Symmetric Multi-Processing)*.

En la figura no se muestra el mecanismo de coherencia. En el caso de las cachés de nivel 1 los procesadores LEON implementan un mecanismo de coherencia Snoopy que sigue siendo válido en este sistema ya que los 14 procesadores comparten un bus único a través de los bridges entre FPGAs. Entre las memorias SRAM que actúan como cachés de nivel 2 no es necesario mantener la coherencia ya que el espacio total de memoria de nivel 2 está dividido entre ambas, estando localizada las direcciones bajas en la memoria de la FPGA 0 y las altas en la memoria de la FPGA 1. No obstante dentro del trabajo futuro se pretende estudiar diferentes mecanismos de coherencia para encontrar la solución óptima.

6. Trabajo futuro

A lo largo del artículo se han ido explicando los problemas planteados y las posibles soluciones. Algunas de estas soluciones no han sido todavía llevadas a cabo y entran dentro del trabajo que se realizará a partir de este momento.

Entre ellas destaca el uso de una caché de nivel 2 que agrupe el tráfico el entre la FPGA y el host aumentando de esta forma el ancho de banda.

Otra solución en la que se está trabajando en este momento es en utilizar una nueva API desarrollada por ArchES Computing denominada ArchesMPI[7]. Utilizando Arches-MPI es posible utilizar la FPGA como si fuera un rango dentro de una aplicación MPI, simplificando de esta forma las comunicaciones. Otra ventaja es que Arches-MPI implementa un nuevo API que permite realizar transferencias con una latencia mucho más baja que con el API proporcionado por Nallatech, del orden de centenares de ns en lugar de us. Además con Arches-MPI se puede definir un rango de memoria compartido entre el host y la FPGA y al cual se puede acceder directamente desde el hardware con llamadas a las funciones

MPI_get() y *MPI_put()*. El sistema multiprocesador será implementado utilizando esta nueva API.

7. Conclusiones

Se ha presentado una posible implementación de un sistema multiprocesador sobre la plataforma Xilinx ACP de Nallatech. Se ha realizado un estudio de las limitaciones de esta plataforma a la hora de desarrollar sistemas de este tipo y se han plateado posibles soluciones a las mismas.

El sistema uniprocesador está actualmente funcionando y se está trabajando en la implementación del sistema multiprocesador junto con las mejoras necesarias para incrementar su rendimiento.

Referencias

[1] RAMP, Research Accelerator for Multiple Processors, http://ramp.eecs.berkeley.edu

[2] A. Krasnov , A. Schultz , J. Wawrzynek , G. Gibeling , P. Droz, Ramp blue: a message-passing manycore system in FPGAs, In 2007 International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 27-29

[3] K. Asanovic, D. A. Patterson, Z. Tan, A. Waterman, R. Avizienis, Y. Lee, RAMP Gold: An FPGA-based Architecture Simulator for Multiprocessors, in The 4th Workshop on Architectural Research Prototyping, 2009

[4] C. Maxfield, FPGA-based solution interfaces to Intel bus http://www.eetimes.com/news/design/showArticle.jhtml?articleID=203100031

[5] Nallatech, FSB Compute Modules, http://www.nallatech.com/Intel-Xeon-FSB-Socket-Fillers/fsb-compute-module.html

[6] Gaisler Research, LEON3 Processor, http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53

[7] M. Saldaña, E. Ramalho, and P. Chow, A Message-Passing Hardware/Software Co-simulation Environment for Reconfigurable Computing Systems, in International Journal of Reconfigurable Computing, 2009.