iLP-NUCA: Cache de Instrucciones Teselada para Procesadores Empotrados

Alexandra Ferrerón Labari, Marta Ortín Obón, Darío Suárez Gracia, Jesús Alastruey Benedé y Víctor Viñals Yúfera[1]

Resumen— **La búsqueda de instrucciones (*fetch*) es una etapa fundamental del segmentado en los procesadores actuales. Su misión es suministrar un flujo continuo de instrucciones a las posteriores etapas, tarea dificultada por los saltos y los fallos de la cache de instrucciones. En el caso de los procesadores multihilo, la unidad de búsqueda es más crítica ya que debe gestionar varios flujos de ejecución.**

Este trabajo propone una nueva cache de instrucciones, *instruction Light Power NUCA* (iLP-NUCA), para procesadores empotrados de gama alta con el objetivo de mejorar el rendimiento. La iLP-NUCA añade una nueva topología en árbol para la red de transporte que permite reducir el tiempo medio de acceso respecto a una organización LP-NUCA convencional.

Nuestros experimentos muestran que, ejecutando SPEC CPU2006, la nueva topología reduce un 8 % la latencia de servicio respecto al diseño original. Además el IPC mejora un 6 % en media con respecto a una jerarquía convencional de altas prestaciones (L1 + L2 separadas para instrucciones y datos), y al evaluar en exclusiva la jerarquía cache de instrucciones se acerca a las prestaciones obtenidas por una jerarquía ideal, 99.5 % del rendimiento de una cache de instrucciones L1 perfecta. Estas mejoras son directamente proporcionales a la tasa de fallos y aumentan con el número de hilos en ejecución. Evaluando una muestra estadísticamente representativa obtenemos unas ganancias del 12.5 % y 20.6 % con 2 y 4 hilos, respectivamente.

Palabras clave—**Instrucciones, jerarquía de memoria, multihilo, SMT, cache, NUCA**

I. Introducción

LOS núcleos de ejecución superescalares demandan flujos de instrucciones continuos para alimentar sus unidades funcionales. Estos flujos son interrumpidos por saltos y por los fallos de cache, lo que puede afectar a la tarea de lanzamiento de instrucciones y, por tanto, a su rendimiento.

Por otra parte, los procesadores *Simultaneous Multi-Threading* (SMT) permiten ocultar operaciones de latencia larga mediante la ejecución de varios hilos [1]. Este modelo de ejecución puede implementarse con una pequeña sobrecarga en área y energía, parámetros de diseño especialmente importantes en procesadores empotrados. Sin embargo, el diseño de la cache de instrucciones se complica por la presión añadida y se convierte en un componente crítico para el rendimiento global del sistema.

Una cache de instrucciones capaz de almacenar el *working set* de las aplicaciones más exigentes reduciría el número de fallos, pero su gran tamaño podría aumentar su tiempo de acceso así como su consumo de energía. Por tanto, es necesario buscar un compromiso entre tamaño y latencia o utilizar una cache adaptativa que pueda variar sus características a lo largo de la ejecución del programa [2][3][4]. Este trabajo explota esta segunda opción y propone una cache de instrucciones basada en L-NUCA [4].

Las contribuciones principales de este trabajo son las siguientes:

- Aplicación de una organización NUCA a la jerarquía de instrucciones. Hasta donde alcanza nuestro conocimiento, este es el primer trabajo que propone una cache NUCA específica para la jerarquía de instrucciones.
- Propuesta de una topología en árbol que reduce la latencia de servicio de la red de transporte original de LP-NUCA.

El nuevo diseño de jerarquía de instrucciones propuesto, iLP-NUCA, supera el rendimiento de una jerarquía convencional multibanco y alcanza el 99.5 % de las prestaciones obtenidas por una cache perfecta. Las ganancias son directamente proporcionales a la tasa de fallos con independencia del número de hilos, con lo que podemos suponer que la mejora sería mayor con otras cargas de trabajo que presionen más la unidad de búsqueda del procesador.

El resto de este artículo está organizado del siguiente modo. La Sección II presenta las caches de instrucciones analizadas en este estudio haciendo énfasis en la propuesta iLP-NUCA. La Sección III propone una nueva topología de transporte para iLP-NUCA. La Sección IV describe la metodología. La Sección V analiza los resultados. La sección VI revisa los trabajos relacionados y en la Sección VII se resumen las conclusiones de este artículo.

II. Jerarquías de Caches de Instrucciones para Procesadores Empotrados

Este trabajo pretende explorar el espacio de diseño de las caches de instrucciones para mejorar el rendimiento.

Vamos a comparar una jerarquía convencional de tres niveles de cache (CV), ver Figura 1, con nuestra propuesta de cache teselada (ILP). En ambas jerarquías los dos primeros niveles son dedicados, mientras que el tercer nivel es multibanco y compartido por instrucciones y datos. Escogemos esta configuración por su mayor potencial en cuanto a prestaciones. En el sistema CV, el segundo nivel de cache es multibanco, las latencias de acceso para L1 y L2 son 2 y 4 ciclos, y las latencias de iniciación, 1 y 2, respectivamente.

Las LP-NUCAs [5] extienden la capacidad del primer nivel de la jerarquía de cache convencional ro-

[1]gaZ. Dpto. de Informática e Ing. de Sistemas. Inst. de Investigación en Ing. de Aragón. Universidad de Zaragoza, e-mail: {ferreron, ortin.marta, dario, jalastru, victor}@unizar.es

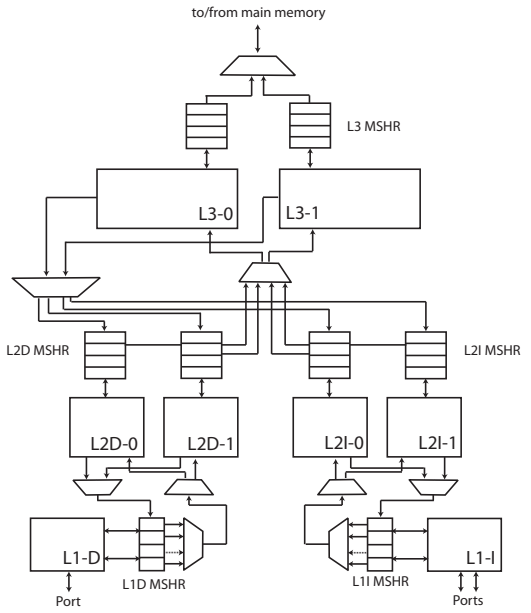


Fig. 1. Esquema del sistema convencional de altas prestaciones. Por simplicidad se muestra únicamente el camino de loads, omitiendo las estructuras necesarias para los stores.

deándolo de uno o más niveles de teselas interconectadas por redes especializadas (Figura 2). El primer nivel de LP-NUCA (root-tile, RT) se corresponde con una cache L1, y sustituimos el segundo nivel de la jerarquía convencional por una estructura teselada en niveles cuyos tiempos efectivos de acceso aumentan gradualmente. Los números que aparecen en cada tesela son su latencia mínima de servicio, definida como el tiempo desde que se inyecta un fallo en la LP-NUCA hasta que el bloque es servido por dicha tesela. El diseño de la LP-NUCA se caracteriza por el uso de tres redes especializadas: búsqueda, transporte y reemplazo, que permiten una implementación sencilla y un elevado ancho de banda. En la Figura 3 se muestra el esquema basado en iLP-NUCAS de dos niveles que se comparará con el sistema CV.

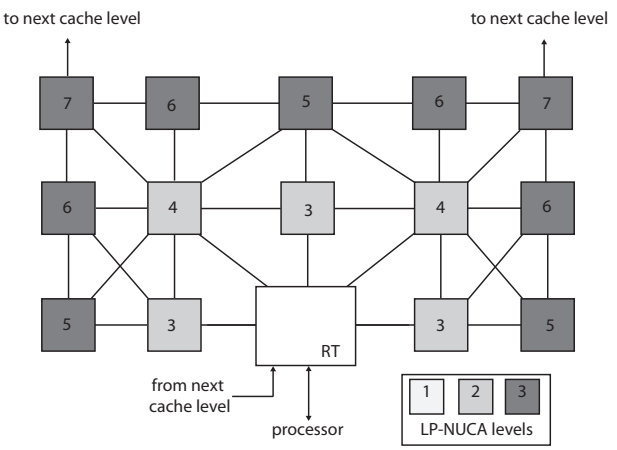


Fig. 2. LP-NUCA con tres niveles. Los números en cada tesela se corresponden con su latencia mínima de servicio.

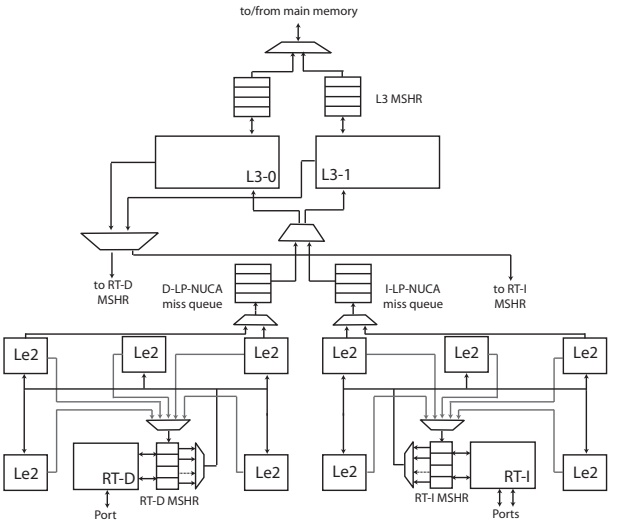


Fig. 3. Esquema del sistema con iLP-NUCAs para datos e instrucciones. Por simplicidad se muestran únicamente dos niveles y el camino de loads, omitiendo las estructuras necesarias para los stores.

III. Red de Transporte en árbol para iLP-NUCA

La red de transporte previamente utilizada era una malla 2-D debido al ancho de banda que proporcionan los múltiples caminos de datos hacia el RT. Con esta topología, las latencias de las teselas son mayores cuanto más alejadas se encuentran del RT.

La Figura 4 muestra una red de transporte alternativa con topología de árbol. Con esta red conseguimos un doble objetivo: disminuir el número de enlaces entre las teselas y reducir la latencia mínima de servicio, especialmente de las teselas más alejadas del RT (niveles superiores de la LP-NUCA). Con esta nueva alternativa se pretende reducir el tiempo medio de servicio de bloques en la LP-NUCA.

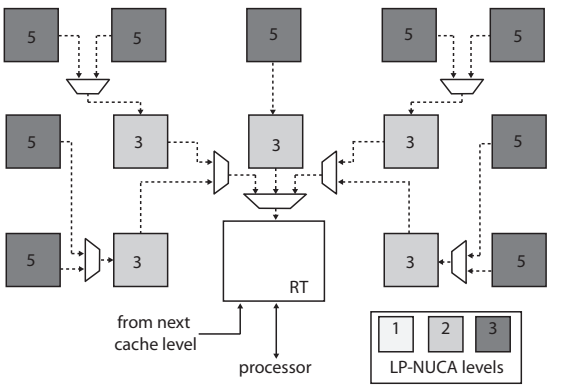


Fig. 4. Nueva red de transporte con topología de árbol para una LP-NUCA de 3 niveles. Los números en cada tesela se corresponden con su latencia mínima de servicio.

IV. Metodología

A. Modelo de Procesador y Entorno de Simulación

El modelado y la experimentación se han realizado con un simulador basado en SimpleScalar [6] que extiende el utilizado en trabajos anteriores de LP-NUCA [5] e incluye jerarquías de cache de ins-

trucciones modeladas en detalle. El procesador base se asemeja al IBM/LSI PowerPC 476FP [7][8] con una jerarquía de memoria más potente y es capaz de ejecutar de manera simultánea hasta 4 hilos. La Tabla I resume los parámetros principales del procesador. Todas las caches tienen un único puerto excepto el primer nivel de instrucciones, que tiene dos.

TABLA I

PROCESADOR Y MEMORIAS MODELADAS. BS, LAT E INIT INDICAN TAMAÑO DE BLOQUE, LATENCIA Y LATENCIA DE INICIACIÓN, RESPECTIVAMENTE

Frequencia de reloj	1 GHz	Anchura Fetch/-Decode/Commit	2
Anchura Issue	2(IN+ME) + 2FP	Entradas ROB / LSQ	32 / 16
Entradas IW INT/FP/-MEM	8 / 8 / 8	Predictor de saltos	bimodal + gshare, 16 bit
Penalización por salto	6		
Entradas MSHR L1/L2/L3	8 / 8 / 4	Latencia fallo TLB	30
Entradas secundarias MSHR	4	Tamaño buffer escritura WB / L2 / L3[a]	16 / 8 / 8

L1/RT[b]	32KB–4Way–32B BS, write-through, 2-cycle lat, 1-cycle init
L2	512KB–8Way–32B BS, serial tag-data, 4-cycle lat, 2-cycle init, copy-back
teselas iLP-NUCA	32KB–2Way–32B BS, parallel tag-data, copy-back, levels: 3, total size: 448KB

L3	4MB eDRAM–16Way–128B BS, 14-cycle lat, 7-cycle init, copy-back
Main Memory	100 ciclos/4 por trozo extra, bus: 16B

[a] Los buffers de escritura de L2, iLP-NUCA y L3 fusionan entradas

[b] Root-tile: copy-back y write-around

La política de búsqueda utilizada es ICOUNT [1] y todos los componentes del procesador como el ROB, las ventanas de lanzamiento (IWs),... son compartidos por todos los hilos. El *commit* se realiza de manera circular entre los hilos y, para evitar la inanición, ningún hilo puede ocupar más de tres cuartas partes de los *write-buffers* de L2/iLP-NUCA.

La tecnología asumida es de 32nm y las latencias y tasas de iniciación de las caches convencionales se han obtenido con Cacti 6.5 [9]. Para los cálculos de las iLP-NUCA se ha utilizado Cacti 6.5 y un layout en 90 nm [9][5].

B. Cargas de Trabajo

Se han utilizado los benchmarks de SPEC CPU2006 [10], excepto 483.xalancbmk, que no hemos podido ejecutar en nuestro entorno. Para cada programa se simulan 100M de instrucciones representativas seleccionadas con la metodología *SimPoints* [11]. En los experimentos monohilo se calientan las caches y el predictor de saltos durante 200M de instrucciones. Las cargas de trabajo multihilo son multiprogramadas, se utiliza la política *last* de simulación, no hay calentamiento de estructuras y las estadísticas sólo se obtienen para los primeros 100M de instrucciones de cada hilo. Para asegurar la representatividad de los resultados se ha seguido una metodología basada en muestreo aleatorio que asegura un error menor del 3 % con un intervalo de confianza del 97 % [12][13][14].

C. Métricas

Para evaluar el rendimiento de las distintas alternativas vamos a utilizar métricas exclusivas de la jerarquía como la tasa de fallos y el tiempo medio de acceso a memoria, y métricas de sistema como el IPC. Para cargas multihilo, además del IPC agregado de todos los hilos (*IPC throughput*) utilizamos el *fairness* que permite cuantificar la equidad de los diseños propuestos [15]. Estas métricas se calculan con las siguientes fórmulas:

$$IPC\ throughput = \sum_{i=1}^{n} IPC_i,\ fairness = \frac{\min_i \left(\frac{CPI_i^{MP}}{CPI_i^{SP}}\right)}{\max_i \left(\frac{CPI_i^{MP}}{CPI_i^{SP}}\right)}$$

donde CPI son ciclos por instrucción y MP y SP se refieren a la ejecución multi- y monohilo, respectivamente.

V. EVALUACIÓN

A. Resultados monohilo

La Figura 5 muestra el rendimiento en IPC (instrucciones por ciclo) para todas las cargas de trabajo consideradas. Hemos agrupado las cargas de trabajo en dos grupos:

1. En la parte de la izquierda de la gráfica aparecen los benchmarks cuyo MPKI[1] para instrucciones es menor que 1. En estos programas, el *working set* cabe en el primer nivel de cache y su rendimiento no se ve afectado por caches de instrucciones más pequeñas y con menos asociatividad.
2. En la parte derecha de la gráfica mostramos el resto de las cargas (MPKI mayor que 1) y que por tanto podrían beneficiarse más del diseño propuesto.

Se incluyen dos barras adicionales con las medias armónicas de los programas cuyo MPKI es alto (High-i-MPKI-Hmean), y del conjunto (HMean), respectivamente.

La figura presenta tres configuraciones con caches de datos perfectas: una jerarquía convencional multibanco (CV_DP), una configuración con iLP-NUCA de tres niveles (ILP_DP, 448 KB) y una jerarquía perfecta de datos e instrucciones (Always-hit).

La primera observación reseñable es que CV_DP alcanza en media el 98.3 % del rendimiento ideal (Always-hit). La ILP_DP se acerca todavía más al rendimiento ideal (99.5 %). Al seleccionar los benchmarks con mayor tasa de fallos ambos porcentajes se reducen, más en el caso de CV_DP que para ILP_DP (97.1 % y 98.8 %, respectivamente).

Si se comparan los IPCs de CV_DP e ILP_DP observamos una ganancia media del 1.2 % y del 1.9 % para todos los benchmarks y para aquellos que fallan

[1] *Misses per kiloinstruction* o fallos por cada mil instrucciones

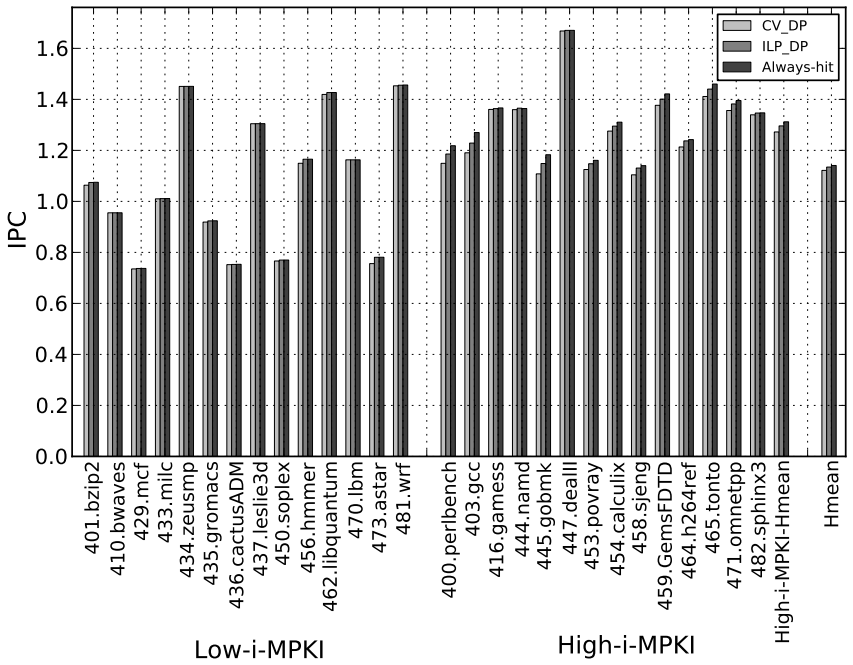


Fig. 5. Rendimiento aislado de la jerarquía de instrucciones en tres configuraciones con cache de datos perfecta: una jerarquía convencional multibanco (CV_DP), una configuración con iLP-NUCA de 3 niveles (ILP_DP, 448 KB) y una jerarquía perfecta de datos e instrucciones (Always-hit).

más, respectivamente. Estos resultados nos permiten suponer que cargas de trabajo con mayores tasas de fallos presentarían ganancias mayores. Este hecho ya había sido observado en anteriores propuestas para caches de datos [4] y se ve corroborado en benchmarks como 445.gobmk, que presenta el MPKI de instrucciones mayor y obtiene la máxima ganancia del 3.7 % respecto a CV_DP.

La Figura 6 muestra el IPC para las mismas configuraciones y conjuntos de benchmarks de la Figura 5 pero considerando caches de datos reales. El rendimiento en este caso también se ve afectado por la jerarquía de datos. Respecto a CV, ILP obtiene una ganancia media del 6 %. Obtenemos mayor rendimiento en todas las cargas de trabajo consideradas excepto en 470.lbm. La razón principal es que la mayor parte de sus accesos son stores que aciertan en la L2 convencional, pero en el caso de la iLP-NUCA lo hacen en la L3 porque iLP-NUCA sigue la política *no-write-allocate*.

También cabe destacar que aumenta el rendimiento considerablemente en programas como 436.cactusADM (43 %), 410.bwaves (33 %) o 481.wrf (27 %). Estas mejoras están relacionadas con la reducción de latencia y el aumento en el ancho de banda de la iLP-NUCA respecto a la jerarquía convencional.

B. Resultados multihilo

En un procesador SMT la latencia de las operaciones lentas se esconde gracias a la ejecución simultánea de instrucciones de múltiples hilos. Por tanto, a medida que aumenta el número de hilos, disminuye en parte la necesidad de reducir la latencia en la jerarquía de memoria. Además, como los *working set* de instrucciones son reducidos, la L2 es capaz de alojarlos y la jerarquía convencional alcanza un rendimiento muy similar a la ideal.

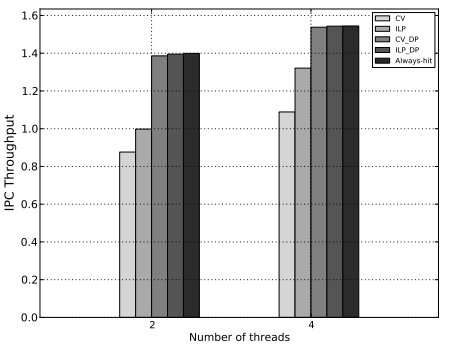


Fig. 7. IPC throughput para mezclas de 2 y 4 hilos

La Figura 7 muestra el *IPC throughput* para combinaciones de 2 y 4 hilos para la jerarquía convencional e iLP-NUCA con caches de datos perfectas (CV_DP e ILP_DP) y no perfectas (CV e ILP). CV_DP consigue un 99 % del rendimiento respecto a la jerarquía ideal (Always-hit) para 2 hilos y alcanza el 99.5 % con cuatro hilos; ILP_DP supera el 99.7 % en ambos casos. En cuanto al *IPC throughput*, conseguimos una mejora máxima del 2 % cuando la cache de datos es perfecta. Con caches de datos reales, las ventajas de iLP-NUCA se ven en instrucciones y datos y las ganancias aumentan considerablemente siendo en media de 12.5 % y 20.6 % para 2 y 4 hilos, respectivamente.

Por último, la Figura 8 muestra el *fairness* del

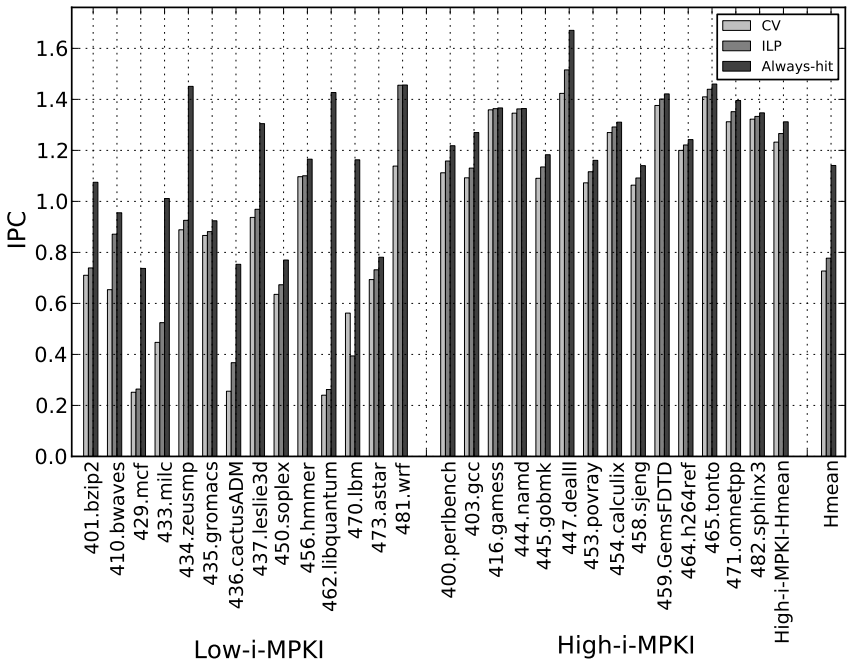
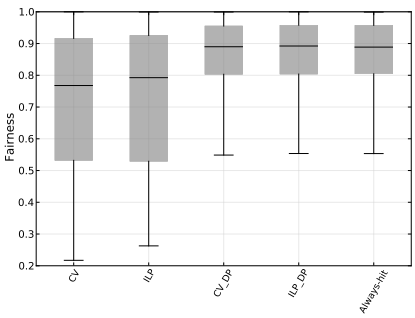
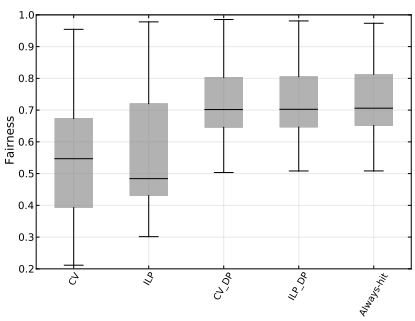


Fig. 6. Rendimiento de la jerarquía en tres configuraciones con caches de datos no ideales: una jerarquía convencional multibanco (CV_DP), una configuración con iLP-NUCA de 3 niveles (ILP_DP, 448 KB) y una jerarquía perfecta de instrucciones (Always-hit).



(a) 2SMT



(b) 4SMT

Fig. 8. *Fairness* variando el número de hilos en ejecución. Se muestra para cada configuración la distribución de *fairness* de todos los benchmarks de todas las combinaciones de la muestra. La línea inferior, la cota inferior del bloque, la línea del bloque, la cota superior del bloque y la línea superior representan el mínimo, cuartil 25, mediana, cuartil 75 y máximo de la distribución, respectivamente

sistema para dos (8a) y cuatro hilos (8b). A la vista de la Figura podemos concluir que aumentamos el rendimiento sin perjudicar a ningún programa. Tanto para 2 como para 4 hilos, ILP tiene un *fairness* mínimo mayor que CV. Si observamos la Figura 8b, vemos que la mediana de ILP es algo menor que para CV para 4 hilos. Esto se debe a que los IPC con un hilo de ILP son mayores que los de CV y que con 4 hilos el sistema está limitado por otros componentes y sufren desaceleraciones mayores.

C. Impacto de la red de transporte en árbol

La ventaja principal de la topología en árbol respecto a la malla es la reducción del número de enlaces en cada nodo. Con una topología en malla 2-D cada nodo tiene dos enlaces de salida, mientras que la topología en árbol los reduce a 1. Aunque disminuye la diversidad de caminos, el tráfico observado en la red de transporte sugiere que la red no va a experimentar contención. Además, se reduce el número de *hops*, lo que se traduce en ahorro energético al servir los aciertos al root-tile.

En efecto, la topología en árbol es capaz de disminuir un 8 % el tiempo medio de servicio de la LP-NUCA[2]. La mayor ganancia en latencia se da en las teselas más alejadas del root-tile, que experimentan un numero menor de aciertos. Suarez *et al.* [4] ya observaron que la mayoría de los aciertos se dan en el nivel 2 de la LP-NUCA (60 % en el caso de SPEC CINT2006 y 40 % en caso de SPEC CFP2006). Por lo tanto, la ganancia en tiempo de servicio tiene un

[2]Tiempo que transcurre desde un fallo en el root-tile hasta que los datos están disponibles para ser insertados en el root-tile

impacto menor en IPC.

VI. Trabajo relacionado

La mayor parte de trabajos para mejorar la unidad de búsqueda en procesadores SMT se centran en la gestión de recursos de la misma. Por ejemplo, Falcón *et al.* proponen que cada ciclo solamente se lean instrucciones de un hilo y plantea un diseño cuyas prestaciones son superiores al de una cache capaz de buscar en el mismo ciclo instrucciones de varios hilos [16]. Gonçalves *et al.* observaron que, para procesadores SMT, incrementando la asociatividad de la cache de instrucciones se obtienen mejoras mayores que con caches de instrucciones privadas para cada hilo [17].

Son más numerosos lo trabajos dedicados al estudio de caches de datos para SMT. La mayor parte inciden en la importancia del ancho de banda para mantener el rendimiento [18][19].

VII. Conclusiones y Trabajo Futuro

Los procesadores empotrados de gama alta requieren potentes etapas de fetch para mantener un flujo continuo de instrucciones independientemente del número de hilos que estén ejecutado. Para alcanzar este objetivo, proponemos una nueva cache de instrucciones teselada, iLP-NUCA, que permite mantener el *working set* a la menor latencia posible.

La iLP-NUCA presenta una nueva red de transporte basada en árbol que reduce la latencia de servicio. Para las cargas analizadas, SPEC CPU2006, iLP-NUCA consigue una ganancia en IPC del 6 % en media respecto a una jerarquía convencional con una L1 y L2 dedicadas a instrucciones. Debido al diseño teselado y al gran ancho de banda de las iLP-NUCA, estas ganancias crecen cuando se incrementa el número de hilos en ejecución y aumenta la presión sobre la jerarquía. En concreto, las ganancias de *IPC throughput* son del 12.5 % y 20.6 % para 2 y 4 hilos, respectivamente. Además, también se mejora el *fairness*, por lo que las ganancias se distribuyen de manera más igualitaria entre todos los hilos en ejecución.

Nuestro trabajo futuro se va a centrar en la evaluación del consumo energético de las iLP-NUCA y en intentar proponer técnicas que reduzcan su consumo y permitan su integración en multiprocesadores dentro del chip. Además, queremos estudiar una organización de la iLP-NUCA en la que cada hilo disponga de una parte privada de cache.

Agradecimientos

Queremos agradecer a Manolis G.H. Katevenis sus sugerencias sobre la red de transporte. Este trabajo ha sido financiado en parte por los proyectos TIN2010-21291-C02-01 (Gobierno de España y Unión Europea), gaZ: T48 grupo de investigación (Gobierno de Aragón y Unión Europea), Consolider CSD2007-00050 (Gobierno de España) y la red de excelencia europea HiPEAC3.

Referencias

[1] D. Tullsen, S. J. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor," in *Proceedings of the 23rd annual international symposium on Computer architecture*, New York, NY, USA, 1996, ISCA '96, pp. 191–202, ACM.

[2] David H. Albonesi, "Selective cache ways: on-demand cache resource allocation," in *MICRO 32: Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture*, Washington, DC, USA, 1999, pp. 248–259, IEEE Computer Society.

[3] Changkyu Kim, Doug Burger, and Stephen W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X)*. October 2002, pp. 211–222, ACM Press.

[4] D. Suarez, T. Monreal, F. Vallejo, R. Beivide, and V. Viñals, "Light NUCA: a proposal for bridging the inter-cache latency gap," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 3001 Leuven, Belgium, Belgium, 2009, DATE '09, pp. 530–535, European Design and Automation Association.

[5] D. Suárez, G. Dimitrakopoulos, T. Monreal, M. G. H. Katevenis, and V. Vi nals, "LP-NUCA: Networks-in-cache for high- performance low-power embedded processors," *IEEE Transactions on Very Large Scale Integration (VLSI) systems*, vol. PP, no. 99, pp. 1, 2011.

[6] Todd Austin and Doug Burger, *SimpleScalar Tutorial (for tool set release 2.0)*, SimpleScalar LCC, 1997.

[7] Tom R. Halfhill, "The rise of licensable SMP," *Microprocessor Report*, vol. 24, no. 2, pp. 11–18, 2010.

[8] LSI Corporation, "PowerPC^{TM} processor (476FP) embedded core product brief, `http://www.lsi.com/DistributionSystem/AssetDocument/PPC476FP-PB-v7.pdf`," January 2010.

[9] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "CACTI 6.0: A tool to model large caches," Tech. Rep. HPL-2009-85, HP Laboratories, April 2009.

[10] John L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[11] Greg Hamerly, Erez Perelman, Jeremy Lau, and Brad Calder, "Simpoint 3.0: Faster and more flexible program analysis," in *Proceedings of Workshop on Modeling, Benchmarking and Simulation*, 2005.

[12] Raj Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, Inc., April 1991.

[13] Dennis Wackerly, William Mendenhall, and Richard L. Scheaffer, *Mathematical Statistics with Applications*, BROOKS/COLE CENGAGE Learning, 7th edition, 2008.

[14] D. Suárez, Monreal T, and V. Viñals, "A comparison of cache hierarchies for SMT processors," in *Proc. of the 22th Jornadas de Paralelismo (JJPAR'11)*, 2011.

[15] R. Gabor, S. Weiss, and A. Mendelson, "Fairness and throughput in switch on event multithreading," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, 2006, MICRO 39, pp. 149–160, IEEE Computer Society.

[16] A. Falcón, A. Ramirez, and M. Valero, "A low-complexity, high-performance fetch unit for simultaneous multithreading processors," in *Proceedings of the 10th Intl. Conference on High Performance Computer Architecture*, 2004, pp. 244–253.

[17] R. Gonçalves, E. Ayguadé, and M. Valero, "A simulator for SMT architectures: Evaluating instruction cache topologies," in *12th Symposium on Computer Architecture and High Performance Computing*, 2000, pp. 279–286.

[18] Sébastien Hily and André Seznec, "Contention on 2nd level cache may limit the effectiveness of simultaneous multithreading," Tech. Rep. 1086, IRISA, février 1997.

[19] M. Nemirovsky and W. Yamamoto, "Quantitative study of data caches on a multistreamed architecture," in *In Workshop on Multithreaded Execution, Architecture and Compilation*, 1998.