UNIVERSIDAD DE CANTABRIA
Facultad de Ciencias

Proyecto Fin de Carrera

SIMULACIÓN Y EVALUACIÓN DE ARQUITECTURAS DE PROCESADOR CON 3D STACKING Y TECNOLOGÍA DE MEMORIA STT-RAM

(Simulation and Evaluation of Processor Architectures with 3D Stacking and STT-RAM Memory Technology)

Para acceder al Titulo de

INGENIERO EN INFORMÁTICA

Autor: Jorge Bellón Castro

Septiembre - 2012

INGENIERÍA EN INFORMÁTICA

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Jorge Bellón Castro
Director del PFC: Pablo Abad Fidalgo
Título: "**Simulación y Evaluación de Arquitecturas de procesador con 3D Stacking y Tecnología de Memoria STT-RAM**"
Title: "Simulation and Evaluation of Processor Architectures with 3D Stacking and STT-RAM Memory Technology"

Presentado a examen el día:

para acceder al Título de

INGENIERO EN INFORMÁTICA

Composición del Tribunal:
Presidente (Apellidos, Nombre): González Harbour, Michael
Secretario (Apellidos, Nombre): Martínez Fernández, María del Carmen
Vocal (Apellidos, Nombre): Menéndez de Llano Rozas, Rafael
Vocal (Apellidos, Nombre): Sánchez Barreiro, Pablo
Vocal (Apellidos, Nombre): Sanz Gil, Roberto

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: Vocal	Fdo.: Vocal
Fdo.: Vocal	Fdo.: El Director del PFC

Tabla de contenido

Lista de figuras

Lista de tablas

Resumen

La divergencia de la velocidad entre el procesador y la memoria es un fenómeno conocido por los arquitectos de computadores desde hace décadas. Desde entonces, múltiples soluciones han visto la luz con la intención de paliar esta diferencia y permitir computadores cada vez más rápidos. No obstante, con la aparición de los multiprocesadores en chip, esta diferencia ha comenzado a crecer vertiginosamente. Las tecnologías de almacenamiento en memoria utilizadas tradicionalmente están convirtiéndose en un impedimento en cuanto a rendimiento y coste energético, por lo que la utilización de nuevas alternativas que superen estas barreras se está haciendo realidad.

En este trabajo se evalúa la viabilidad, desde el punto de vista del rendimiento, de una de las candidatas más importantes para sustituir a las memorias internas en el chip, conocida como Spin Torque Transfer-RAM (STT-RAM). Para ello, se valorará una propuesta ya publicada y se propondrán diversas alternativas que permitirán superar algunos de los problemas que plantea el uso de estas memorias.

Abstract

Processor and memory speed divergence is a well known issue by computer architects. Many researchers have come out with solutions able to reduce this growing gap to enable faster computers. However, chip multiprocessors' appearance has made this difference to start growing much faster than it used. Traditional memory technologies are becoming an impediment due to their limited performance and high energy consumption, turning the utilization of alternative technologies a reality to overcome this barrier.

This paper evaluates the feasibility, from the performance point of view, of one of the most important candidates to replace on-chip memory, known as Spin Torque Transfer-RAM (STT-RAM). An already published proposal will be assessed and a few alternatives solving part of the limitations of this technology will be proposed.

Palabras clave

STT-RAM, 3D-Stacking, Bandwidth wall, Memory wall.

Introducción

A lo largo de las últimas décadas, el incremento de la densidad de integración del número de transistores en un mismo área de silicio ha doblado aproximadamente el número de transistores en un chip cada 2 años, como predijo en 1965 la ley de Moore [1] [2]. Esto ha permitido a los arquitectos de computadores mejorar el rendimiento de los procesadores con el paso del tiempo.

La constante disminución de tamaño de los transistores ha permitido aumentar de forma continuada la frecuencia de trabajo de los procesadores. Adicionalmente, técnicas como la segmentación, que divide la ejecución de las instrucciones en etapas, han permitido aun mayores incrementos en la frecuencia de operación de los procesadores. Como muestra, desde el lanzamiento del Intel 4004 hasta las últimas ediciones del Pentium IV, el tamaño de los transistores se redujo de 10µm a 90nm y la frecuencia se incrementó de 740KHz a más de 3GHz. Esta evolución en la frecuencia de operación de los procesadores ha sido un factor importante en el creciente número de instrucciones por segundo que una máquina es capaz de ejecutar, que asciende de las 92kIPS (Instrucciones Por Segundo) del 4004 a las 9726MIPS del Pentium IV, reduciendo de forma drástica el tiempo de ejecución de las aplicaciones. Por ejemplo, de acuerdo con la Standard Performance Evaluation Corporation (SPEC) [3], un servidor actual con un procesador Intel Xeon X5690 a 3.46 GHz es 40 veces más veloz ejecutando un benchmark de aplicaciones que una máquina de referencia de 1997, con un procesador UltraSPARC II a 296MHz.

En un intento por conseguir todavía mayores cotas de rendimiento, los arquitectos de computadores han hecho uso del creciente número de transistores disponibles para proponer mejoras de rendimiento todavía mayores. Muchas de estas propuestas han tenido como objetivo explotar una característica presente en toda aplicación, que se conoce como Paralelismo a Nivel de Instrucción (ILP). Toda aplicación está compuesta por una serie de instrucciones que se ejecutan de manera secuencial. Sin embargo, muchas de estas instrucciones no guardan relación entre ellas, por lo que podrían ser ejecutadas simultáneamente y obtener el mismo resultado que en una ejecución secuencial. La implementación de pipelines superescalares, que consiste en replicar las unidades funcionales que componen un microprocesador para ejecutar múltiples instrucciones en la misma etapa, ha permitido incrementar el número de instrucciones ejecutadas por segundo en cotas aún mayores a las ofrecidas por el aumento de frecuencia. El primer microprocesador superescalar comercial fue el Intel i960CA en el año 1989, con dos vías de

ejecución [4], mientras que hoy en día pueden verse microprocesadores con hasta seis vías de ejecución [5]. Finalmente, la posibilidad de utilizar cada vez un mayor número de transistores para su fabricación ha permitido dotar a los procesadores de mecanismos cada vez más inteligentes para incrementar todavía más las cotas de rendimiento. Un ejemplo claro de esto son las unidades de punto flotante o los pipelines con ejecución fuera de orden. Las unidades de punto flotante son dispositivos hardware diseñados específicamente para realizar operaciones con números reales (algo que anteriormente se debía emular a través de algoritmos numéricos vía software), permitiendo una ejecución significativamente más rápida. La ejecución fuera de orden aprovecha ciclos de ejecución que se desperdiciarían por algún tipo de parada (cuando se produce un fallo en el acceso a la memoria, por ejemplo) para ejecutar otras instrucciones, cambiando el orden del flujo de ejecución del programa sin cambiar su resultado. De esta forma se puede evitar que el procesador se quede sin trabajo y permite acortar el tiempo de ejecución de un programa. La ejecución fuera de orden fue introducida con el IBM POWER1 en 1990, exclusivamente para operaciones en punto flotante, mientras que el primer microprocesador en incorporar un pipeline completamente fuera de orden fue el PowerPC 601 [6], en 1993.

Desgraciadamente, gran parte de las técnicas empleadas en la mejora de rendimiento parecieron tocar fondo a comienzos de la década anterior. El ILP de las aplicaciones ha sido siempre limitado, incrementando de forma prohibitiva el coste de ejecutar más de 3-4 instrucciones en paralelo [7]. El aumento en la complejidad de los procesadores ha disparado el coste de test y verificación de los mismos, desaconsejando continuar aumentando su complejidad [8] [9]. Finalmente, el paulatino aumento de la frecuencia a lo largo de los años también ha contribuido a incrementar el consumo energético del microprocesador, llegando al punto de convertirse en el principal factor limitante del progreso [9]. Como ejemplo, la diferencia de rendimiento entre el Intel 4004 y el Pentium IV era, obviamente, muy elevada, pero al coste de incrementar el consumo energético desde los 0.6W hasta los 80W, es decir, más de 100 veces.

El consumo energético se ha convertido en un factor muy importante en el diseño y fabricación de microprocesadores. Esta importancia es debida a que un consumo de potencia excesivo puede tener consecuencias negativas en diferentes aspectos, todos relacionados con diferentes costes. El primero está relacionado con el coste de adquisición. A partir de cierto nivel de consumo (aproximadamente los 100W), la superficie del chip adquiere una temperatura demasiado elevada, lo que hace necesaria la utilización de sistemas de

refrigeración más eficientes que la refrigeración por aire, como la refrigeración líquida [10], cuyos sistemas son mucho más caros de instalar y de mantener. En cuanto al coste de utilización, No se limita únicamente a la cantidad de energía eléctrica que se necesita proporcionar a un chip para funcionar sino que además es necesario contabilizar la energía necesaria para refrigerarlo. Se estima que el coste energético necesario para refrigerar un procesador es equivalente al 30% del coste producido por el consumo energético del mismo [11] [12]. Dado el precio actual de la energía eléctrica, en los Centros de Procesado de Datos (CPD) actuales los costes derivados del funcionamiento de las máquinas superan cada vez más rápidamente aquellos derivados de la adquisición de los equipos. Finalmente, en tecnologías que buscan depender cada vez menos de la toma de corriente (teléfonos móviles, tablets, portátiles, etc.), este aspecto es especialmente importante y un reducido consumo energético es vital para aumentar el tiempo que el sistema puede funcionar antes de recargarse.

Continuar complicando el diseño del procesador para mejorar su rendimiento se volvió inviable debido a que el coste de diseñar nuevas mejoras era cada vez más alto para el escaso aumento del rendimiento. Además, aumentar la frecuencia ya no era una opción, como ya hemos visto. Surgió entonces la idea de intentar explotar un tipo de paralelismo diferente dentro del chip, a un nivel superior al de instrucción. En lugar de utilizar un único procesador muy complejo que intenta ejecutar muchas instrucciones simultáneamente, se divide el problema a resolver en varios de menor tamaño que se distribuyen entre varios procesadores más pequeños y sencillos. La inclusión de varios de estos procesadores (también llamados “cores”) dentro de la misma área de silicio dio lugar a los multi-procesadores (o CMP del inglés Chip-multiprocessor). La replicación dentro del chip permite aumentar la sencillez del diseño y la verificación a la par que reducir el consumo de energía comparado con el equivalente mono-núcleo en rendimiento. De acuerdo con la ley de Amdhal [13], un problema completamente paralelizado es capaz de ejecutarse en la mitad de tiempo si se utilizan dos procesadores, sin necesidad de aumentar la frecuencia de trabajo. Esta forma de extraer rendimiento de un chip ha permitido que los arquitectos hayan sido capaces de continuar reduciendo el tiempo de ejecución de las aplicaciones y ha provocado que la industria deje de competir por la mayor frecuencia del reloj para competir por el mayor número de procesadores.

1.1 Chip MultiProcessors

La integración de múltiples procesadores de propósito general en un único chip comenzó a utilizarse comercialmente en el año 2001, con la aparición del IBM Power 4 [14], un procesador de dos cores. En estos primeros modelos, se regresaba a frecuencias más bajas que en los últimos procesadores de un sólo núcleo (entorno a los 1-2GHz), intentando reducir el consumo energético del procesador. Los primeros CMP comerciales constaban de dos cores con dos niveles de memoria cache cada uno, un primer nivel de dos caches separadas y 32KB generalmente y un segundo nivel de un tamaño comprendido entre 1 y 4 MB. Aunque en un principio ambos niveles de cache eran privados para cada procesador y la conexión entre el segundo nivel de memoria cache de cada core se realizaba a través de un bus (FSB), rápidamente aparecieron arquitecturas en las que el segundo nivel era compartido para todo el CMP. Un ejemplo de esta primera generación de CMPs es la arquitectura Intel Core, que se comercializó a partir del 2006 y se fabricaba mediante un proceso de 65nm y 45nm [15]. Posteriormente, aparecieron los primeros CMP de cuatro y ocho cores fabricados en un único chip. Algunos de estos microprocesadores incorporaron un tercer nivel de memoria cache (aumentando el total de cache disponible dentro del chip), de forma que cada core disponía entonces de dos niveles de cache privados. Una cache L1 similar a la de generaciones anteriores, y una cache L2 generalmente de 256KB por cada core (de 1 a 2 MB en total). El tamaño de la cache L3, compartida entre todos los cores, variaba entre 4 y 12MB. La arquitectura Intel que sucedió a Core en 2008, llamada Nehalem [16], se caracterizaba, además de por lo ya descrito, por integrar algunas de las funcionalidades de la placa base, como los controladores de memoria principal, y por sustituir el bus por una red punto a punto como método de comunicación de todos los cores y los diferentes niveles de cache, ya que dicha red, a diferencia del bus, escala en ancho de banda con el número de cores [17].

Actualmente ya existen arquitecturas con un gran número de cores. Compañías como TILERA ya comercializan multiprocesadores con 36 y 64 cores [18] desde 2010 y ya han anunciado el lanzamiento de procesadores de hasta 100 cores. Estos multiprocesadores, enfocados en proveer alto rendimiento y un consumo energético eficiente, están compuestos por una malla de cores idénticos y muy sencillos y con entre 5 y 12MB totales de memoria cache que pueden permanecer selectivamente en un estado de bajo consumo en caso de que no haya trabajos pendientes. Por otra parte, Intel está desarrollando una nueva arquitectura conocida como TeraScale [19], también centrada en ofrecer chips con un gran número de cores que incorporan, entre ellos, hardware de propósito específico pensado para acelerar cargas de trabajo como encriptación y procesamiento gráfico. Además, también se persigue mejorar la

eficiencia energética, proporcionando la capacidad para desactivar y activar diferentes cores bajo demanda.

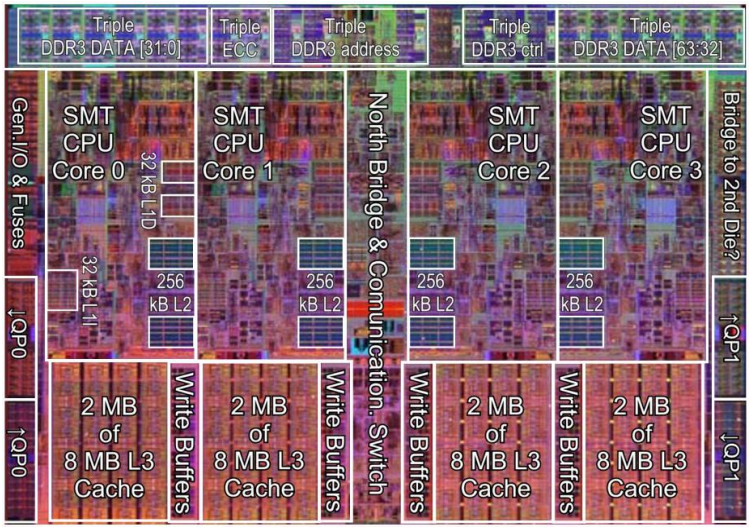

Figura 1 Layout de un procesador Intel Nehalem de cuatro cores.

Como se puede observar en la evolución pasada de los CMPs y en las proyecciones futuras, hay una tendencia clara a añadir en un chip cada vez más cores e incluso componentes hardware de propósito específico que antes no formaban parte de una CPU de propósito general. Este aumento del número de cores permitirá seguir extrayendo rendimiento del paralelismo a nivel de dato, pero no está exento de problemas graves que pueden limitar su desarrollo. Uno de los principales problemas a los que se enfrentan los CMPs es la imposibilidad de manejar el creciente tráfico de datos entre la memoria principal y el chip causado por un número creciente de procesadores.

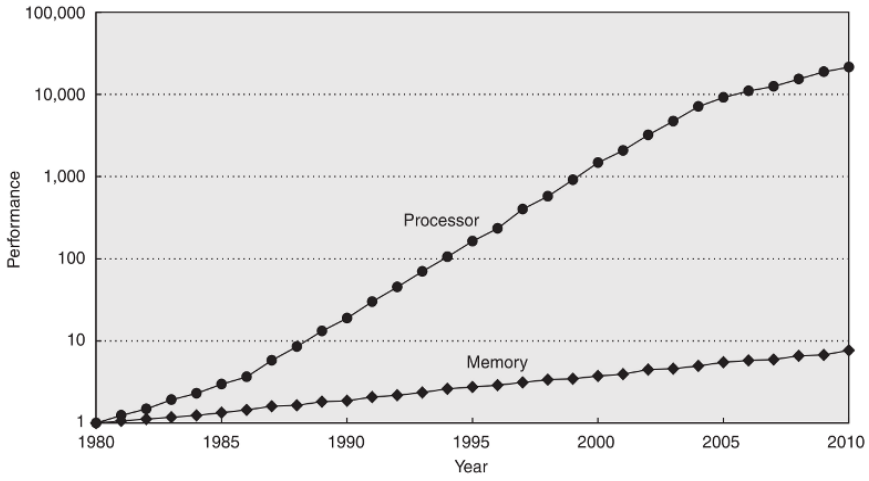

Figura 2 Evolución del rendimiento de procesadores y memorias tomando como referencia el año 1980.

1.2 El Memory Wall y el Bandwidth Wall

La velocidad de acceso a los datos ha supuesto, desde el comienzo, una limitación severa para el rendimiento de los computadores, ya que la evolución de la velocidad de los microprocesadores es mucho mayor que la de la velocidad de la memoria DRAM, utilizada actualmente en la memoria principal. Este límite se conoce como Memory Wall [20] y, como se muestra en la Figura 2, ha supuesto que la diferencia de rendimiento entre ambos no ha dejado de acentuarse con el paso del tiempo.

A este problema ya conocido se le añade un nuevo problema, el límite del ancho de banda. Con el aumento del número de cores, la cantidad de ancho de banda correspondiente a cada core es cada vez menor. Como en el caso de la velocidad, este ancho de banda mejora a un ritmo mucho más lento añadiendo una limitación mucho más difícil de evitar, conocida como Bandwith Wall [21]. La tendencia existente para tratar de evitar este problema tanto como sea posible consiste en añadir la mayor cantidad de memoria posible dentro del chip. De este modo, podemos desviar parte de esta demanda a memoria hacia el interior del CMP, donde se pueden conseguir mayores anchos de banda con menor dificultad. Sin embargo, diversos estudios demuestran que con el ritmo actual de crecimiento en número de procesadores, la memoria cache ocupará un área del chip cercana al 90% en unas pocas generaciones si se quiere mantener el mismo incremento de rendimiento [21].

Por suerte, en los últimos años han aparecido soluciones tecnológicas que probablemente permitirán aumentar de forma considerable la capacidad de la memoria en chip sin requerir aumentar la fracción de área necesaria para su ubicación. Una de las técnicas, conocida como 3D Stacking (o apilado vertical), permite la unión de múltiples capas de silicio, unas encima de otras. Esta técnica permitirá seguir incrementando el número de transistores en un chip incluso cuando la tecnología no permita reducir más su tamaño, simplemente apilando más y más capas en el eje vertical. Adicionalmente, el apilado vertical ha abierto las puertas a múltiples posibilidades en el diseño de computadores, como la reducción del área del chip o la utilización de diferentes procesos de fabricación en diferentes capas.

La combinación de múltiples procesos de fabricación incompatibles abre las puertas a tecnologías de almacenamiento en memoria más rápidas y eficientes que las actuales y que, hasta ahora, era imposible implementar. Así, nuevas celdas de memoria no volátiles, como la STT-RAM (Spin Torque Transfer Random Access Memory) y la PCRAM (Phase Change RAM), están abriéndose paso como firmes sucesoras de las tradicionales SRAM (Static RAM) en memoria cache y DRAM (Dynamic RAM) en memoria principal, ya que proporcionan

rendimientos muy similares mejorando la capacidad de almacenamiento y el consumo energético.

1.3 Contribuciones del Proyecto

Este trabajo se centra en el análisis del rendimiento de un sistema multiprocesador en chip, concretamente en los efectos que puede producir la red de conexión en el rendimiento de un CMP al introducir las soluciones tecnológicas descritas en el apartado anterior. Se trabajará con arquitecturas con varias capas apiladas verticalmente y dos tecnologías diferentes de almacenamiento de datos: SRAM y STTRAM. Se evaluará su rendimiento y se trabajara en la propuesta de soluciones para relajar alguno de los inconvenientes que conlleva utilizar celdas de memoria STT-RAM desde el punto de vista de la propia memoria.

A continuación se describe el contenido de los siguientes capítulos de los que consta este trabajo.

En el Capítulo 2 se explican de forma breve algunos conceptos que han sido necesarios para el desarrollo del proyecto. Se centra, principalmente, en el funcionamiento de las redes de interconexión y de las celdas de memoria, tanto la tecnología utilizada hoy en día, como las alternativas más importantes que han surgido últimamente. También se explica en qué consiste el apilado vertical, que permite combinar los dos conceptos anteriores para desarrollar el hilo principal del trabajo.

En el Capítulo 3 se realiza una descripción del sistema que vamos a utilizar y justifica el motivo por el que se han establecido los parámetros que lo caracterizan. Asimismo profundiza en los problemas que implica la utilización de celdas STTRAM.

El capítulo 4 contendrá la propuesta principal elaborada durante el proyecto. Comienza analizando una propuesta existente que trabaja sobre el mismo campo que este proyecto y a continuación se explicarán varias propuestas que se han desarrollado.

El capítulo 5 incluye otras propuestas que no se han evaluado en profundidad pero que han ido surgiendo mediante la observación y comprensión de las características del sistema que estamos utilizando.

Finalmente el capítulo 6 describe las conclusiones más importantes extraídas de los resultados que se han obtenido e indica diferentes formas de ampliar el trabajo desarrollado.

Capítulo 2. Background

Dada la complejidad de los temas que se van a tratar, explicaremos a continuación los conceptos más importantes, permitiendo al lector comprender con claridad la totalidad de los razonamientos que se exponen.

2.1 3D Stacking

La mejora continuada del rendimiento de los transistores ha producido que elementos invariantes como las interconexiones se hayan convertido ahora en un cuello de botella difícil de superar. Con cada nueva generación tecnológica, la reducción de tamaño de las vías de comunicación dentro del chip hace que las comunicaciones sean cada vez más costosas. El tiempo necesario que requiere una señal para atravesar el chip no ha dejado de crecer con cada reducción de la tecnología. Adicionalmente, el consumo energético disipado por los cables supera el 30% [22] del consumo de un microprocesador.

El apilado vertical reduce drásticamente las distancias entre componentes, dado el minúsculo grosor de cada capa (nm). Esto permite reducir la cantidad de interconexiones de un circuito integrado gracias a la agrupación de varias capas, unas encimas de otras, conectadas entre sí mediante conexiones verticales. Ya no será siempre necesario atravesar el chip en horizontal sino que también se hará en vertical, reduciendo de forma notable la latencia y el consumo energético de las comunicaciones. Por último, el uso de múltiples capas con tecnologías de integración heterogéneas permite a nuevos procesos de fabricación, hasta ahora incompatibles con CMOS, formar parte de la arquitectura de un procesador [23]. A pesar de los numerosos beneficios que aporta, el apilado vertical también trae consigo nuevos retos que deberán ser superados para poder sacarle el máximo partido. Un ejemplo claro es la disipación térmica, ya que con la agregación de nuevas capas es más difícil extraer el calor de aquellas capas más lejanas al disipador.

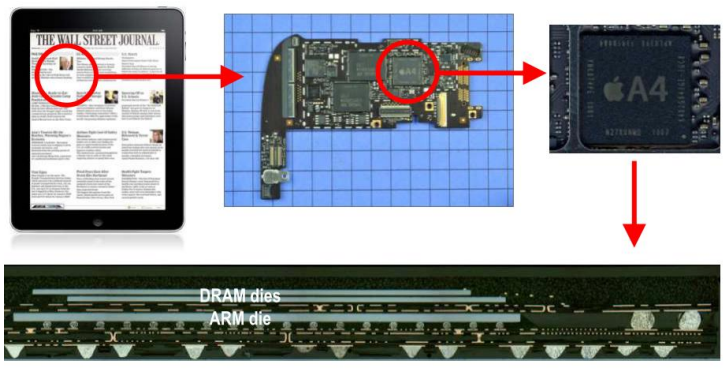

Figura 3 Procesador y memoria conectados mediante el uso del apilado vertical.

Actualmente podemos encontrar en el mercado múltiples productos que hacen unos de apilado vertical en alguno de sus componentes. En el año 2008 ya existían componentes comerciales con varias capas, como los sensores de imagen CMOS de Toshiba [24]. En los últimos años son cada vez más los módulos de memoria principal que se fabrican usando tecnologías 3D [25] [26] y en sistemas portátiles, como tablets, teléfonos y PDAs por ejemplo; se pueden encontrar memorias y procesadores empaquetados conjuntamente y unidos mediante cables (

Figura 3). Existen múltiples técnicas diferentes de apilado, con distintas características y diferentes niveles de madurez. A continuación describiremos aquellos que han tenido mayor relevancia en los últimos años.

Enlace mediante cables

La forma más conservadora de apilado consiste en conectar cada chip con cables uniendo sus superficies. En esta técnica los chips no son físicamente apilados unos encima de otros, por lo que también se la conoce como 2.5D stacking. Lo más habitual es que las conexiones vayan de un chip a otro pasando por una placa, aunque también es posible enlazar dos chips directamente. Esta solución está limitada por el área disponible para realizar la soldadura de los cables al chip. Un número demasiado grande de conexiones en un chip provocaría que un área desorbitada se ocupe únicamente por las conexiones de los cables (Un cable de 15 micras necesita soldarse a una base de, al menos, 35x35µm). Además, estas conexiones pueden hacerse únicamente en los bordes del chip, como se puede observar en la Figura 4, de forma que la densidad de integración puede no ser suficiente.

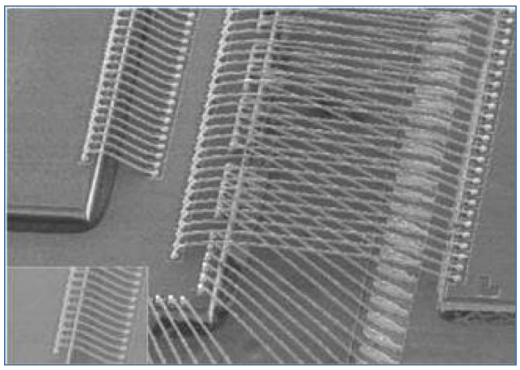

Figura 4 Comunicación de varios circuitos integrados utilizando cables.

Micro bultos

Los microbultos utilizan soldaduras o bultos de oro u otros materiales en la superficie de un chip para hacer las conexiones. En esta técnica se conoce como empaquetado 3D, ya que los

microbultos se encuentran en el exterior del chip. En este caso sí que existe apilado vertical, siendo el resultado un cubo compacto con todos los chips unidos. Su ventaja sobre los cables es una densidad de conexiones mucho mayor, pero el calor generado dentro del cubo impide la implementación de demasiadas capas, que dificultarían la refrigeración.

Sin contacto

Este tipo de unión se aprovecha del acoplamiento inductivo o capacitivo para comunicarse entre niveles, evitando la necesidad de cualquier proceso de fabricación especial para proporcionar conectividad entre ellos. Esto reduce el coste de fabricación con respecto a la conexión mediante microbultos o cables.

El acoplamiento capacitivo [27] se basa en construir un condensador a partir de la capa metálica superior de dos chips. La densidad de estas interconexiones depende de factores tales como la distancia entre las capas o el material dieléctrico que las separa y ha demostrado ser del mismo orden de magnitud que la tecnología CMOS en la que se incluye. El principal problema de este tipo de unión es que está limitado a dos niveles, debido a que necesita que ambos niveles estén cara a cara (última capa metálica de cada chip enfrentada) para acercar las placas de los condensadores lo más posible.

Es posible sustituir los condensadores por inductores para comunicar varias capas [28]. El acoplamiento inductivo es más adecuado cuando la separación de los elementos acoplados es excesiva y las capacidades resultantes serían demasiado elevadas. Esto permite la inclusión de más de dos capas, ya que no deben estar enfrentadas cara a cara. Sin embargo, su densidad es menor que la del caso capacitivo, dado que las espirales metálicas que forman el inductor (Figura 5a) tienen un tamaño considerablemente mayor que el de los enlaces capacitivos.

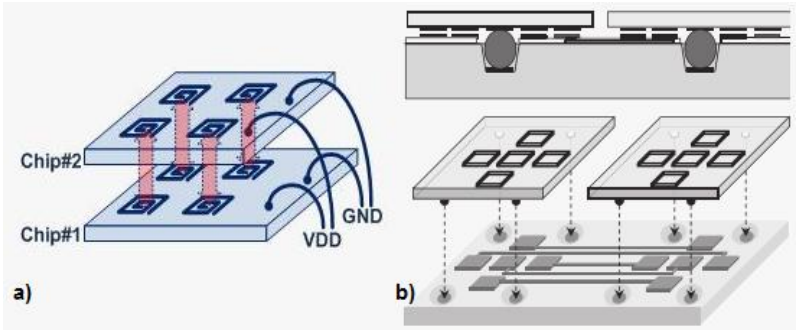

Figura 5 Apilado vertical sin contacto. a) Inductivo. La alimentación se provee utilizando cables. b) Capacitivo, usando bultos enterrados para alimentar el chip.

Pilares metálicos entre capas (TSV)

La interconexión a través de pilares (through-silicon-via (TSV)) consiste en la comunicación de múltiples capas mediante de vías de material conductor que las atraviesan. El montaje se realiza colocando unas obleas sobre otras, grabando los agujeros necesarios a través de la oblea superior dentro de la inferior para rellenarlos con material metálico y uniendo las obleas mediante termocompresión. Con el proceso conocido como Silicon-on-insulator no se necesita de aislamiento entre capas, por lo que es posible conseguir la mínima separación entre niveles, de apenas 5µm [29]. El principal límite al número de niveles, tal y como ocurre con los microbultos, también está dictado por la capacidad de disipación del calor generado y no por el proceso de fabricación.

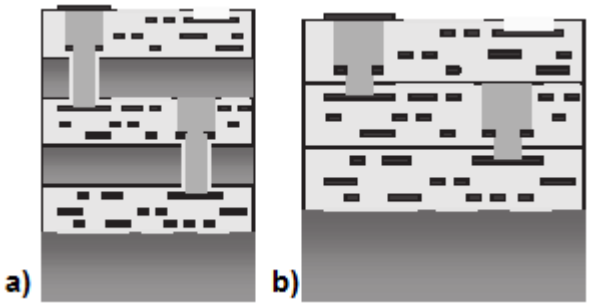

Figura 6 Bloque de circuitos integrados pegados mediante termocompresión y comunicados mediante TSV. El uso de litografías de volumen (a) necesita de un aislamiento previo de las vías. La integración mediante Silicon-on-Insulator (b) consigue la menor distancia entre capas.

TSV es uno debido a que permite una mayor densidad de interconexiones, llegando a conseguir separación entre vías de tan sólo 4-10µm [30]. Además, se ha demostrado que la latencia producida al atravesar 20 capas es de tan sólo 12ps [31].

En sus orígenes, esta tecnología estaba limitada por el excesivo tamaño de los pilares metálicos. El diámetro de una de estas vías podía llegar a ocupar hasta 10 veces el tamaño de una celda de memoria convencional. Esto reducía su densidad, ya que la distancia entre pilares adyacentes debía ser muy elevada para evitar problemas de acoplamiento o crosstalking. Afortunadamente, el diámetro de estos pilares se ha reducido drásticamente, hasta diámetros inferiores a los 5 µm [30]. Esta densidad, unida a la ínfima latencia entre capas (apenas 1ps), hacen de los TSVs de los tipos de interconexión vertical más prometedores. De hecho, gran parte de los chips de memoria DRAM actuales utilizan esta tecnología para aumentar su capacidad. Por estas razones, esta será la tecnología escogida para los estudios realizados en este proyecto.

2.2 Nuevas tecnologías de almacenamiento

La jerarquía de la memoria es un aspecto muy importante a la hora de diseñar computadores. Por lo general, según nos acercamos al procesador se utilizan memorias con menores latencias pero mayor tamaño por bit. En una jerarquía convencional podemos diferenciar tres niveles de almacenamiento, ordenados de menor a mayor capacidad y latencia:

Memoria cache: son las memorias alojadas dentro del procesador. La capacidad de almacenamiento es del orden de MegaBytes y están construidas con celdas de memoria SRAM (static random access memory).

Memoria principal: se sitúan fuera del chip y se implementan utilizando tecnología DRAM, que permite una capacidad de almacenamiento del orden de GigaBytes.

Disco duro: Es el nivel más lento, con la mayor capacidad. Sus latencias son más de 1000 veces superiores a las de la memoria principal, pero su capacidad de almacenamiento también aumenta, prácticamente, en esa proporción.

En este proyecto nos centraremos en el nivel de la jerarquía más cercano al procesador, la memoria cache. Para comprender algunos de los problemas que han surgido y que se intentarán resolver en este trabajo, es necesario conocer cómo funcionan las celdas de memoria utilizadas en las memorias cache de los procesadores.

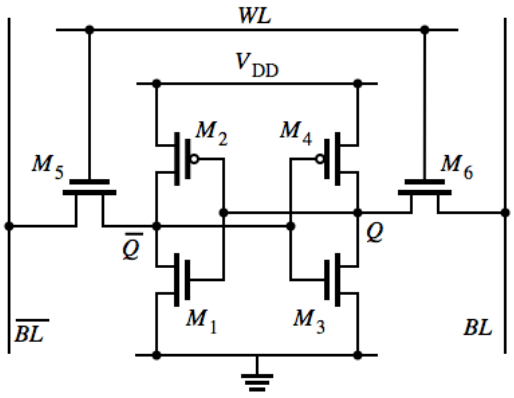

Figura 7 Estructura de una celda SRAM.

SRAM es el acrónimo de *Static Random Access Memory*. Básicamente, una celda SRAM es un flip-flop que mantiene el valor que se ha escrito en ella. Su estructura se puede observar en la Figura 7. Los dos procesos básicos sobre este tipo de celda son la lectura y la escritura de un dato. El proceso de escritura comienza mediante la activación de la señal *Wordline* (*WL*), que

permite el acceso de los valores de ambas BitLines a la celda a través de dos transistores. Si queremos almacenar un "1" entonces el valor de *BL* y $\overline{BL}$ será "1" y "0" respectivamente o, en el caso de querer almacenar un "0", se intercambiarían ambos valores. En el caso de las lecturas, los BitLines se cargan con el valor almacenado, y mediante un amplificador se detecta la polaridad de la diferencia de tensión existente entre las líneas *BL* y $\overline{BL}$.

A pesar de su alto rendimiento, este tipo de celdas presenta un problema fundamental, su tamaño. Esto se debe a que cada celda de memoria necesita 6 transistores en total. Por lo tanto, el área de cada una de ellas es mayor que, por ejemplo, las celdas DRAM, que utilizan un único transistor y un condensador. Como vimos en la introducción, la limitación del ancho de banda disponible a memoria fuerza a los procesadores a incluir cada vez mayores capacidades de memoria dentro del chip, lo que puede resultar bastante complicado para celdas de este tipo. Por lo tanto, para suavizar el problema del memory wall, necesitaremos una nueva tecnología que proporcione mayor densidad de integración y que posea un rendimiento similar o superior.

Actualmente existen numerosas propuestas tecnológicas para construir celdas de memoria. En esta sección describiremos solamente aquellas que poseen las características necesarias para poder formar parte de la jerarquía de cache de las próximas generaciones de procesadores. La mayor parte de los nuevos tipos de memorias son no volátiles. Se están desarrollando nuevas ideas y materiales centrados en sustituir a todos los tipos anteriores de memoria, consiguiendo alta capacidad de almacenamiento, bajo consumo energético y transferencias muy rápidas.

FeRAM

En este tipo de celdas, los datos se almacenan como la polaridad de un material ferroeléctrico. Estos materiales se caracterizan por poseer dos estados con distinta polarización. Esta polarización puede ser intercambiada mediante un campo magnético externo, la cual se mantiene incluso tras eliminar dicho campo. Si utilizamos un condensador ferroeléctrico, se pueden conseguir dos estados en función del campo eléctrico aplicado en sus placas. Estos dos estados pueden corresponder a almacenar un "0" ó "1". Utilizando ese condensador se puede construir una celda de memoria similar a la de tipo DRAM.

La principal desventaja de esta tecnología es que el tamaño de celda es al menos el doble que el de una celda típica DRAM, por lo que su densidad de integración no es muy elevada. Otra desventaja es que a pesar de tratarse de celdas con lectura no destructiva (a diferencia de las

DRAM), los tiempos de retención que se han alcanzado no son suficientes como para utilizarlos en memorias no volátiles.

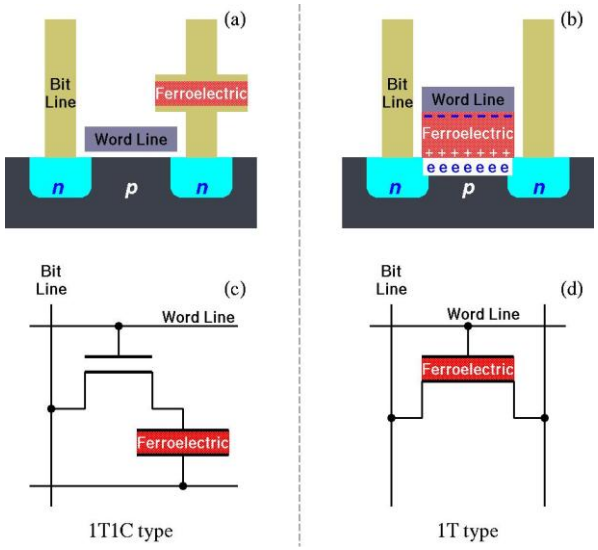

Figura 8 Diferentes estructuras de una celda FeRAM

MRAM

El cambio de la resistencia de un material debido a su exposición a un campo magnético se conoce como magnetorresistencia. El índice de magnetorresistencia es el cociente entre las resistencias mínima y máxima de un material. Todos los tipos de memoria MRAM utilizan la dirección de la magnetización para almacenar información y la diferencia de la resistencia eléctrica resultante para la lectura de dicha información.

En los primeros tipos de magentoresistencias su índice presentaba valores muy bajos (en torno al 0.5%) [32], lo que dificultaba la lectura de un dato dada la escasa diferencia de resistencia entre un 0 y un 1. La evolución hacia una nueva estructura conocida como magnetorresistencia gigante permitió incrementar de forma significativa las diferencias de resistencia y disminuir el tiempo de lectura. De hecho, los discos duros actuales se fabrican con este tipo de estructura. En los últimos años se han seguido optimizando este tipo de memorias, con estructuras con índices todavía mayores que permiten operaciones más rápidas, conocidas como magnetic tunnel junction (MTJ).

Las celdas construidas con MTJ presentan unas características muy favorables y son un posible sustituto de las celdas de memoria cache. Por esta razón serán objeto de estudio en este proyecto y una descripción más detallada de sus características se proporciona en la sección 2.3 .

Phase Change RAM (PCRAM)

Las celdas de memoria PCRAM hacen uso de las características únicas de unos materiales conocidos como cristales de calcógeno. Se conoce como calcógeno al grupo de elementos de la tabla periódica formado por Oxígeno, Azufre, Selenio, Teluro y Polonio. En estos materiales, el calor producido por una corriente eléctrica cambia el material entre dos estados, cristalino y amorfo. En la fase cristalina este material es altamente conductivo, mientras que en la fase amorfa presenta una resistencia muy elevada.

Este tipo de memorias presentan la gran ventaja de poder almacenar más de un bit por celda, gracias a su capacidad para funcionar a múltiples niveles de resistencia. Sin embargo, uno de sus principales problemas es la cantidad de densidad de corriente necesaria para realizar los cambios de estado (10^7 A/cm^2 comparado con los 10^5A/cm^2 de un transistor).

Conductive Bridge RAM (CBRAM)

Una celda CBRAM está compuesta por dos electrodos metálicos entre los que se introduce un electrolito sólido (material que contiene iones libres). Una capa metálica (usualmente de plata o cobre) en contacto con el electrolito forma un elemento en donde la información se almacena gracias a cambios eléctricos, producidos por la oxidación de dicho metal y la reducción del número de iones en el electrolito. Estas reacciones provocan la creación o eliminación de un canal conductor eléctrico entre ambas capas metálicas, gracias al agrupamiento o dispersión de los iones libres del electrolito en función de la corriente eléctrica aplicada.

2.3 Spin-torque-transfer RAM (STT-RAM)

A diferencia de tecnologías más tradicionales como SRAM o DRAM que utilizan cargas eléctricas para almacenar la información, la celda STT-RAM utiliza una unión túnel-magnética (MTJ) para ese cometido. Una unión MTJ consiste en dos capas ferromagnéticas (materiales con los momentos magnéticos ordenados) separadas por un material aislante de muy poco grosor denominado túnel. Una de las capas ferromagnéticas (capa de referencia) está diseñada para mantener su polaridad de forma permanente mientras que la otra (capa libre) puede ser cambiada mediante una corriente que la atraviesa. La dirección relativa de la magnetización de ambas capas determina el valor de resistencia de la unión MTJ. Si ambas capas tienen la misma polaridad la resistencia será baja, representando un "0"; si tienen polaridades opuestas la resistencia será alta, representando un "1".

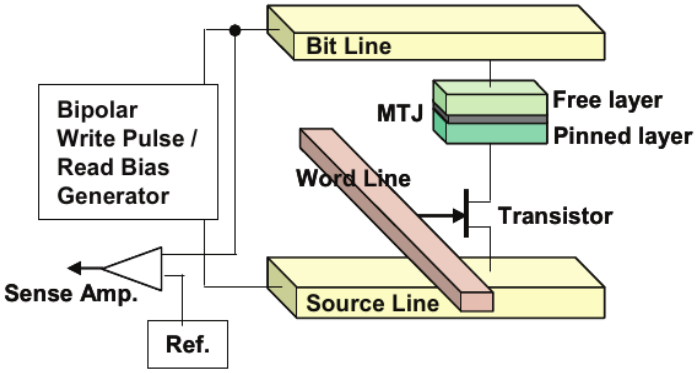

Figura 9 Esquema de la estructura de una celda STT-RAM.

A diferencia de los anteriores diseños de MRAM, que utilizaban campos magnéticos externos, la STTRAM hace circular corriente con spin polarizado a través del MTJ como método para cambiar la polarización de la capa libre. Gracias a este procedimiento se consigue reducir en gran medida la corriente necesaria para cambiar la polarización (aproximadamente de 10mA a 100-200uA). La estructura STTRAM más utilizada, mostrada en la Figura 9, está compuesta por un transistor NMOS, utilizado para acceder a la celda, y un MTJ, para el almacenamiento. Esta estructura también se conoce como 1T-1J y simplemente conecta el MTJ al transistor en serie. A continuación se detalla el procedimiento a seguir para leer y escribir datos en una celda STTRAM.

Lectura: Se aplica una tensión negativa entre la bit-line y la source-line, produciendo una corriente eléctrica que circula desde la a través del MTJ. Dicha tensión es pequeña (generalmente de -0.1V), de forma que la corriente generada no provoque una escritura accidentalmente. El valor de dicha corriente depende de la resistencia del MTJ y se mide con un amplificador de medida que la compara con una referencia. Dependiendo del valor de dicha corriente con respecto a esa referencia se deduce si en la celda hay un "0" o un "1" almacenado.

Escritura: Para escribir un "0" se aplica una tensión positiva (normalmente 1.2V) entre la SL y la BL, generando una corriente que circula desde la SL hasta la BL. En el caso de escribir un "1", la tensión utilizada es negativa (normalmente -1.0V) creando una corriente que circula en dirección opuesta a la anterior. La intensidad de corriente umbral del pulso de escritura es inversamente proporcional al tiempo, de forma que pulsos de escritura cortos necesitan mayor intensidad que pulsos de mayor duración [33]. En cualquier caso, los valores de las corrientes generadas son significativamente mayores que los utilizados en las operaciones de lectura y su duración también es sustancialmente mayor.

La velocidad a la que la capa libre puede cambiar su polaridad (escritura de un dato) viene determinada por la cantidad de corriente que se hace circular por la unión MTJ. A mayor cantidad de corriente, menor será el tiempo necesario para realizar el cambio de valor. La Figura 10 muestra el número de ciclos necesarios (a 2GHz) para realizar una escritura a diferentes intensidades de corriente. A la vista del gráfico, lo ideal sería aplicar la máxima corriente posible para escribir en el menor número de ciclos posibles. Sin embargo, a medida que aumentamos la intensidad, mayor deberá ser el tamaño del transistor NMOS de la celda, por lo que nuestro nivel de integración disminuirá. La Figura 11 muestra la relación entre el tamaño de una celda SRAM y una STT-RAM a diferentes intensidades. Por tanto, existe una relación muy estrecha entre la densidad de integración y la velocidad de lectura, que será analizada con más detalle en este proyecto.

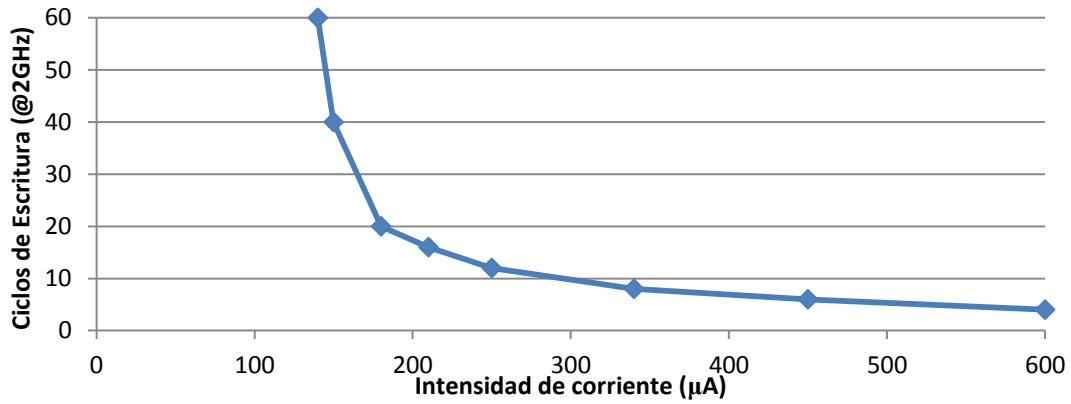

Figura 10 Duración (en ciclos de un procesador a 2GHz) de una escritura en una celda STT-RAM, en función de la intensidad de corriente necesaria en µA.

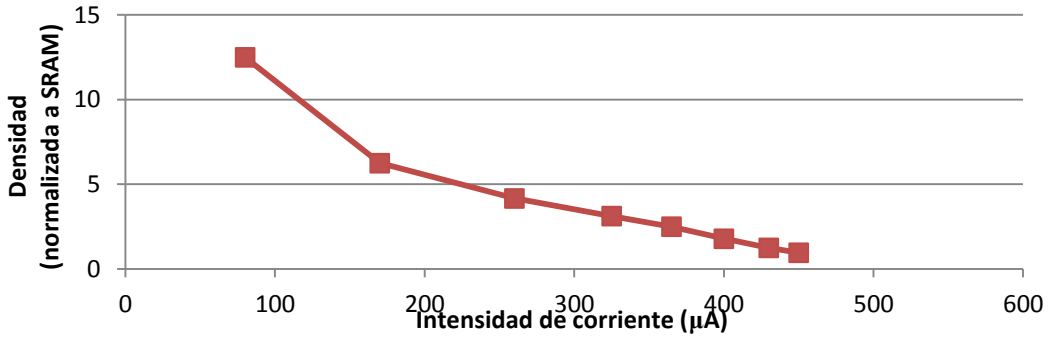

Figura 11 Densidad máxima de integración alcanzable dependiendo de la intensidad de corriente utilizada.

Las memorias STT-RAM han recibido mucha atención en los últimos años [34] y parece ser una de las mejores candidatas para sustituir a la SRAM como memoria cache de último nivel [35], debido a su mínimo consumo de potencia estático [36], la densidad de integración que ofrece y los tiempos de acceso que se pueden conseguir [37]. Por esta razón, el trabajo

realizado en este proyecto intentará analizar la viabilidad de esta tecnología como cache de último nivel en arquitecturas de procesador con apilado vertical.

2.4 Redes de Interconexión

Como hemos podido observar, el aumento continuo de cores en un microprocesador es una realidad. Para continuar incrementando el rendimiento, es necesario disponer de una red escalable que los conecte y que proporcione el ancho de banda necesario. En caso contrario, el ancho de banda se iría reduciendo proporcionalmente, razón por la que redes no escalables, como los buses, están quedando obsoletas.

El uso de sistemas de comunicación centralizados, como buses o crossbars, es solamente una opción en casos en que el número de nodos no sea excesivamente grande. Los sistemas basados en bus tan sólo escalan en un número reducido de cores, ya que alcanza rápidamente la saturación según se incrementa la demanda. Los crossbars solucionan el problema de ancho de banda de los buses y han sido utilizados como interconexión para un número pequeño de cores, como ocurre, por ejemplo, en el procesador UltraSparc T2 [38]. Sin embargo, para redes con mayor número de terminales, el número de conexiones necesarias aumenta de tal manera que es más rentable el empleo de otras alternativas.

Las redes de interconexión en chip son una buena alternativa a los buses y los crossbars por múltiples razones. Representan una solución más eficiente, debido a que su ancho de banda escala a menor coste de consumo y área. Las redes de interconexión con topologías regulares poseen enlaces locales y cortos con una longitud fija, lo que facilita de forma significativa el proceso de *place and route* de estos enlaces metálicos. Finalmente, cada router de la red posee una estructura idéntica al resto, lo que permite optimizar su diseño y facilita los procesos de implementación y verificación. Debido a las ventajas que ofrecen, cada vez más micro-arquitecturas optan por el uso de estas redes de interconexión. Los multiprocesadores de TILERA, con modelos desde 16 hasta 64 cores, utilizan una topología (forma de conectar los diferentes nodos) de malla 2D para interconectar todos sus cores [39]. Se puede ver un ejemplo de una malla similar en la Figura 12b.

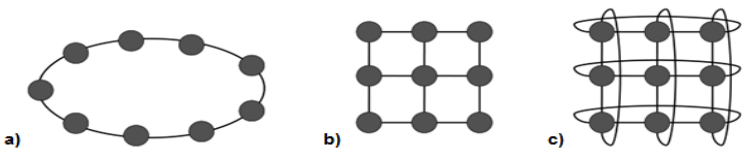

Figura 12 Redes de interconexión más comunes: Anillo (a) Malla (b) y Toro (c)

Por lo general, las topologías en malla y en toro (Figura 12c) son las más utilizadas, ya que poseen un número reducido de enlaces entre nodos comparado con otro tipo de redes de interconexión (cuatro para una red bidimensional como la de la figura). Adicionalmente, su estructura regular hace que todos los enlaces posean la misma longitud y minimiza el número de enlaces que se cruzan, pudiendo construir toda la red en una sola capa de metal (malla) o en dos como máximo (toro).

En el caso de las redes en chip, en las que la frecuencia de operación es muy elevada, las operaciones a realizar dentro del router deben ser lo más rápidas posibles. Para decidir la ruta de un mensaje entre origen y destino, se suelen utilizar algoritmos deterministas, en donde los paquetes siempre recorren el mismo camino para un origen y un destino dados. El encaminamiento adaptativo, en el que un mensaje puede elegir diferentes rutas para alcanzar su destino, es una técnica poco utilizada en este tipo de entornos dada su complejidad de implementación. La forma más sencilla de encaminamiento es utilizar un enrutamiento ordenado por dimensión (DOR) que, para una malla 2D por ejemplo, dirige los mensajes primero en la dimensión X y luego en la dimensión Y.

Finalmente, es necesario un control de flujo para evitar la pérdida de información por desbordamiento. El control de flujo se encarga de gobernar el acceso de los mensajes a los buffers de cada router. Las redes de conmutación de paquetes (las más comunes en redes en chip) realizan la reserva de recursos a medida que el paquete avanza, siendo solamente necesario obtener espacio de almacenamiento en el siguiente router hacia destino. Los recursos pueden ser reservados para almacenar un paquete completo, como en el control de flujo VCT [40]. También podemos fragmentar los mensajes en porciones más pequeñas denominadas flits si el canal de comunicación no es suficientemente ancho para el mensaje completo. El control de flujo Wormhole [40] realiza la reserva de buffer a nivel de flit. Este último control de flujo necesita menor espacio de almacenamiento para funcionar, por lo que es el utilizado con mayor frecuencia en las redes en chip con el fin de minimizar el área ocupada por los routers de la red.

Capítulo 3. Motivación

Debido al problema del memory wall, la memoria se ha convertido actualmente en un cuello de botella importante del sistema, limitando su rendimiento global. El aumento tan rápido del número de procesadores en un mismo chip que se ha producido a lo largo de los últimos años ha provocado un incremento del volumen de comunicaciones con el exterior del chip, mientras que el ancho de banda disponible para dichas comunicaciones apenas ha variado. Dicho incremento ha producido que el memory wall se vea agravado por la falta de ancho de banda suficiente para atender a todo el tráfico, problema conocido como bandwidth wall [41].

La importancia de esta limitación puede ponerse de manifiesto de forma mucho más clara con un ejemplo gráfico. En el trabajo de [41], se analizaba la cantidad de área del chip que debería ser dedicada a procesadores y caches con el fin de mantener cotas de rendimiento similares a las actuales. En este estudio se asumía una evolución del aumento de ancho de banda a memoria similar a la actual, y se obtuvieron los resultados que se muestran en la Figura 13. El eje X de este gráfico representa cuántos procesadores del tipo actual se podrían fabricar en el mismo área gracias a los avances tecnológicos. Así, si consideramos un procesador actual con 8 cores como punto de partida, dentro de 4 generaciones (el punto 16X) se podrían fabricar en el mismo área 128 cores equivalentes. Sin embargo, como muestra el eje Y, para mantener las cotas de rendimiento actuales, los procesadores de cada generación deberán incluir más memoria dentro del chip. Así, la realidad es que en 4 generaciones solamente el 10% del área se podrá dedicar a implementar procesadores, y por tanto su número será de unos 24, un valor bastante lejano a 128.

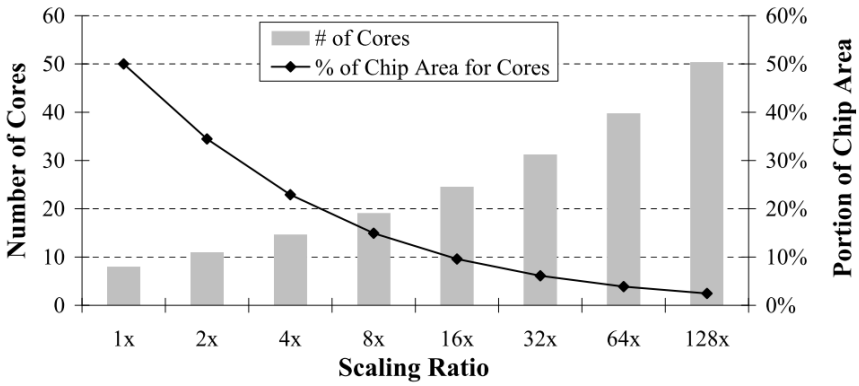

Figura 13 Evolución del número de cores y la proporción del área del chip ocupada por los mismos según el escalado progresa.

Este trabajo dejaba patente que los problemas de ancho de banda limitarían de forma significativa el crecimiento en número de procesadores. Esto ha enfocado el trabajo de muchos investigadores hacia la búsqueda de tecnologías que permitan paliar el problema. Como el crecimiento del espacio de almacenamiento en chip parece inevitable, muchas de las soluciones propuestas han tenido como objetivo mejorar la densidad de integración de la jerarquía de memoria en chip. Tanto el 3D Stacking como la utilización de nuevas tecnologías de memoria son dos claros ejemplos de esto. En este capítulo realizaremos un primer análisis de cómo pueden mejorar el rendimiento de un sistema las tecnologías propuestas. Adicionalmente, nos centraremos en el caso de las tecnologías de memoria para analizar también los problemas a los que se enfrentan, para en posteriores capítulos proponer diversas soluciones.

3.1 Herramientas de Simulación

Antes de continuar con nuestro trabajo, consideramos apropiado incluir un breve apartado en el que se describirán las herramientas utilizadas para la realización del proyecto. A pesar de lo escueto de la descripción, los simuladores son esenciales en la investigación en arquitectura de computadores, y el grueso del proyecto se ha dedicado al aprendizaje y manejo de este software.

En la mayoría de los casos, la investigación en arquitectura de computadores hace uso de una o varias herramientas de simulación. Estas herramientas deben cumplir dos requisitos fundamentales. Primero, el nivel de detalle debe ser suficientemente alto. Esto nos permitirá evaluar la interacción de los componentes del sistema para evitar conclusiones erróneas. Por ejemplo, las aplicaciones multi-thread dependen de múltiples servicios del sistema operativo, por lo que nuestro hardware simulado debería ser capaz de ejecutar un sistema operativo sin modificar. Segundo, las simulaciones deben analizar el rendimiento de aplicaciones realistas. Estas aplicaciones deben representar software que pueda estar presente en cualquier equipo comercial, ya sea de cálculo o incluso de sobremesa.

En la Universidad de Wisconsin-Madison se ha desarrollado una herramienta de simulación que cumple con estos requerimientos. GEMS [42] es un simulador que permite realizar simulaciones de sistema completo, permitiendo ejecutar un sistema operativo sin modificar sobre el hardware simulado. Utilizando SIMICS como núcleo, GEMS proporciona módulos que describen con gran detalle desde el procesador hasta los diferentes elementos de la jerarquía

de memoria. La Figura 14 proporciona un esquema de la organización de la infraestructura de simulación completa.

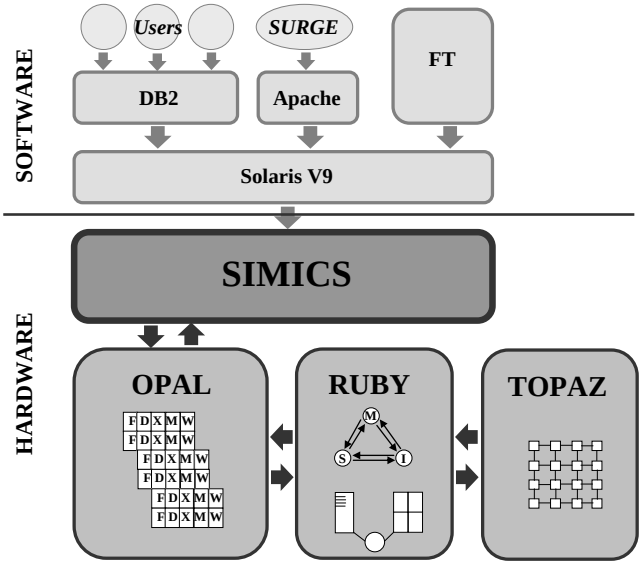

Figura 14 Infraestructura de simulación utilzada para elaborar el trabajo.

Los dos módulos principales de GEMS son RUBY y OPAL. RUBY modela los diferentes componentes de memoria, como las caches, los controladores de cache y memoria y la red de interconexión. Por su parte, OPAL es capaz de simular procesadores de gran complejidad, con pipelines superescalares con planificación dinámica y ejecución fuera de orden. A pesar de su flexibilidad, los modelos de RUBY para la red de interconexión carecen del nivel de detalle necesario para el trabajo realizado en este proyecto. Por esa razón un módulo adicional, llamado TOPAZ, se ha añadido al simulador, para reemplazar a los modelos de red originales de RUBY. TOPAZ proporciona una descripción muy cercana al hardware y es altamente configurable. Dado que la mayoría de las propuestas hechas en este proyecto están implementadas sobre diversos componentes de la red de interconexión, el trabajo se programación realizado en estos meses se ha centrado en crear nuevos componentes en TOPAZ para simular los mecanismos descritos en posteriores capítulos.

Dado que SIMICS nos permite utilizar un Sistema Operativo en nuestras simulaciones, podremos hacer uso de aplicaciones de uso común en computadores actuales, aumentando el valor de nuestros resultados. Dada la gran diversidad de las aplicaciones actuales, conseguir un conjunto significativo es siempre complicado. Por esta razón, es común el uso de colecciones de benchmarks especializadas, que cubren diversos campos. En nuestro caso nos centraremos en dos de estas suites. La primera, desarrollada por la Universidad de Wisconsin, contiene aplicaciones de tipo servidor [43], bien de aplicaciones bien de

contenido web. El segundo benchmark se centra en computación científica. Las aplicaciones NAS en su versión paralela [44] resuelven problemas numéricos relacionados con la dinámica de fluidos. La Tabla 1 muestra con mayor nivel de detalle las diferentes aplicaciones evaluadas en cada benchmark.

Aplicación	Descripción
Wisconsin Commercial Workload suite	
Apache	**Servidor web de tareas en paralelo**
Jbb	**Aplicación middleware de Java**
Zeus	**Servidor web de alto rendimiento**
Oltp	**Procesamiento de transacciones en línea**
NAS parallel benchmark	
FT	**Resolución de ecuaciones parciales diferenciales utilizando la transformada rápida de Fourier**
IS	**Ordenación de enteros**
SP	**Resolución escalar pentadiagonal**
LU	**Resolución de LU**

Tabla 1 Colecciones de benchmarks utilizados.

3.2 Organización de un CMP con múltiples capas y STT-RAM como cache de último nivel

Nuestro punto de partida será una arquitectura de procesador de próxima generación, que incluye apilado vertical y tecnología STT-RAM para la implementación del último nivel de cache. A continuación proporcionamos una descripción detallada de los componentes del sistema.

Procesador: El CMP está compuesto por 16 cores idénticos funcionando a una frecuencia de 2GHz. Existen arquitecturas con un mayor número de cores como, por ejemplo, las gamas más altas de procesadores de TILERA con hasta 64 cores, pero esto es debido a que se trata de cores extremadamente sencillos. Arquitecturas de propósito general más habituales, como Sandy Bridge de Intel, POWER de IBM y Bulldozer de AMD, poseen un número de cores alrededor de 8. Entendemos que, continuando con la progresión, la comercialización de procesadores de 16 cores será una realidad en no mucho tiempo.

Jerarquía de Memoria: Nuestra arquitectura utiliza una cache de dos niveles. Un primer nivel de cache de SRAM privado, dividido en datos e instrucciones. Con 32KB y un tamaño de bloque de 64B, este primer nivel está optimizado para que el tiempo

de acceso a la cache sea de 1 ciclo de procesador. El segundo nivel de cache SRAM posee 8MB de capacidad total, compartido entre todos los procesadores. Con el fin de reducir la contención de acceso a este nivel, se divide en 16 bancos (SNUCA), con un tiempo de acceso a un banco de 5 ciclos. Finalmente, la memoria principal DRAM contiene 4GB de almacenamiento y un tiempo de acceso promedio de 250 ciclos.

Red de interconexión: Finalmente, para interconectar todos los cores y los bancos de memoria se utiliza una topología de red en malla 3D 4x4x2. En el nivel superior se sitúan los 16 cores y sus caches L1, mientras que en el nivel inferior se encuentran las caches L2 y los controladores de memoria. El tamaño de los enlaces es de 128 bits, tanto horizontales como verticales y su latencia es de 1 ciclo de procesador. A pesar de que ya se ha demostrado que la latencia de las conexiones verticales mediante TSVs es bastante inferior a la que se produce en enlaces al mismo nivel, se ha optado por una red en donde todos los enlaces poseen la misma latencia.

La Figura 15 muestra la distribución física de los componentes en el chip, junto con la topología utilizada para conectarlos entre sí. La distribución de los distintos componentes del CMP entre las diferentes capas de la arquitectura no ha sido escogida al azar, respondiendo a condicionantes físicos. Como los cores son la mayor fuente de calor producida en un CMP necesitan una mayor proximidad al disipador. Los controladores de memoria, por otra parte, se comunican con la memoria principal situada en la placa base, utilizando los pines de entrada/salida del chip. Por estas dos razones se sitúa en la capa superior a todos los cores, lo más cerca posible del disipador, y en la capa inferior las caches L2 y los cuatro controladores de memoria, lo más próximo posible a los pines de entrada/salida.

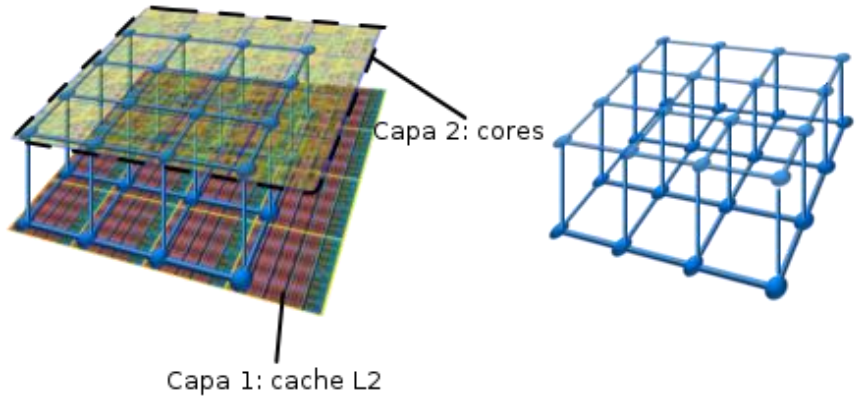

Figura 15 Estructura del CMP propuesto. A la derecha se muestra la topología utilizada para la red de interconexión, una malla de 3 dimensiones 4x4x2.

Partiendo de esta arquitectura, vamos a analizar cuál sería el efecto producido por incorporar tecnologías de memoria diferentes a la SRAM, que nos permitieran incrementar el tamaño de L2 manteniendo el mismo área. Para ello, ejecutaremos el conjunto de aplicaciones utilizado como benchmark y evaluaremos el tiempo que tarda en completarse la ejecución en cada

caso. Incrementaremos de forma progresiva el tamaño de la cache de último nivel, entre 4 y 32 MB. La Figura 16 muestra los resultados obtenidos.

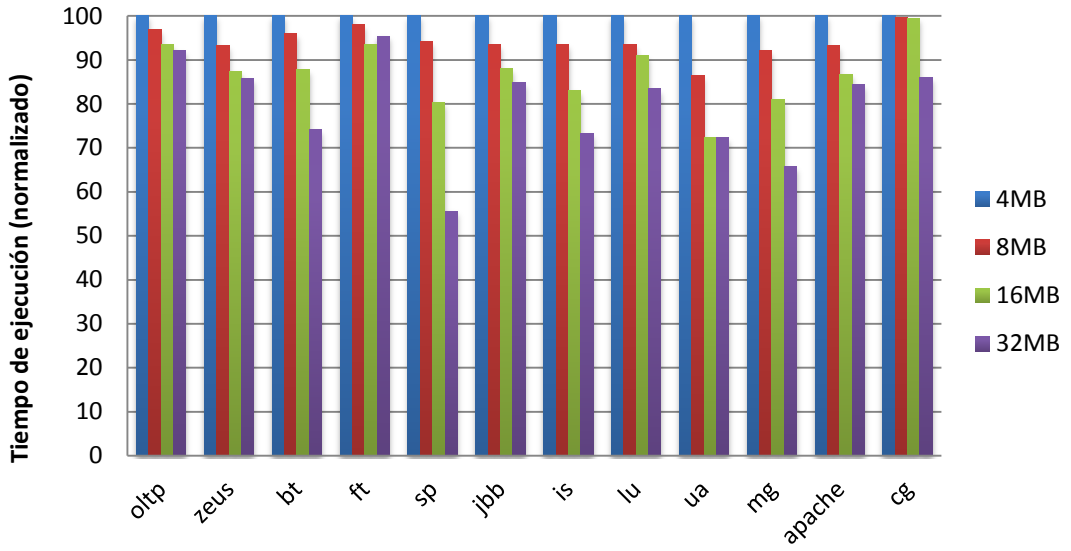

Figura 16 Mejora del tiempo de ejecución con el aumento de la capacidad de la cache L2.

Como se puede observar, cuanto mayor es el tamaño de cache, el tiempo necesario para ejecutar un programa se reduce notablemente. El origen de esta mejora está en que el aumento de la capacidad de la cache de segundo nivel aumenta la probabilidad de que un dato solicitado por el programa se encuentre en dicha cache y no sea necesario propagar la petición a memoria principal, cuyo tiempo de respuesta y velocidad de transmisión son más lentos. Por este motivo, la utilización de tecnologías con mayor densidad de almacenamiento que la tradicional SRAM permite potencialmente reducir el número de misses en este último nivel de cache y, por lo tanto, mejorar el rendimiento.

Estos resultados justifican de forma clara la búsqueda de soluciones para celdas de memoria con mayor densidad de integración. En este trabajo analizaremos el uso de STTRAM para este propósito. Dadas las limitaciones que se analizarán en este capítulo, la propuesta más razonable cosiste en sustituir únicamente en el segundo nivel de cache, dejando el primer nivel con celdas SRAM. Como se ha visto en el capítulo anterior, el tamaño de una celda STT-RAM fija el tiempo que se tardará en escribir un dato en ella (Figura 10 y Figura 11). Adicionalmente, el tamaño de la celda también determina la cantidad de energía necesaria para realizar una escritura, tal y como muestra la Figura 17. Siendo el consumo energético uno de los principales factores de diseño actualmente, la densidad de integración que parece más conveniente desde el punto de vista energético es de un tamaño 4 veces superior al de

SRAM. Por esta razón, a partir de este punto utilizaremos tamaños de L2 cuatro veces superiores cuando comparemos una STT-RAM con una L2 SRAM.

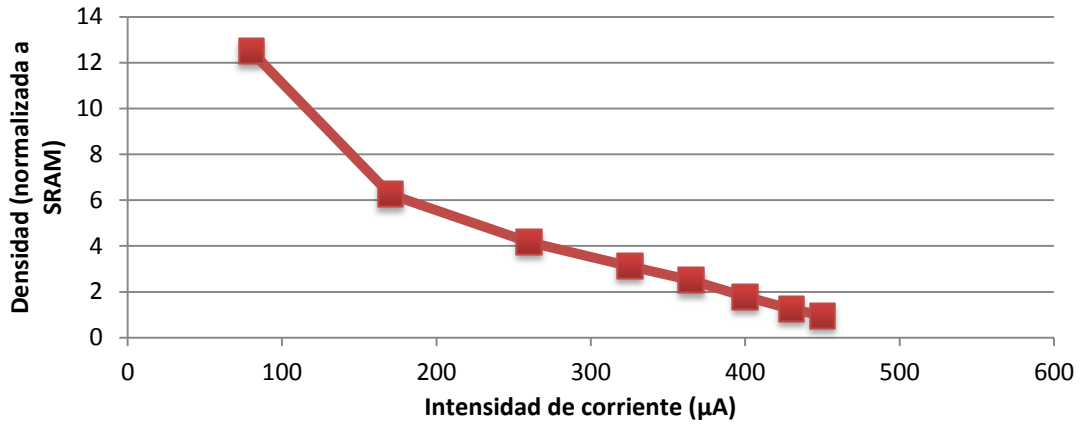

Figura 17 Densidad de integración en función de la intensidad de corriente utilizada.

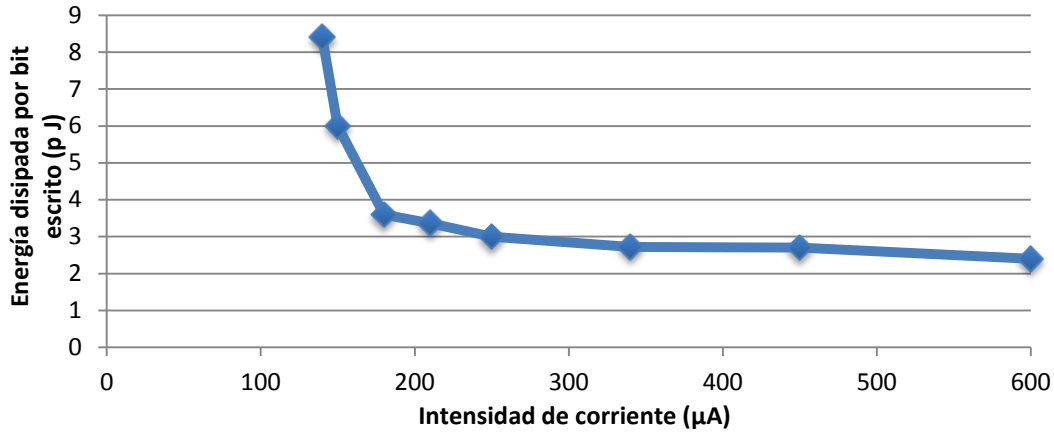

Figura 18 Consumo energético por bit escrito en función de la intensidad de corriente utilizada en la escritura.

3.3 Limitaciones de la tecnología STT-RAM

Las STT-RAM parecen las candidatas más sólidas para reemplazar a las actuales memorias cache SRAM debido a su mayor densidad de integración y a su reducido consumo de energía estática (debida a la corriente de fuga de los transistores). Sin embargo, poseen también otras características no tan favorables como son el consumo de energía dinámica (debida a las transiciones en los transistores) y la latencia de las operaciones de escritura, que son significativamente superiores a la de las celdas SRAM. En este apartado describiremos en más detalle los efectos producidos por estas limitaciones, así como las soluciones ya existentes para ambos casos.

Overhead de energía

La energía disipada por las corrientes de pérdidas en una celda STT-RAM es insignificante en comparación con una celda SRAM. Esto se debe a que una celda de este tipo tan sólo utiliza un transistor (los elementos que más pérdidas producen), a diferencia de los seis que contienen las SRAM. La unión MTJ, al ser un elemento pasivo, no disipa energía mientras no se efectúe ninguna operación. Durante una operación de lectura, la energía consumida es muy similar a la de cualquier otra celda SRAM convencional, dado que la operación conlleva un proceso similar en ambos casos. Sin embargo, la diferencia de consumo es bastante relevante en el caso de una escritura. Durante este proceso, es necesario hacer circular corriente de una intensidad notable por el MTJ para conseguir que la capa libre de éste pueda cambiar de polaridad y, por tanto, cambiar su estado.

La Figura 19 muestra el consumo energético total de un sistema como el descrito en la sección anterior, con dos configuraciones de L2, 8MB de SRAM y 32 de STT-RAM. Como podemos comprobar, el consumo de energía es más significativo en el caso de la STT-RAM, debido al overhead de las escrituras.

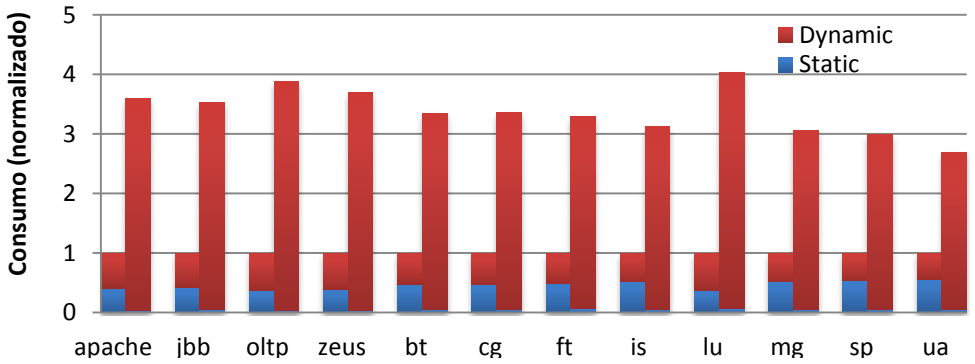

Figura 19 Consumo energético de una cache SRAM y una STT-RAM, normalizado al de SRAM.

Dada su relevancia, existen multitud de propuestas que pretenden reducir el alto consumo energético que caracteriza a las escrituras de una STT-RAM, centradas en diferentes áreas de la jerarquía de memoria.

El trabajo de [45] se centra en reducir el número de escrituras que se producen en el último nivel de cache reduciendo, cuando sea posible, el número de writebacks que se envían desde el nivel superior, es decir, el más cercano al procesador. Los writebacks se producen cuando un bloque de cache modificado ha sido reemplazado y necesita escribirse en el siguiente nivel (STT-RAM en este caso) para mantener los valores actualizados. El algoritmo de reemplazo

propuesto reduce el número de bloques modificados e incita el reemplazo de bloques no modificados y que, por lo tanto, no producirán un writeback.

En [46] y [47] se propone una arquitectura de cache híbrida, es decir, utilizan diferentes tecnologías de almacenamiento en un mismo nivel de cache. Debido al alto coste de las escrituras, dividen la cache en dos regiones diferentes: una para almacenar datos que se leen con frecuencia, hecha a partir de celdas STT-RAM, y otra para almacenar datos que se escriben repetidamente, hecha a partir de celdas SRAM que no tienen sobrecoste en las escrituras. En [48] se propone una solución muy similar. Se restringe el acceso de los cores a la cache L2 híbrida, de forma que no es totalmente compartida sino que cada core únicamente comparte sus bancos de memoria con sus vecinos. Dedican mayor capacidad a bancos de STT-RAM, para lecturas, que a bancos de SRAM. Además, si se produce un acierto con una operación de lectura y el bloque activo se encuentra en la región de escritura (SRAM) se mueve a la de lectura (STTRAM) sin traer otro dato, dado que dedican mayor capacidad a la zona de lectura que a la de escritura.

Otros trabajos se centran en soluciones enfocadas a la tecnología en vez de a la arquitectura. Así, en [49] se observó que la reducción del tamaño de la capa libre del MTJ produce una disminución en el tiempo de retención del dato almacenado, es decir, se reduce el tiempo que un dato permanece almacenado en la celda. En un principio, las caches STTRAM poseen un tiempo de retención de aproximadamente diez años, algo exageradamente alto en la cache de un microprocesador, donde los datos cambian con mucha frecuencia. Esta reducción del tamaño de la capa libre rebaja el tiempo y energía empleados en polarizarla, permitiendo operaciones de escritura más rápidas con menor coste energético. Sin embargo, dado el ajustado tiempo de vida de un dato, se torna necesario implementar un mecanismo que refresque los datos almacenados para que no se pierdan, similar al que se utiliza en DRAM.

El uso de STT-RAM en L1 y L2 con diferentes niveles de retención también se propone en [50]. Debido a que los datos en L1 cambian con más frecuencia, se pueden utilizar celdas con tiempos de retención menor, reduciendo la duración y energía empleada en las escrituras. En caches de nivel más bajo (L2) se utilizan niveles de retención mayores combinados con un sistema que refresca los valores almacenados periódicamente.

Finalmente, en [51] proponen un sistema que reduce el tiempo y consumo energético de las escrituras. Una gran proporción de los bits escritos en las memorias STT-RAM son

redundantes, es decir, el valor que se escribe en una celda es el mismo que el que ya estaba almacenado. Por este motivo, diseñan un sistema que sólo escribe en las celdas en donde el valor nuevo y el viejo no concuerdan sin realizar una lectura adicional antes de cada escritura. Gracias a que la intensidad de la corriente apenas varía en las primeras etapas de una escritura en STT-RAM, es posible leer esta intensidad de corriente, que corresponde al valor antiguo, simultáneamente e interrumpir la escritura si el valor que se escribe y el existente coinciden.

Pérdida de rendimiento por escrituras lentas

Una celda MTJ necesita una intensidad de corriente determinada durante un cierto intervalo temporal para que el cambio de polaridad se haga efectivo. A pesar de que se pueden agilizar las escrituras si se incrementa la intensidad de corriente que circula a través del MTJ, esta solución forzaría a incluir en cada celda un transistor NMOS de tamaño muy elevado, reduciendo el nivel de integración a niveles de SRAM.

Para comprender cómo puede afectar el tiempo de escritura a nuestro sistema es necesario describir algunos aspectos básicos del protocolo de coherencia empleado. El protocolo de coherencia es el encargado de gestionar los datos compartidos entre todos los procesadores, de tal forma que cada procesador tenga el valor correcto de un dato en todo momento. Nuestro protocolo, de tipo broadcast, envía una solicitud al resto de procesadores y a la cache de segundo nivel cada vez que necesita un dato, respondiendo aquellos que tengan una copia en ese momento. En este protocolo, un bloque se escribe en la cache L2 por un motivo, cuando en una cache de primer nivel un nuevo dato no tiene espacio y necesita expulsar uno antiguo, se produce un reemplazo (o writeback) y el dato antiguo es expulsado a la cache L2. El servicio de estas escrituras no está dentro del camino crítico, ya que un procesador no necesita esperar a que el dato expulsado acabe de escribirse en L2, pero puede entorpecer otras peticiones más importantes si mantiene el banco ocupado durante un tiempo relativamente alto. Por ejemplo, si expulsamos un dato a un banco determinado e inmediatamente después necesitamos leer un dato que se encuentra en dicho banco, la lectura no podrá tener lugar hasta que finalice la escritura. Esta inversión de prioridad puede provocar que un programa se pare por completo, aumentando el tiempo necesario para ejecutarse.

Un banco de L2 con tecnología SRAM presenta un tiempo de escritura relativamente rápido, similar al de lectura (aproximadamente 5 ciclos en un banco de 1Mbyte operando a 2GHz). Por el contrario, la tecnología STT-RAM puede presentar tiempos de escritura hasta 10 veces

superiores. La Figura 20 muestra el efecto que puede tener sobre el rendimiento la latencia de escritura en el último nivel de cache. Para ello tomaremos como base un sistema con tecnología SRAM en la L2 de 8MB (512KBytes por banco) y 5 ciclos de lectura y escritura en un banco, y lo compararemos con una L2 con capacidad 4 veces superior (32MB) pero con un tiempo de lectura creciente, desde 5 a 45 ciclos.

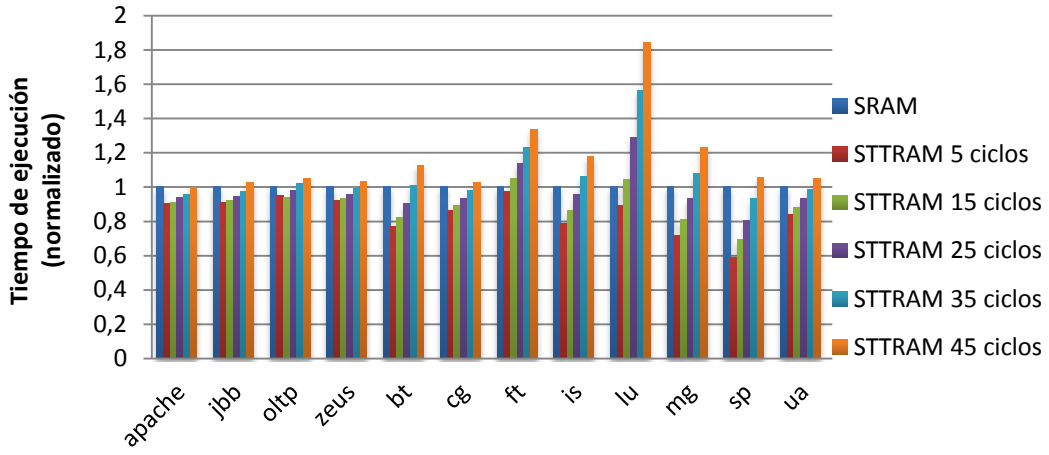

Figura 20 Efecto del aumento del tiempo de escritura en una cache STT-RAM.

De acuerdo con los resultados en [52], una STTRAM con un nivel de integración 4 veces superior presenta unos tiempos de escritura de aproximadamente 35 ciclos. Por tanto, a la vista de la Figura 20, observamos que en este caso el aumento en capacidad de L2 no es capaz de compensar este problema, y el sistema con SRAM presenta mejores resultados en algunos casos. Por lo tanto, es de vital importancia servir las peticiones de lectura antes que otras de menor prioridad, que no afectan al rendimiento en tanta medida, para reducir el impacto negativo que éstas causan en el rendimiento.

La degradación de rendimiento tiene dos causas fundamentales. Primero, si una petición de memoria llega a su destino cuando el banco al que desea acceder está ocupado, ésta ha de esperar a que el banco finalice su trabajo antes de ser procesada. Esta espera aumenta la latencia promedio de acceso al banco de L2. Además, en caso de que estas peticiones se acumulen, se irán distribuyendo por la red, entorpeciendo a otros mensajes que ni siquiera comparten el mismo destino. De esta forma, un mensaje que se dirige a un banco determinado puede ser entorpecido por otro que esté esperando a que un banco completamente diferente se haya liberado. Para comprobar el peso de cada causa hemos utilizado los resultados de la figura anterior, midiendo en este caso la latencia de los mensajes a L2. Descompondremos esa latencia en tres componentes diferentes:

- Inyección: Determina el tiempo que debe esperar un paquete antes de ser inyectado en la red de interconexión. El mensaje ya ha sido generado por un controlador, pero debe esperar a que se inyecten todos los que fueron generados antes.
- Red: tiempo de tránsito, empleado en avanzar desde el router de origen al router de destino.
- Consumo: Una vez que el paquete alcanza su destino, tiempo que tarda en completarse la lectura o la escritura en el banco de cache al que iba dirigido el dato.

Estos resultados son mostrados en la Figura 21. Como se puede observar, cuando cambiamos de un sistema con 5 ciclos de escritura a uno con 35, la latencia de las peticiones a L2 crece, y la fracción de cada componente de latencia cambia. El tiempo de inyección se mantiene constante en ambos casos. Por el contrario, el tiempo de consumo crece sustancialmente debido a las escrituras lentas. El tiempo de espera en el consumidor parece ser el principal responsable de la pérdida de rendimiento, por lo que las propuestas de este proyecto van encaminadas a paliar este problema.

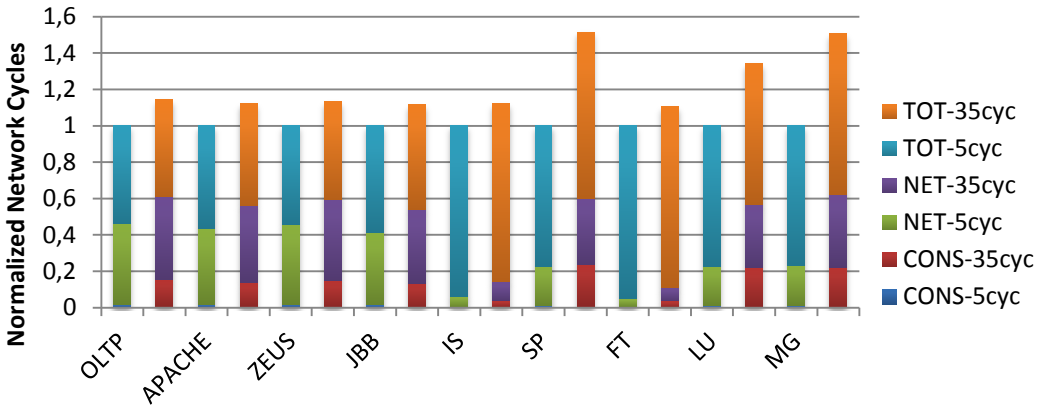

Figura 21 Comparación de la latencia de peticiones en un sistema con SRAM y otro con STT-RAM. Latencia de Red (NET), latencia de consumo (CONS) y latencia de inyección (TOT).

A pesar de la gran cantidad de propuestas existentes para paliar el problema energético de las STT-RAM, el número de soluciones en el campo de la latencia de escritura son escasas. En el siguiente capítulo analizaremos un trabajo existente al respecto y realizaremos nuestra propia propuesta.

Capítulo 4. Mecanismos para esconder la latencia de las escrituras

Como hemos descrito en el capítulo anterior, el tiempo necesario para atender una petición de escritura en un banco STT-RAM es demasiado grande como para que una implementación sin mecanismos adicionales mejore de forma significativa los resultados de bloques de cache construidos con tecnologías más tradicionales. En el capítulo anterior hemos introducido algunas propuestas que tratan de corregir el problema del consumo energético. En este capítulo nos centraremos en el análisis del problema restante que afecta a las memorias construidas con celdas STT-RAM. Comenzaremos nuestra evaluación describiendo y estudiando una propuesta ya existente que trata de reducir la penalización en rendimiento de las escrituras con ayuda de los componentes de la red de interconexión. En el análisis de dicho trabajo realizado durante el desarrollo del proyecto nos hemos dado cuenta de que parte de las decisiones tomadas en la configuración original se han tomado con criterios equivocados, por lo que decidimos repetir la evaluación de las propuestas con una configuración más adecuada. En este capítulo mostraremos tanto los resultados originales del trabajo con los obtenidos en el proyecto.

Tras este análisis preliminar, realizaremos varias propuestas alternativas en la que intentaremos buscar soluciones al problema de la latencia de escritura. De la misma forma que la descrita en el párrafo anterior, nuestra primera propuesta intentará utilizar componentes de la red de interconexión para mejorar el rendimiento del sistema. Por el contrario, en nuestra segunda propuesta intentaremos dotar a los bancos de la cache de último nivel de mecanismos adicionales que permitan priorizar de alguna forma las lecturas sobre las escrituras.

4.1 Una propuesta de mejora: ISCA 2011

En el trabajo realizado en [52], presentado en una de las conferencias más relevantes del área de arquitectura de computadores, se propone una modificación en la red de interconexión que permite reducir el impacto que producen en el rendimiento las escrituras en una cache L2 STT-RAM. La solución se centra en tratar de minimizar la congestión en la red de interconexión originada por la acumulación de mensajes que se produce cuando un banco está demasiado tiempo ocupado, es decir, cuando se está escribiendo un dato en uno de sus bloques.

Descripción

Para evitar la acumulación de paquetes, el trabajo propone incorporar a cada router un mecanismo de arbitraje (algoritmo que controla la asignación de puertos de salida entre los mensajes localizados en los buffers de entrada) que prioriza el avance de aquellos paquetes que se dirigen a bancos de cache en los que en ese momento no se está realizando una escritura, frente a aquellos que se encuentran ocupados, siendo los causantes de la congestión en la red.

El problema fundamental al que se enfrenta esta solución es la dificultad de distribuir la información sobre el estado de cada banco entre todos los nodos de la red. Dado que mantener esta información requeriría un overhead de comunicaciones importante, se opta por implementar un encaminamiento especial que evite tener que realizar esta tarea. Todos los mensajes destinados a un determinado banco de L2 deberán pasar por un mismo router, que se utilizará como punto de serialización y se denomina router "padre". Cada router padre posee múltiples routers hijo, o lo que es lo mismo, múltiples bancos de L2 comparten el mismo punto de serialización.

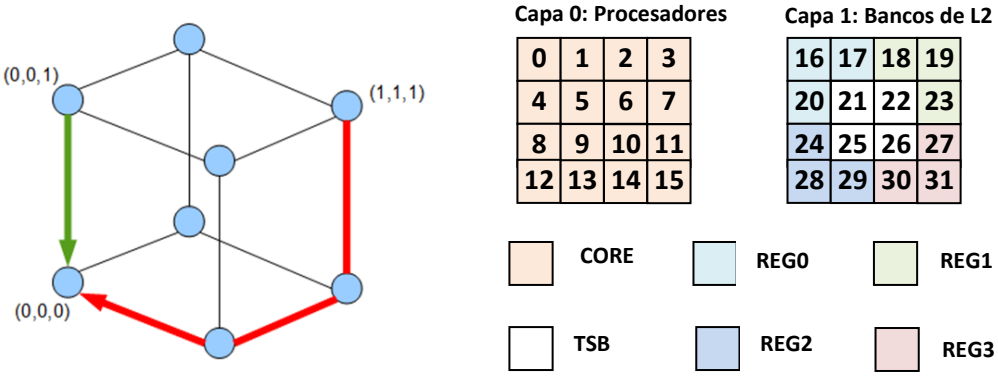

Figura 22 A la izquierda, descripción gráfica de un enrutamiento ZXY. A la derecha, estructura del sistema utilizado para las pruebas.

Con las políticas de encaminamiento comunes en este tipo de entornos existirán mensajes con diferente origen y mismo destino que no compartan ningún router intermedio. La Figura 22 muestra un ejemplo de dicha situación cuando se implementa una política de encaminamiento en orden de dimensión (Z-X-Y). Un mensaje generado en el core (0,0,1) que se dirige al banco situado en (0,0,0) simplemente tiene que atravesar un enlace vertical para llegar a su destino, mientras que en uno generado en (1,1,1) el mensaje será primero enviado a la capa inferior y luego avanzará en dirección X e Y para llegar a su destino. Estas dos rutas no tienen nodos en común, por lo tanto se necesita una nueva política de encaminamiento

que permita canalizar todos los paquetes dirigidos a un mismo destino por un punto intermedio compartido.

El trabajo en [52] escoge una solución sencilla a este problema. Divide la capa en la que se encuentran los bancos de L2 en regiones, y solamente se permite la utilización de un enlace vertical para acceder a cada región, que se denominará TSB. Para ilustrar este mecanismo utilizaremos un sistema de ejemplo formado por 16 procesadores en una capa y 12 bancos de L2 en la superior, todo ello conectado mediante una red en malla de dimensiones 4x4x2. La Figura 22 muestra un esquema del sistema descrito. En este caso la capa superior (cache L2) se divide en cuatro regiones, y a cada una de estas se le asigna un enlace vertical correspondiente al router en el centro de la malla y dentro de esa región. Por ejemplo, el router 22 es el padre de los routers 18, 19 y 23. Por tanto, un mensaje desde cualquier procesador que tenga como destino estos nodos debe ser enrutado por el router 22. En resumen, todas las peticiones dirigidas a un banco de memoria en particular realizan el mismo procedimiento para llegar a su destino:

1. Avanzan hasta un nodo en particular que pertenezca a la capa de los cores siguiendo un enrutamiento X-Y.
2. Realizan un movimiento en vertical, hacia un router de la región de memoria (el nodo intermedio).
3. Se dirigen a su destino utilizando de nuevo el enrutamiento X-Y, pasando por el nodo padre.

En la organización como la mostrada en Figura 22 solamente aquellos mensajes dirigidos a los bancos de L2 pueden sufrir mayores tiempos de espera debido a las escrituras lentas, por lo que esta política de encaminamiento solamente se aplicará a este tipo de mensajes. De esta forma, cuando un procesador pida un dato a un banco de L2, esta petición sí que debe ser encaminada por un TSB. Sin embargo, cuando el banco de L2 mande el dato al procesador, este mensaje podrá utilizar cualquier enlace vertical, limitado solo por el algoritmo de encaminamiento. Una vez la red está estructurada, conocer si un banco determinado está ocupado es una tarea sencilla. El nodo padre apunta el momento en que una petición de escritura se dirige hacia un banco que le corresponde. En ese instante, interrumpe el resto de los mensajes que compartan ese mismo destino, hasta que estime que dicho banco de memoria se haya liberado. El tráfico no se interrumpe con las lecturas debido a que la latencia de las mismas no es tan relevante como para parar el resto del tráfico. Una vez que los mensajes a un destino son bloqueados, para evitar que éstos impidan el paso a otros que sí podrían seguir transmitiéndose se utiliza lo que se conoce como canales virtuales. Los

canales virtuales consisten en una réplica de los buffers en cada puerto de entrada de los routers. De esta manera, los paquetes de un canal virtual que están parados no molestan a otros paquetes situados en canales virtuales diferentes. En este caso, se utilizan para que las peticiones dirigidas a nodos libres sean capaces de evitar a otras que han sido bloqueadas en su nodo padre. Para ello, es necesario asignar un canal virtual a cada petición en cada salto, es decir, cada vez que un paquete llega un router se asigna un nuevo canal virtual, que puede ser o no igual al que se asignó previamente. De esta forma, si hay demasiados paquetes atascados en un canal virtual, esta petición puede evitarlos si el router le asigna un canal virtual diferente, como se puede observar en el ejemplo de la Figura 23.

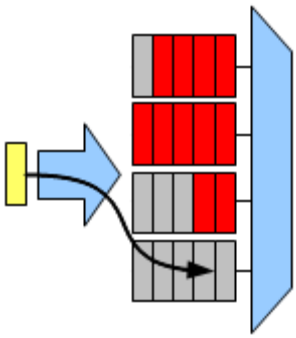

Figura 23 Utilización de Canales Virtuales para evitar bloqueos.

Una vez descrito el mecanismo de parada, falta por describir cuánto tiempo se mantiene ese mensaje bloqueado antes de reanudar su marcha. El tiempo que un nodo padre bloquea la transmisión de tráfico a un banco afecta al rendimiento del sistema. Una espera demasiado corta produce que la congestión en el nodo destino, aunque se reduce con respecto a la situación base, no desaparece por completo. Por el contrario, una espera demasiado larga produce que los bancos de memoria permanezcan libres esperando la llegada de nuevas peticiones que ya se enviaron pero están bloqueadas.

Podemos descomponer dicho tiempo de espera en tres partes; **Latencia de envío** (el tiempo que tarda un paquete en enviarse desde el nodo padre al nodo hijo), **Latencia estimada por congestión** (tiempo estimado que se perderá en el envío debido a la congestión producida por el tráfico de la red) y **Tiempo de servicio** (tiempo que tarda el banco en procesar la petición, es decir, el tiempo que tarda el banco en escribir). En el trabajo en [52] se sigue la política de retener los mensajes tras una escritura un número de ciclos igual al tiempo de espera. Atendiendo a la forma de calcular dicho tiempo de espera, se evalúan diferentes algoritmos para calcular el número de ciclos que se bloquea a las peticiones consecutivas:

Simplistic-scheme (SS): en esta implementación no se considera la latencia debida a la congestión, y se asume que tiene valor cero, por lo que solamente se bloquean los mensajes durante el tiempo de envío más el tiempo de servicio. En este caso no se necesitan lógica ni cableado adicionales y por lo tanto el consumo energético es menor.

Regional Congestion Aware (RCA) scheme: en esta implementación se intenta estimar un valor para la latencia de congestión. Los routers calculan la ocupación de sus interfaces de entrada y se lo comunican a sus vecinos. La estimación de la congestión se calcula ponderando las ocupaciones del propio router con las de sus vecinos.

Window-based (WB): en este caso, se utiliza una técnica más precisa para determinar el tiempo de congestión. Cada cierto intervalo de tiempo, se añade un contador de tiempo al contenido de una petición de lectura. El router destino, una vez llegado el mensaje, genera un mensaje de vuelta conteniendo el mismo contador. Dicho contador va incrementando su valor cada ciclo de reloj, tanto en la ida como en la vuelta. Por lo tanto, el tiempo que ha tardado el mensaje en llegar a su destino es igual a la mitad del valor del contador. Para realizar todas estas tareas, los routers necesitan de lógica adicional.

Con cada una de estas tres configuraciones, en [52] se lleva a cabo una evaluación de las propuestas sobre un sistema con una organización similar a la descrita en el Capítulo 3 de este proyecto, simplemente modificando el número de procesadores y bancos. En este caso las simulaciones se llevan a cabo para un sistema con dos capas, en el que la capa inferior contiene 64 procesadores con una cache de primer nivel privada de 32Kbytes, mientras que la segunda capa contiene los 64 bancos de L2 compartida, de 1Mbyte cada uno. Todos estos componentes se encuentran unidos por una red en malla de tamaño 8x8x2. La Figura 24 muestra los resultados obtenidos para un conjunto amplio de aplicaciones, de naturaleza similar a las que nosotros empleamos. En esta figura se comparan las siguientes configuraciones:

- SRAM-64TSB: Caso base en el que la cache L2 está construida con tecnología SRAM, con 1Mbyte por banco. La política de encaminamiento por la red será del tipo convencional en orden de dimensión (X-Y-Z).
- MRAM-64TSB: En este caso se cambia la tecnología de L2 por celdas STT-RAM, por lo que cada banco pasa a tener 4MBytes. La política de encaminamiento se mantiene.
- MRAM-4TSB-**: Finalmente, esta configuración combina tecnología STT-RAM con la solución descrita en esta sección para evitar bloqueos por escrituras lentas. Se presentan resultados para cada una de las formas de calcular el tiempo de bloqueo.

Los resultados presentan valores de IPC (Instrucciones por Ciclo) normalizados, por lo que valores mayores indican mejor rendimiento. A la vista de los resultados, podemos observar que el paso de SRAM a STT-RAM (aquí se denomina MRAM) produce resultados dispares

según la aplicación. En aquellas en las que se producen pocas escrituras en L2 el aumento de capacidad hace que obtengan mejores resultados, como bscls ó bdtrk. Sin embargo, cuando el número de escrituras es elevado, MRAM-64TSB no es capaz de obtener mejores resultados que SRAM-64TSB.

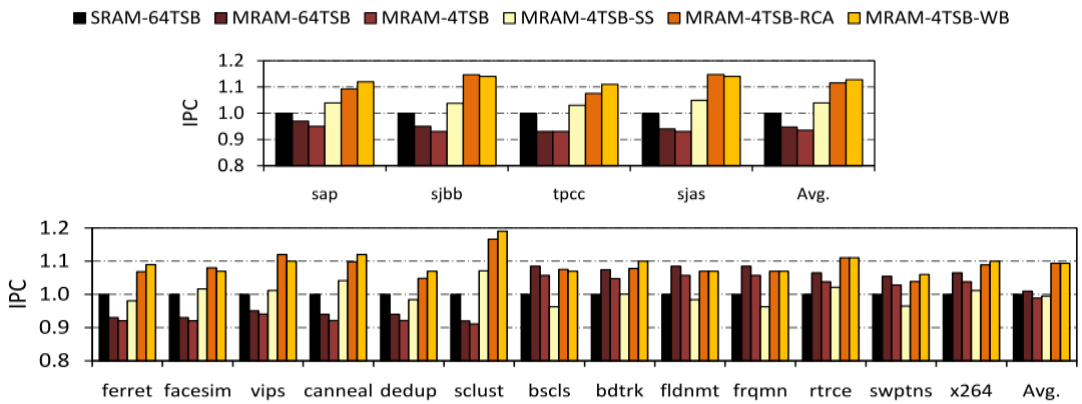

Figura 24 Comparación de rendimiento de un sistema base con SRAM y MRAM y el sistema propuesto con MRAM.

Evaluación

La primera tarea llevada a cabo en este proyecto fue la implementación de todos los mecanismos descritos en la sección previa en las herramientas de simulación del grupo de Arquitectura de Computadores, más concretamente en su simulador de redes de interconexión TOPAZ [53]. El objetivo ha sido intentar replicar los experimentos con el simulador de sistema completo del grupo. Dada la precisión de nuestras herramientas, el tiempo requerido para realizar las simulaciones es extremadamente elevado y muy sensible al número de procesadores. Por esta razón, el sistema simulado en nuestro caso será más reducido, con solo 16 procesadores.

Tras un análisis más detallado del contenido del trabajo nos dimos cuenta de que existían una serie de decisiones de diseño que podían cuestionar la validez de los resultados obtenidos. Así mismo, se detectaron fallos de concepto importantes en los mecanismos para calcular los tiempos de espera en el router padre. Antes de presentar nuestra evaluación describimos con más detalle los problemas encontrados a los que hacemos referencia.

El diseño de la red es un factor muy importante a tener en cuenta para que el sistema funcione correctamente. En el trabajo citado, se hace uso de una red con características que se alejan de las redes de interconexión convencionales. Por ejemplo, el tamaño de los buffers de entrada de un router no son lo suficientemente grandes como para poder almacenar un

paquete por completo. Esto obliga a que cada vez que un paquete detiene su avance quede distribuirlo por varios routers de la red, lo que aumenta los niveles de congestión. Si además esta parada se produce en un router padre, la situación se agrava dado que el número de ciclos de parada puede ser elevado. Los routers, además, son capaces de procesar, arbitrar y enviar un flit por cada ciclo de procesador, algo que resulta difícil trabajando a frecuencias de procesador habituales en este tipo de entorno. Con un tiempo de paso por router de 1 ciclo, el overhead producido por desviar las peticiones hacia los TSB puede ser mínimo, pero cuando ese tiempo de paso crece a valores más habituales de varios ciclos, la latencia de un mensaje se puede disparar. Además del aumento de latencia, concentrar un gran volúmen de tráfico en un número reducido de nodos (cuatro, en nuestro caso) reduce drásticamente el ancho de banda que puede aportar la red, debido a que unos pocos canales han de soportar mucho más tráfico que el resto. De nuevo, esta decisión vuelve a aumentar los niveles de congestión en ciertos sectores de la red. Finalmente, en este artículo se considera que los mensajes esperan en un buffer de entrada del router antes de ser escritos en el banco de L2. Este planteamiento parece poco realista dada la existencia de interfaces de red entre el router y el banco en este tipo de sistemas. La omisión de un buffer entre el router y el controlador de cache provoca que los paquetes se mantengan en los puertos de entrada del router a la espera de que el controlador esté nuevamente disponible, entorpeciendo al resto del tráfico que simplemente atraviesa ese router. Otra decisión que aumenta de forma "deliberada" el nivel de congestión de la red.

La asignación de múltiples canales virtuales a un mismo tipo de mensaje, utilizada para evitar las peticiones detenidas, puede producir un interbloqueo, dejando inservible parte o la totalidad de la red. Este interbloqueo se conoce como *message-dependent deadlock* o *end-to-end deadlock* [54] y únicamente se manifiesta en las redes donde la inyección de un mensaje esté sometida a la recepción de un segundo mensaje, como la que se muestra en la Figura 25. En nuestro sistema, las peticiones de lectura en memoria generan un mensaje de respuesta que contiene el dato solicitado, por lo que es vulnerable a este tipo de bloqueo.

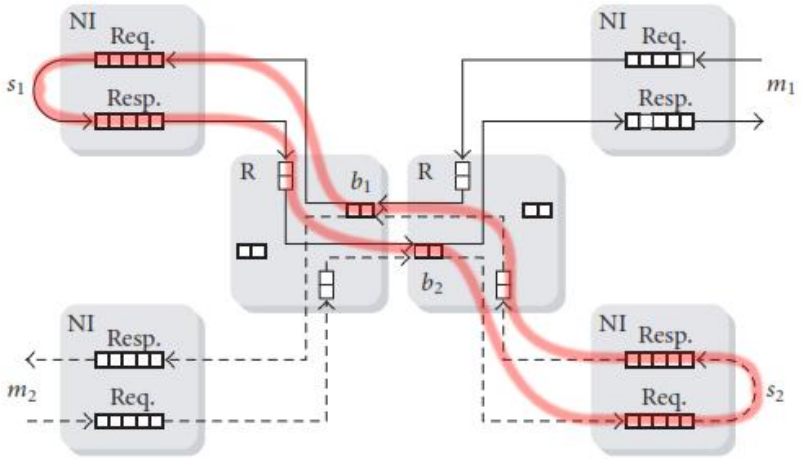

Figura 25 Interbloqueo por dependencia de mensajes. La dependencia cíclica, resaltada en rojo, se produce entre los elementos esclavos, los controladores de cache.

Por otra parte, hemos podido observar que el tiempo de bloqueo de los mensajes en un nodo padre se calcula como la suma de tres componentes: el tiempo de espera, de envío y de congestión. Sin embargo, este cálculo es erróneo y provoca que, tras una escritura, el banco quede libre durante un instante, aun existiendo otras peticiones pendientes. Pongamos por ejemplo que el tiempo de envío es de 5 ciclos, el tiempo de servicio de una escritura es de 35 y, para simplificar, que no existe congestión. Supongamos que una petición de escritura sobrepasa el nodo padre correspondiente a su destino en t=0, lo que significa que éste destino está libre. Esta primera petición comenzará a procesarse según llegue a su destino, es decir, en $t=0+T_{envío}= 5$. Si una segunda petición llega al nodo padre, éste bloqueará su paso hasta que estime que el banco al que quiere dirigirse haya sido liberado. Según lo propuesto por [52], la petición podrá avanzar una vez hayan transcurrido $T_{envio}+T_{servicio} = 40$ ciclos de que la primera petición ha avanzado. Por lo tanto, el segundo mensaje abandonará el nodo padre en t=40 y llegará a su destino en t=45. Sin embargo, el banco al que se dirige terminó de procesar la petición en $t=5+T_{servicio} = 40$, por lo que ha estado libre durante 5 ciclos esperando la llegada de la siguiente petición. Puede observarse una representación gráfica de este ejemplo en la Figura 26. Este error ha sido corregido en nuestra simulación, quitando de la ecuación la latencia de envío entre el nodo padre y el destino y, por lo tanto, minimizando el tiempo que el controlador permanece libre.

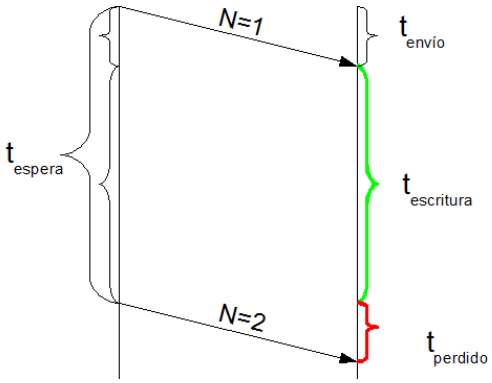

Figura 26 El tiempo de espera propuesto provoca que los bancos permanezcan parados un leve intervalo de tiempo entre peticiones.

Por último, la medición del rendimiento del sistema se hace mediante el IPC (número de instrucciones por ciclo), en lugar de hacer uso de la duración de los programas ejecutados. El IPC toma en cuenta cualquier instrucción que se haya ejecutado, tanto si es una instrucción útil como si no lo es. Por ejemplo, en una aplicación multi-thread, cuando un hilo ha sido suspendido debido a una barrera de sincronización, el procesador comienza a ejecutar lo que se conoce como *idle loop* (bucle de inactividad). Este *idle loop* consta de un número muy reducido de instrucciones que no producen ningún tipo de bloqueo (fallos de cache, etc.), por lo se ejecutan con mucha frecuencia (IPC elevado). Por lo tanto, si algún procesador ejecuta un *idle loop* durante la ejecución de una simulación incrementará en gran medida su valor de IPC, lo que contamina el análisis de rendimiento que se está llevando a cabo.

Todos los motivos aquí descritos parecen anular la validez de los resultados presentados en [52]. La red parece haber sido diseñada para obtener el rendimiento más bajo posible y aumentar de forma artificial los niveles de congestión, mientras que algunos parámetros presentan valores poco realistas a favor de las propuestas realizadas. Aspectos esenciales para que el sistema funcione de forma correcta no se han tenido en cuenta, y finalmente los algoritmos de cálculo de tiempos de bloqueo parecen ser erroneos. Por estas razones, en vez de intentar reproducir sus resultados, consideramos de mayor interés evaluar sus propuestas en un sistema diseñado de manera mucho más razonable. Para ello, en nuestra configuración el tamaño de los buffers de los routers se incrementa a 10 flits, que permiten almacenar más de un paquete completo. Se asignan canales virtuales a cada tipo de mensaje, quedando las peticiones de lectura en un canal virtual diferente a las respuestas, para evitar un interbloqueo [54]. Finalmente, se aumenta el número de etapas necesarias para que un router procese un flit, desde su recepción hasta su envío, aumentando su latencia de 1 ciclo a 3. Con estos cambios y la configuración de sistema del Capítulo 3, simularemos nuestro set de

aplicaciones. Repetiremos las configuraciones SRAM-64TSB, MRAM-4TSB, MRAM-4TSB-SS y MRAM-4TSB-WB.

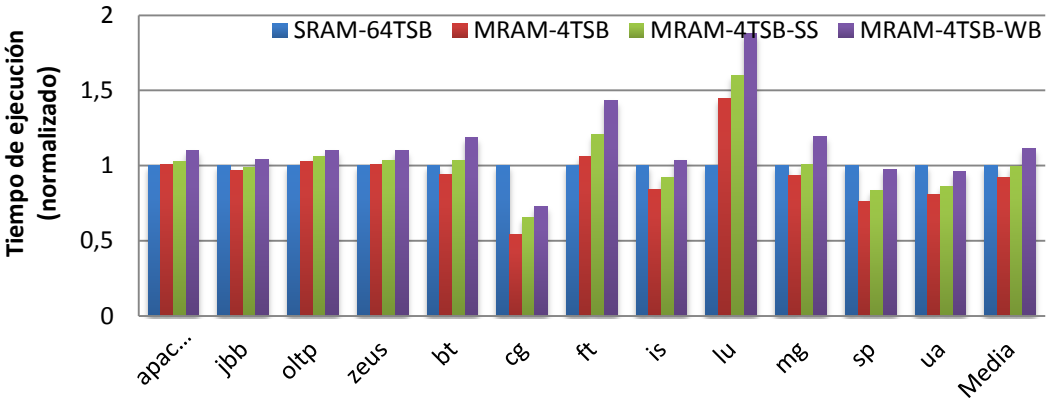

Figura 27 Comparación del sistema base con SRAM y la propuesta evaluada.

Los resultados de nuestra simulación se muestran en Figura 27. En este caso mostramos tiempos de ejecución en vez de IPC, por lo que valores menores implican mejor rendimiento. Como se puede observar, nuestros resultados son totalmente contrarios a los presentados en [52] y mostrados en . En nuestro caso, tanto SRAM-64TSB como MRAM-64TSB obtienen mejores resultados que cualquiera de las propuestas de encaminamiento especial y bloqueo en router padre. Ni SS ni WB son capaces de obtener mejora alguna en ninguna de las aplicaciones analizadas. La explicación de este comportamiento es muy sencilla. Cuando se mejora el diseño de la red los niveles de congestión se relajan, y la latencia de los mensajes depende más de la distancia entre origen y destino que del nivel de congestión. En este caso, la indirección de SS y WB por los cuatro pilares centrales hace que su latencia se incremente de forma notable, afectando seriamente al rendimiento. Este resultado invalida estas propuestas bajo configuraciones de sistema más razonables que las empleadas en trabajo original.

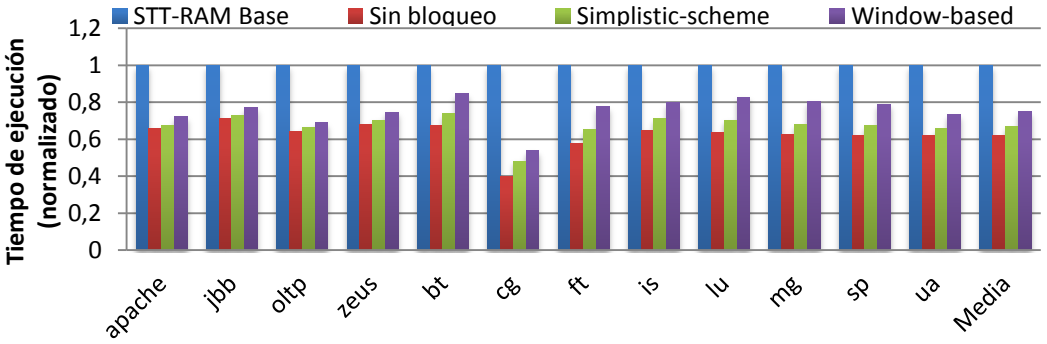

Figura 28 Comparación entre una red de interconexión bien diseñada (sin mejoras) y la propuesta evaluada.

La Interfaz de Red (NIC)

Como hemos visto, el sistema evaluado supone erróneamente que las peticiones deben esperar en un buffer de entrada del router antes de ser atendidas por el controlador de cache. En realidad esto no es así, y este tipo de sistemas presentan un componente intermedio que se encarga de gestionar la comunicación entre el banco de cache o el procesador y la red de interconexión. Este componente se conoce como Interfaz de red o Network Interface Controller (NIC). Cada NIC posee diversas colas de almacenamiento en las que los mensajes esperan a ser atendidos por el controlador de cache, como muestra el esquema de Figura 29.

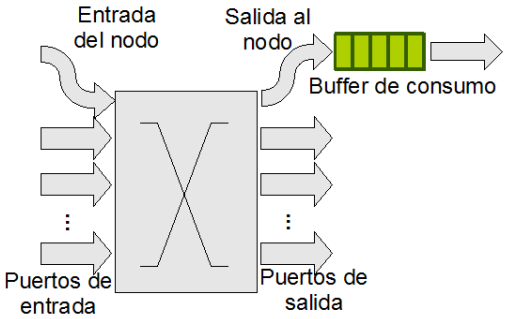

Figura 29 Buffer de consumo introducido en NIC de cada router de la capa de memoria

Un primer análisis que nos pareció interesante fue determinar cuál es el efecto del tamaño de almacenamiento en el NIC sobre el rendimiento del sistema. Para ello realizamos la simulación de nuestro set de aplicaciones para un tamaño creciente del espacio de almacenamiento en consumo. La Figura 30 muestra los resultados del experimento. Los resultados se han normalizado a los obtenidos por un sistema sin NIC. Se han valorado tamaños proporcionales al tamaño de paquete, como 10, 15 ó 20 flits. Finalmente también se ha evaluado el caso de espacio de almacenamiento prácticamente ilimitado.

Como puede observarse, a medida que el tamaño de buffer aumenta, el tiempo de ejecución de las aplicaciones se reduce. En algunos casos, como apache o jbb, un tamaño de 10 flits es suficiente, y a mayores tamaños ya no se observan mejoras. En el resto de casos el rendimiento sigue mejorando tras 10 flits, pero el retorno en rendimiento es mucho menor que en el salto de 0 a 10 flits. Dado que aumentar el tamaño de estos buffers conlleva un aumento del área del NIC y de su consumo de potencia, parece razonable limitar el incremento de tamaño hasta valores que muestren mejoras de rendimiento significativas. Por esta razón, para el resto del proyecto utilizaremos un tamaño de buffer de 10 flits en el interfaz de red.

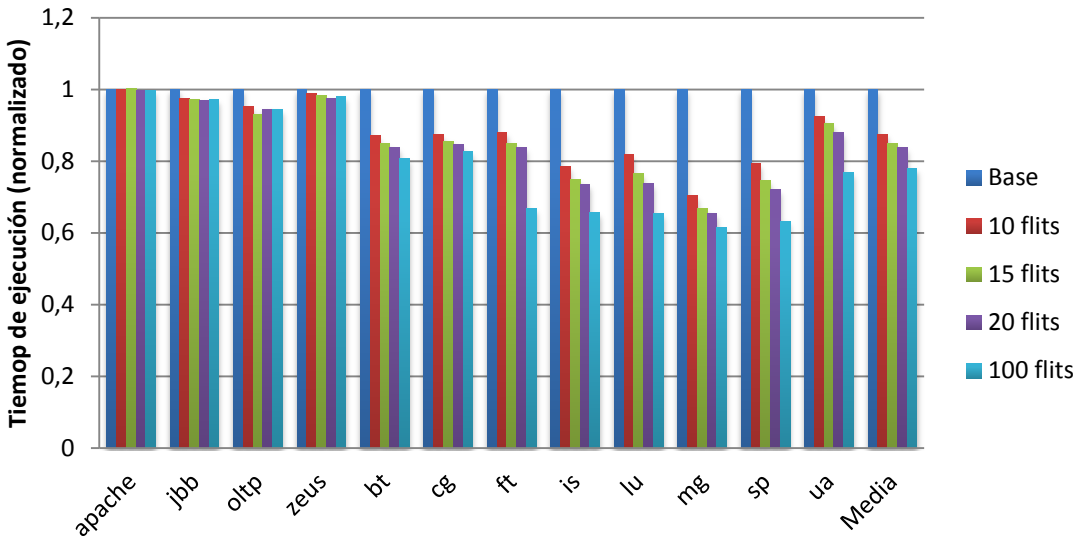

Figura 30 Efecto del tamaño del buffer del NIC en el rendimiento del sistema.

4.2 Nuestra Propuesta

En el apartado previo hemos podido observar que el tamaño del buffer en el NIC influye en el rendimiento del sistema. Sin embargo, el comportamiento de un buffer convencional no permite priorizar mensajes críticos en detrimento de otros menos importantes. Por lo tanto, proponemos en esta sección una modificación que permita dar la mayor prioridad posible a las peticiones de lectura sobre las de escritura.

En el sistema utilizado, únicamente se producen escrituras en cache L2 cuando se produce un writeback, es decir, la cache de primer nivel expulsa un dato modificado para sustituirlo por otro dato necesario para continuar la ejecución. El algoritmo de expulsión de los datos de la cache de primer nivel es de tipo LRU (Least Recently Used). Cuando un dato es reemplazado, se busca un bloque al que hace mucho tiempo que no se accede, dado que la probabilidad de que se vuelva a acceder a él es alta. Cabe destacar que, en nuestro este sistema, los datos procedentes de la memoria principal, por los cuales el procesador está esperando, se escriben directamente en L1 y no en L2. En consecuencia, las únicas peticiones de escritura que se dirigirán a un banco de cache L2 están originadas por un writeback, es decir, por la escritura de datos que no son necesarios para el procesador, y cuya probabilidad de lectura en un tiempo corto es especialmente baja. Por tanto, incrementar el tiempo que tarda ese dato en ser escrito en la cache de nivel 2 no va a tener apenas efecto sobre el rendimiento, mientras que si esta escritura bloquea la lectura de datos solicitados por el procesador, el tiempo de ejecución de una aplicación aumentará.

Cambio de política de consumo

Tal y como se acaba de explicar, las operaciones de escritura no están en el camino crítico de la ejecución, es decir, si se produce un retraso en una escritura, el tiempo de ejecución de un programa no se altera. No ocurre lo mismo con las lecturas. A pesar de que los microprocesadores actuales pueden realizar otras operaciones mientras están a la espera de un dato de memoria gracias a la ejecución fuera de orden y al multithreading, una espera demasiado larga puede perjudicar seriamente su rendimiento, aumentando proporcionalmente el tiempo necesario para finalizar la ejecución del programa. Por este motivo, una política que dé prioridad a los mensajes de escritura con respecto a las lecturas favorece a la reducción del tiempo que el procesador permanece a la espera del dato solicitado.

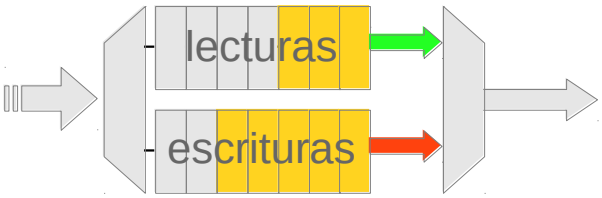

Figura 31 Peticiones de lectura y escritura separadas en dos buffers.

La utilización de dos buffers separados en el NIC de un banco de cache L2 permite separar las peticiones entrantes por su tipo, como muestra La Figura 31. Una vez cada tipo de petición ha sido aislado, se pueden proponer múltiples técnicas para gestionar el acceso al banco y priorizar las lecturas sobre las escrituras. Dado que el algoritmo que realice esta tarea está en el camino crítico del tiempo de acceso a memoria, hemos optado por utilizar una solución lo más sencilla posible. Siempre y cuando haya una petición en el buffer de lectura, será procesada antes que las alojadas en el buffer de escritura, independientemente del instante de su llegada. De esta forma, si una lectura llega a continuación de una escritura, será la lectura la primera de las dos en ser atendida en primer lugar. Como todo sistema basado en prioridades, es posible que las peticiones menos prioritarias sufran de inanición. En este caso, la llegada ininterrumpida de lecturas provoca una espera interminable por parte de las escrituras. Para evitarlo, cada vez que se llena uno de los dos buffers, se interrumpe la entrada de nuevas peticiones hasta que la ocupación de dicho buffer se reduce. De esta forma, si se llena el buffer de escritura, se interrumpe la llegada de nuevas peticiones (tanto de lectura, para evitar inanición, como de escritura, debido a la falta de espacio) y se procesan todas las existentes hasta que éste buffer se vacíe, conservando siempre el orden

de prioridad, es decir, primero se vacía el buffer de lecturas y, a continuación, el de escrituras.

Interrupción de la escritura

La utilización de un buffer dividido para priorizar las lecturas no evita que todavía existan paquetes de lectura que tengan que esperar demasiado tiempo. Ya se ha explicado que una escritura se procesa cuando no exista ninguna lectura pendiente en su buffer correspondiente. Sin embargo, en el caso de que se comience a procesar una escritura y, simultáneamente, llegue una petición de lectura al banco, la lectura ha de esperar a que el controlador haya sido liberado para, posteriormente, poder ser atendida. En el caso de una celda STT-RAM, esto supone una espera de más de 30 ciclos, lo que puede suponer un inconveniente importante.

Afortunadamente, la forma en la que se producen los cambios de valor en una celda STT-RAM durante una escritura nos ofrece la solución a este problema. El cambio de resistencia producido en una celda STT-RAM no es un proceso gradual. En realidad, el valor de la resistencia cambio de forma abrupta en los últimos ciclos de la escritura [51] [55]. Se puede observar este fenómeno mediante la evolución de la intensidad de corriente mostrada en la Figura 32. Por tanto, sería posible interrumpir una escritura durante gran parte de su desarrollo sin que ello afecte de modo alguno al dato previamente almacenado. Este fenómeno permite interrumpir una escritura para favorecer peticiones más prioritarias que lleguen demasiado tarde.

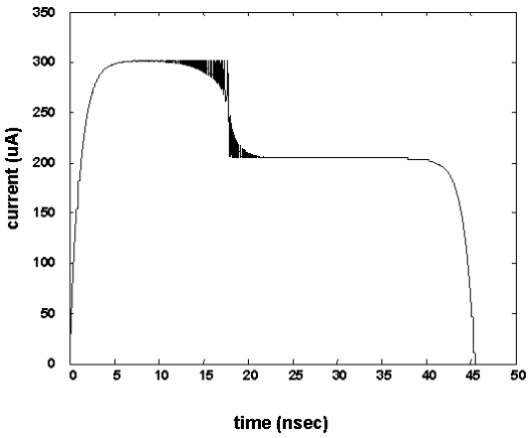

Figura 32 Evolución de la intensidad de corriente que atraviesa el MTJ durante una operación de escritura.

En resumen, es posible incluir nueva lógica a la interfaz de red del banco de memoria para favorecer el servicio a las lecturas y, por lo tanto, reducir el impacto negativo que tiene la

latencia de las escrituras. Mediante la separación de las peticiones permitimos a las lecturas adelantar a las escrituras a la hora de ser servidas y, en caso de que las primeras lleguen cuando el banco está comenzando a escribir, interrumpimos la operación para dar paso a la escritura. Para asegurar que el dato se mantiene inalterado tras una escritura interrumpida, solamente se detendrá el proceso de escritura si aun no han transcurrido 30 de los 35 ciclos que tarda.

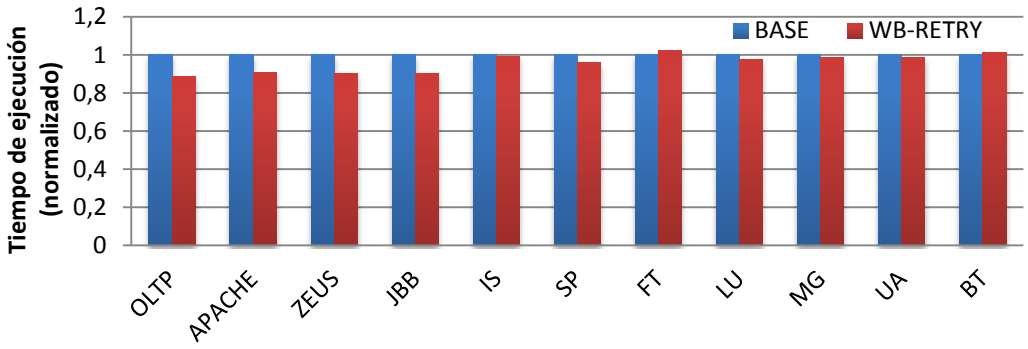

Figura 33 Efecto de la interrupción de las escrituras.

La Figura 33 muestra una comparación del rendimiento entre un consumidor habilitado para la priorización de lecturas e interrupción de escrituras y un consumidor convencional, ambos conectados a un banco de cache STT-RAM con 30 ciclos de escritura. Como se puede observar, con este sencillo mecanismo las aplicaciones con menor intensidad de escrituras (oltp, apache, zeus, jbb) son capaces de incrementar de forma significativa su rendimiento, superando en algunos casos el 10%. A medida que aumenta el número de escrituras, el efecto positivo se va disipando (ft, bt). Esto se debe a que el buffer de escritura se llena con mayor frecuencia y, por lo tanto, es preciso procesar todas las peticiones pendientes antes de continuar, aumentando en gran medida la latencia de las escrituras que todavía no entraron en el nodo, es decir, las peticiones acaban serializándose, arruinando las posibilidades de que las lecturas adelanten a las escrituras. Cabe destacar que los experimentos se han realizado con un tamaño de cache de primer nivel de 32Kbytes, lo que aumenta de forma significativa la presión sobre los bancos de L2. Actualmente el tamaño usual de este nivel de cache es un poco mayor, de 64Kbytes. Para una configuración de este tipo el mecanismo funcionaría mejor con total seguridad. Dados los plazos marcados para la realización de este proyecto, no hemos podido finalizar este experimento.

Finalmente, hemos llevado a cabo un estudio adicional de hasta qué punto era capaz de paliar nuestro mecanismo los largos tiempos de escritura de una cache STT-RAM. Para ello repetimos los resultados previos para diferentes valores de tiempo de escritura, desde 5

hasta 50 ciclos. En las figuras se muestra la diferencia en el deterioro del rendimiento según aumenta el tiempo de escritura (5, 10, 20, 30, 40 y 50 ciclos), tanto para el sistema base como para un sistema con priorización de lecturas e interrupción de escrituras. Los resultados sugieren que siempre que la frecuencia de escrituras no sea demasiado elevada el mecanismo funciona razonablemente bien. Si tomamos como ejemplo el caso de la aplicación oltp, sin realizar ninguna mejora, pasar de 5 a 50 ciclos el tiempo de escritura en L2 ralentiza en un 40% la ejecución de la aplicación. Con nuestra propuesta, somos capaces de reducir esa penalización en un 20%.

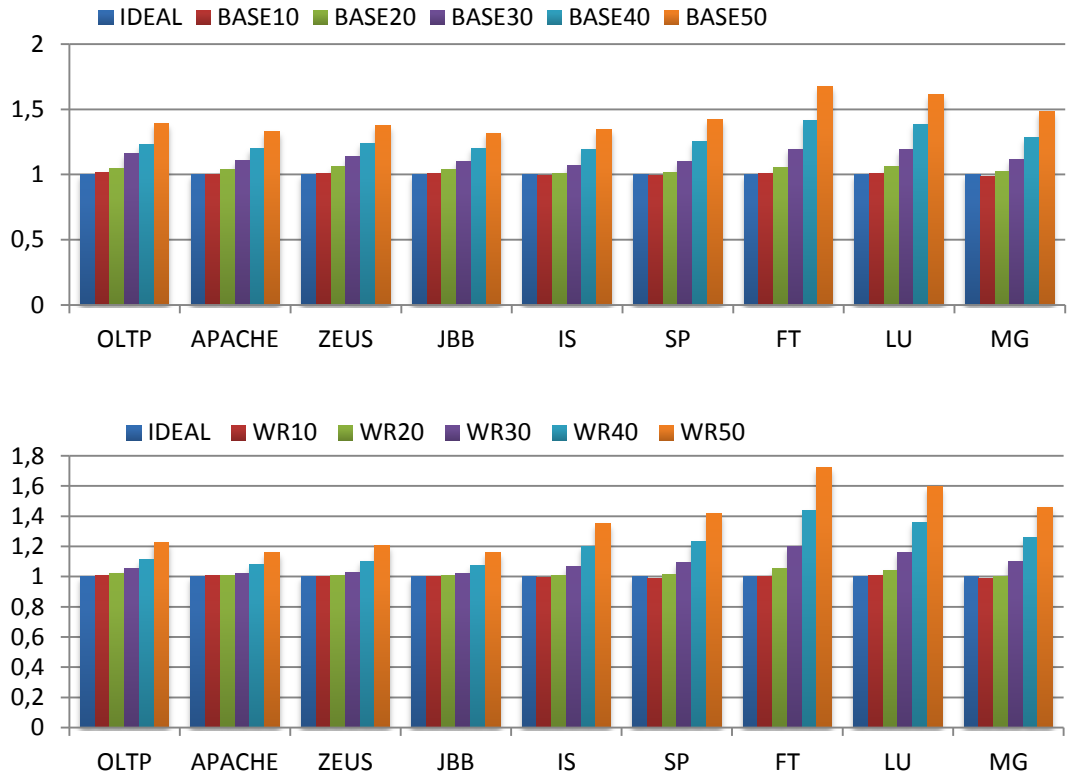

Figura 34 Efecto de la interrupción de escrituras para diferentes latencias de escritura. Arriba el sistema base. Abajo el sistema propuesto.

Dada su gran sencillez y sus buenos resultados, el mecanismo de interrupción de escrituras parece una propuesta interesante que puede dar respuesta a uno de los problemas más significativos que afectan a las memorias STT-RAM, las escrituras.

Capítulo 5. Optimizaciones adicionales

Para la realización de este proyecto se ha llevado a cabo un exhaustivo análisis de las características del sistema simulado. El simulador de red TOPAZ proporciona un conjunto de resultados que muestran con gran detalle las características del tráfico generado durante la ejecución de las aplicaciones. Esto nos ha permitido analizar en detalle estos resultados con el fin de buscar optimizaciones adicionales capaces de mejorar más el rendimiento. En este capítulo describimos de forma breve dos de esas optimizaciones evaluadas.

Algoritmos de encaminamiento alternativos

El protocolo de coherencia influye en gran medida en el comportamiento de la red durante el funcionamiento del sistema. Así, protocolos basados en broadcast, como el que aquí se utiliza, generan mayor cantidad de tráfico que protocolos basados en directorio. Como ya hemos explicado, en un protocolo basado en broadcast cada vez que un procesador desea modificar un dato compartido se lo debe comunicar al resto de procesadores. Adicionalmente, cuando un procesador requiere un dato para operar con él, realizará una petición al resto de procesadores por si alguno lo tuviera y al banco de L2 en el que debería estar. Esto provoca que, en un sistema de dos capas como el que aquí se propone, gran parte del tráfico generado proviene y se dirige a la capa de procesamiento.

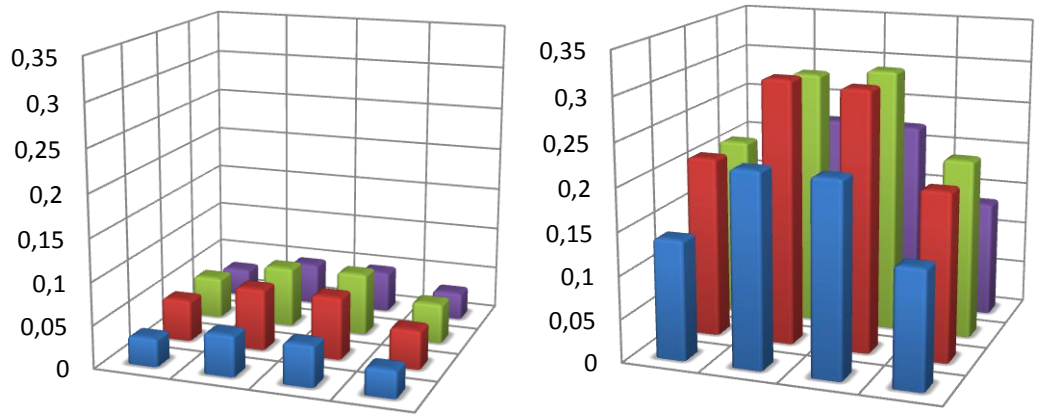

Figura 35 Utilización de los enlaces de la capa de cache (izquierda) y en la capa de procesamiento (derecha).

La utilización de los enlaces en este caso, por lo tanto, es bastante mayor en la capa de procesamiento (a) que en la capa de cache L2 (b), tal como muestra la Figura 35. En la misma figura podemos observar la mayor utilización de los enlaces en los nodos centrales que caracteriza a las redes en malla, como la que utiliza nuestro sistema (otras redes como los toros, equilibran con más eficacia el uso de los enlaces gracias a la incorporación de enlaces

periféricos). Esta congestión en la capa de procesamiento afecta principalmente a los mensajes que circulan por dicha capa que, debido a tener que "pelear" con un mayor número de mensajes por los recursos compartidos, ven aumentado el tiempo necesario para llegar a su destino.

Teniendo en cuenta que, como punto de partida, se utiliza un encaminamiento ordenado por dimensión XYZ, se puede optimizar el uso de la red y reducir la latencia promedio cambiando el rumbo que toman estos mensajes para equilibrar la carga global de la red. Dependiendo de dónde se crean y hacia dónde se dirijan podemos clasificarlos en tres grupos:

- Mensajes cuyo origen se encuentra en la capa de procesamiento, pero se dirigen a un nodo de la capa de cache. En este caso, en lugar de encaminar primero por la capa de procesamiento (XY) y, finalmente, realizar un movimiento vertical (Z) para llegar al destino, podemos abandonar cuanto antes la capa congestionada priorizando el movimiento vertical y, posteriormente, continuar con los movimientos horizontales, es decir, primero Z y luego XY. Las peticiones de lectura y escritura, por ejemplo, se encuentran dentro de este grupo. También pertenecen a este grupo una pequeña proporción de los mensajes de protocolo.
- Mensajes cuyo origen se encuentra en la capa de memoria y se dirigen a un nodo de la capa de cache. Estos mensajes pueden conservar el enrutamiento XYZ ya que no abandonan la capa más libre hasta realizar el último salto. Pertenecen a este grupo las respuestas de las peticiones de lectura, que llevan el dato de la memoria al procesador correspondiente.
- Mensajes cuyo origen y destino pertenecen a la misma capa de procesamiento. Dependiendo de la distancia que se necesita recorrer puede ser rentable realizar dos movimientos verticales: uno al comienzo, para abandonar la capa, y otro al final, para regresar a la misma. Por ejemplo, para mensajes dirigidos a nodos adyacentes es contraproducente, ya que la distancia a recorrer se duplica. Sin embargo, si se necesita recorrer la capa de extremo a extremo puede ser útil incrementar el número de saltos si con ello conseguimos reducir lo suficiente la latencia total del mensaje. Este grupo se compone principalmente por los mensajes del protocolo de coherencia y transferencias de datos entre caches.

Cada grupo se forma por tipos de mensajes diferentes, con alguna excepción como los mensajes del protocolo de coherencia. Como cada tipo de mensaje ocupa un canal virtual diferente para evitar el message-dependent deadlock, el algoritmo de enrutamiento se mantiene constante para todos los mensajes que comparten el mismo canal virtual. Por esta razón, se evita la formación de dependencias cíclicas en la red y el bloqueo de los mensajes de forma permanente (deadlock). Con esta forma de encaminar los mensajes es posible utilizar los enlaces de la red de forma diferente para cada tipo de mensaje y balancear, con ello, la carga global de la red sobre las dos capas.

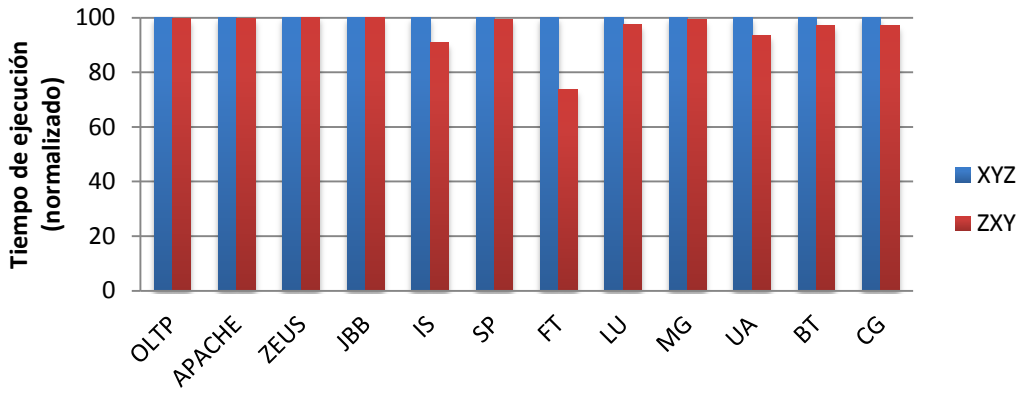

Figura 36 Efecto del orden de encaminamiento.

Para este proyecto solamente hemos tenido tiempo de realizar un experimento que muestra las posibles ventajas de enrutar de manera más inteligente, utilizando solamente los dos primeros grupos descritos anteriormente. Los resultados de este experimento se muestran en la Figura 36. Ésta muestra un efecto muy diferente dependiendo de la cantidad de tráfico generado por la aplicación. Así, aplicaciones como oltp o apache, en las que se genera poco tráfico, encuentran poca congestión en sus comunicaciones, por lo que un cambio de enrutamiento apenas las afecta. Sin embargo, cuando la presión sobre la red es elevada, como en el caso de ft, este sencillo cambio puede suponer una diferencia de más del 20% en su rendimiento.

Early restart

Existen múltiples optimizaciones avanzadas que permiten incrementar el rendimiento de las memorias cache, permitiendo al procesador dedicar menor cantidad de tiempo a esperar por el dato que solicitó. Cuando se produce un fallo en una lectura de cache, en lugar de traer únicamente el dato que se solicita, se trae el bloque completo del que forma parte. De esta forma se explota la localidad espacial de los datos, que consiste en que los datos anexos al que se accede tienen mayor probabilidad de cargarse, como ocurre con estructuras de datos como arrays por ejemplo. Debido a que se carga el bloque completo, de mayor tamaño que el dato cargado, el tiempo de transferencia es mayor que si se cargara únicamente el dato solicitado. Por este motivo, existen dos optimizaciones que permiten al procesador continuar con su trabajo sin ni siquiera haber terminado de transferir el bloque completo. La técnica conocida como Critical Word First consiste en enviar primero el dato que busca el procesador, que puede continuar su ejecución, y continuar la transferencia del bloque en segundo plano. De forma similar, en el caso de que la lectura de las palabras de cada bloque no se pueda efectuar de forma desordenada, la alternativa llamada Early Restart procesa las

palabras en el orden normal pero envía el dato crítico al procesador tan pronto como éste se haya leído. El beneficio que aporta esta mejora es proporcional al tamaño del bloque de cache. Con tamaños de bloque muy grandes, la diferencia entre utilizar o no una de estas dos mejoras es bastante significativa. Por el contrario, con tamaños de bloque pequeños las mejoras pueden ser insignificantes, de tal modo que añadir lógica puede no ser viable.

Así como entre el procesador y los datos con los que opera, esta técnica también puede ser explotada para los mensajes que se generan entre las diferentes controladoras de cache. De esta forma, cuando los mensajes deben ser descompuestos en fracciones de menor tamaño y su envío debe ser serializado, podemos ordenar de forma inteligente esos fragmentos. Si solamente vamos a utilizar parte del contenido de dicho mensaje y somos capaces de colocarlo en primer lugar, disminuiremos el tiempo de espera. Para finalizar con este proyecto hemos analizado el efecto que tienen estas dos técnicas en un sistema con STT-RAM. Se trata de un análisis preliminar, intentando ver cuál es la cota superior de ganancia. Este análisis es optimista, es decir, supone que, en todo momento, la palabra crítica se encuentra en el comienzo del bloque.

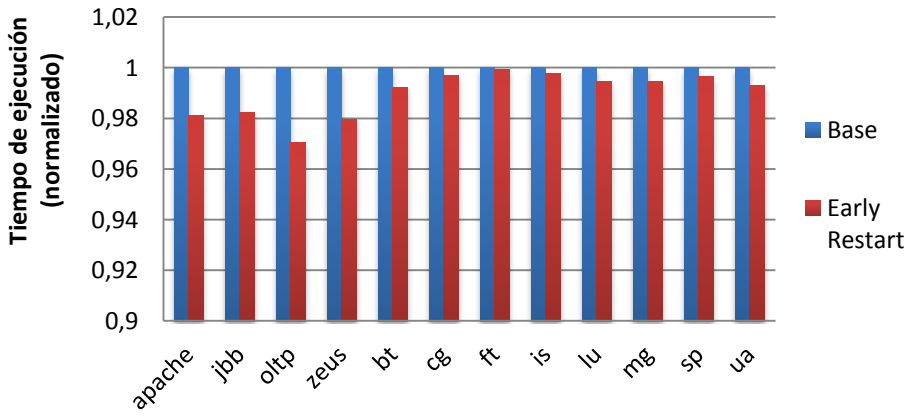

Figura 37 Comparación entre un sistema base, con STT-RAM, y un sistema optimizado con Early Restart.

La Figura 37 muestra, por tanto, la comparación optimista del rendimiento entre un sistema con STT-RAM y el mismo optimizado con la mejora CWF/ER. Como se puede observar, la mejora es casi inapreciable, llegando en algunos casos a superar apenas el 2%. Esto se debe principalmente a dos motivos. Primero, el tamaño de bloque utilizado en el trabajo es de 64B. El hecho de que el tamaño no sea mayor reduce en gran medida la magnitud de la mejora introducida. Por otra parte, gran parte de los mensajes que se utilizan en este tipo de sistemas tienen un tamaño suficientemente pequeño como para que no haga falta

descomponerlos en varios fragmentos. Probablemente esta propuesta tenga mayor sentido en sistemas con mayor tamaño de bloque y con menor ancho de banda, lo que incrementará el número de mensajes que deban ser descompuestos.

Capítulo 6. Conclusiones y trabajo futuro

6.1 3D Stacking

La aparición del 3D-Stacking se puede convertir en un punto de inflexión para el diseño de dispositivos electrónicos. Gracias a esto, cada vez más dispositivos se aprovechan de la mayor densidad de integración en menor área que esta técnica ofrece. Sin ir más lejos, con la definición del nuevo estándar de almacenamiento flash SD-XC, ya han surgido algunos productos que incorporan hasta 64GB de capacidad en apenas milímetros, gracias al apilamiento de hasta 9 capas. A corto plazo, el apilado 3D permitirá evitar el problema del Bandwith Wall introducido con los multiprocesadores, acumulando una o varias capas de DRAM por encima de la capa de cores o distribuyendo memoria y procesadores a lo largo de las diferentes capas. Por estas razones en este proyecto hemos creído conveniente realizar todos nuestros experimentos con sistemas que ya hagan un uso de esta tecnología, aunque sea de forma muy moderada, con dos capas solamente.

6.2 Almacenamiento no volátil

Durante la elaboración del proyecto, se ha podido observar que las tecnologías de almacenamiento no volátil permiten reducir en una proporción muy grande el consumo energético producido por las corrientes de pérdidas, algo que, a los niveles de integración de hoy en día, caracteriza de forma significativa a los chips microelectrónicos. Tecnologías como STT-RAM y PCRAM además de reducir significativamente el consumo estático, permiten superar en otros aspectos a las tecnologías convencionales (SRAM y DRAM), como la densidad de integración y el rendimiento. La barrera más importante que separa a estas nuevas tecnologías de su comercialización es el coste, tanto temporal como energético, de las escrituras. Por ello, una parte importante de la investigación se centra en la propuesta de soluciones, tanto a nivel físico como a nivel de diseño. Todas estas propuestas pueden dividirse en dos grandes grupos. En el primero, se centra en la reducción del volumen de escrituras que se realizan en las celdas STT-RAM, que varían desde una modificación del algoritmo de reemplazo hasta la restricción a nivel de bit de las escrituras que no modifican el estado de la celda. Por otra parte, el resto de propuestas se centran en alterar la estructura física de la propia celda modificando, por ejemplo, el tiempo de retención. Nuestra propuesta ha intentado dar una respuesta con dos claras ventajas sobre el resto. En primer lugar, hemos intentado proporcionar una solución que no implica el uso de tecnologías mixtas en la misma capa, mezclando celdas SRAM con STT-RAM. Esto puede suponer una gran ventaja a la hora

de simplificar las labores de diseño y fabricación de una capa en un sistema 3D. En segundo lugar, nuestra solución se caracteriza por ser muy poco "invasiva". Gran parte de las propuestas requieren modificar de forma importante aspectos tan críticos de un sistema como puede ser el protocolo de coherencia. En nuestro caso, únicamente el buffer del interfaz de red requiere modificaciones para funcionar.

Está claro que las tecnologías de almacenamiento no volátil aquí expuestas necesitan madurar pero, tarde o temprano, su instalación en computadores, desde centros de computación de alto rendimiento hasta sistemas portátiles, parece inevitable. Con este trabajo intentamos aportar nuestro grano de arena para reducir el impacto de las desventajas en este tipo de tecnología.

6.3 Trabajo futuro

Este proyecto es un trabajo en el que hemos partido de cero. Han sido necesarias muchas horas de lectura y búsqueda para comprender cómo funciona una celda STT-RAM y qué hay propuesto hasta el momento. Se trata de un campo relativamente novedoso, de hecho, solamente otra propuesta, que se ha evaluado en el proyecto, intenta solucionar el problema desde un punto de vista similar. Por tanto, la ampliación del trabajo se puede desarrollar en múltiples direcciones, dependiendo de los objetivos que se deseen cumplir.

El mecanismo de priorización presentado es relativamente sencillo. Evaluar otras propuestas capaces de ordenar los accesos al banco de forma más inteligente podría ser un buen punto de partida.

Debido al mayor consumo energético que se ha introducido con la interrupción de las escrituras, un análisis de la incorporación de una propuesta existente que reduzca el consumo energético al simulador de sistema completo que hemos estado utilizando, puede ser una continuación bastante viable. Por ejemplo, la reducción de los writebacks producida por un algoritmo de reemplazo diferente [34] puede reducir el impacto negativo que aplicaciones de escritura intensiva pueden tener sobre nuestra propuesta.

La utilización de herramientas CAD específicas para modelar los distintos componentes del sistema permitiría ajustar de forma más precisa las características de los mismos, aumentando la precisión de las simulaciones y acercando aún más los resultados a los que posiblemente se obtendrían en realidad

Finalmente, las soluciones analizadas en el Capítulo 5 también podrían ser fácilmente ampliadas. Múltiples variaciones del algoritmo de encaminamiento son posibles para seguir mejorando la forma de distribuir el ancho de banda disponible. Adicionalmente, es posible evaluar el mecanismo CWF/ER en entornos donde su utilización reporte mejores resultados.

Capítulo 7. Trabajos citados

[1] G. E. Moore, «Cramming More Components onto Integrated Circuits,» 1965.

[2] G. E. Moore, «Progress In Digital Integrated Electronics,» 1975.

[3] «Standard Performance Evaluation Corporation (SPEC),» [En línea]. Available: www.spec.org. [Último acceso: 06 09 2012].

[4] Intel Corporation, «i960 CA/i960 CF 32-Bit Superscalar Embedded Microprocessor,» 1994. [En línea]. Available: http://datasheets.chipdb.org/Intel/80960/PRODBREF/272211_3.PDF. [Último acceso: 1 9 2012].

[5] Intel Corporation, «Intel 64 and IA-32 Architectures Optimization Reference Manual,» 04 2012. [En línea]. Available: http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf. [Último acceso: 01 09 2012].

[6] IBM Microelectronics, Motorola, «PowerPC 601™ RISC Microprocessor Hardware Specifications,» 1995. [En línea]. Available: http://www.datasheetcatalog.org/datasheet/motorola/MPC601.pdf. [Último acceso: 01 09 2012].

[7] D. W. Wall, «Limits of Instruction-level Parallelism,» *WRL Research Report 93/6, Digital Western Research laboratory, Palo Alto, CA,* 1993.

[8] M. Flynn, P. Hung y K. Rudd, «Deep-Submicron Microprocessor Design,» *IEEE Micro,* Julio de 1999.

[9] K. Olukotun y L. Hammond, «The Future of Microprocessors,» *ACM QUEUE Magazine,* Septiembre de 2005.

[10] K. Olukotun y L. Hammond, «The Future Of Microprocessors,» 2005.

[11] D. Wang, «Meeting Green Computing Challenges,» 2008.

[12] «The Data Center Power and Cooling Challenge,» Noviembre de 2007.

[13] M. D. Hill y M. R. Marty, «Amdahl's Law in the Multicore Era,» Julio de 2008.

[14] IBM, «Power 4, The First Multi-Core, 1GHz Processor,» [En línea]. Available: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/power4/. [Último acceso: 29 08 2012].

[15] Intel Corporation, «Introduction of 45nm Next-Gen Intel® Core™ Microarchitecture,» 2007. [En línea]. Available: http://www.intel.com/content/dam/doc/white-paper/45nm-next-generation-core-microarchitecture-white-paper.pdf. [Último acceso: 29 08 2012].

[16] Intel Corporation, «First the Tick, Now the Tock: Intel® Microarchitecture (Nehalem),» 2009. [En línea]. Available: http://www.intel.com/content/dam/doc/white-paper/intel-microarchitecture-white-paper.pdf. [Último acceso: 29 08 2012].

[17] Intel Corporation, «An Introduction to the Intel® QuickPath Interconnect,» Enero 2009. [En línea]. Available: http://www.intel.com/content/dam/doc/white-paper/quick-path-interconnect-introduction-paper.pdf. [Último acceso: 29 08 2012].

[18] A. Agarwal, «The Tile processor: A 64-core multi-core for embedded processing,» 2007.

[19] M. A. e. al., «Integration Challenges and Tradeoffs for Tera-scale,» vol. 11, nº 03, 2007.

[20] W. A. Wulf y S. A. McKee, «Hitting the Memory Wall: Implications of the Obvious,» vol. 23, nº 1, Marzo de 1995.

[21] B. Rogers, A. Krishna, G. Bell, K. Vu, X. Jian y Y. Solihin, «Scaling the badwidth wall: challenges in and avenues for CMP scaling,» Junio de 2009.

[22] N. Magen, A. Kolodny, U. Weiser and N. Shamir, "Interconect-Power Dissipation in a Microprocessor," in *Proc. 2004 Int. Workshop System Level Interconnect Prediction Session (ACM, 2004) Interconnect Anal. SoCs Microprocess*.

[23] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jian, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen and C. Webb, "Die Stacking (3D) Microarchitecture".

[24] K. Takahashi y M. Sekiguchi, «Through Silicon Via and 3-D Wafer/Chip Stacking Technology,» 2006.

[25] K. Saban, «Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency,» 21 11 2011. [En línea]. Available: http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf.

[26] K. Cioffi y W.-T. Hsu, «32KHz MEMS-Based Oscillator for Low-Power Applications,» 2005.

[27] R. Drost y I. Sutherland, «Proximity communication and time [capacitively coupled IC communication],» 2005.

[28] S. Mick, J. Wilson y P. Franzon, «4 Gbps High-Density AC Coupled Interconnection,» 2002.

[29] V. Suntharalingam, R. Berger, J. Burns, C. Chen, C. Keast, J. Knecht, R. Lambert, K. Newcomb, D. O'Mara, D. Rathman, D. Shaver, A. Soares, C. Stevenson, B. Tyrrell, K. Warner and B. Wheeler, "Megapixel CMOS Image Sensor Fabricated in Three-Dimensional Integrated Circuit Technology".

[30] S. Gupta, M. Hilbert, S. Hong and R. Patti, "Techniques for Producing 3D ICs with High-Density Interconnect," 2004.

[31] G. L. Loi, B. Agarwal, N. Srivastava, S.-C. Lin and T. Sherwood, "A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hirearchy.," 2006.

[32] J. Daughton, "Thin Solid Films," in *Magnetoresistive Memory Technology*, 1992.

[33] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane and H. Yamada, "A Novel Nonvolatile Memory with Spin Torque Transfer Magnetization Switching: Spin-RAM," 2005.

[34] M. Rasquinha, D. Choudhary, S. Chatterjee, S. Mukhopadhyay and S. Yalamanchili, "An Energy Efficient Cache Design Using Spin Torque Transfer (STT) RAM".

[35] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li y Y. Chen, «Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement,» 2008.

[36] X. Wu, J. Li, L. Zhang, E. Speight y Y. Xie, «Power and Performance of Read-Write Aware Hybrid Caches with Non-volatile Memories».

[37] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi y M. Stan, «Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches,» 2011.

[38] Sun microsystems, «OpenSPARC™ Microarchitecture Specification,» [En línea]. Available: http://www.opensparc.net/pubs/t2/docs/OpenSPARCT2_Core_Micro_Arch.pdf. [Último acceso: 01 09 2012].

[39] D. Wentzlaff, P. Griffin, H. Hoffman, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III and A. Agarwal, "On-Chip Interconection Architecture of the Tile Processor," 2007.

[40] W. J. Dally y B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers, 2004.

[41] B. Rogers, A. Krishna, G. Bell, K. Vu, X. Jian y Y. Solihin, «Scaling the bandwidth wall: challenges in and avenues for CMP scaling,» 2009.

[42] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill y D. A. Wood, «Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset,» vol. 33, nº 4, 2005.

[43] A. Alameldeen, M. Martin, C. Mauer, K. Moore, M. Xu, M. Hill, D. Wood y D. Sorin, «Simulating a $2M commercial server on a $2K PC,» vol. 36, nº 2, 2003.

[44] NASA Advanced Supercomputing Division, «NAS Parallel Benchmarks,» [En línea]. Available: http://www.nas.nasa.gov/publications/npb.html. [Último acceso: 10 09 2012].

[45] M. Rasquinha, DhruvChoudhary, S. Chatterjee, S. Mukhopadhyay y S. Yalamanchili,

«An energy efficient cache design using Spin Torque Transfer (STT) RAM,» 2010.

[46] X. Wu, J. Li, L. Zhang, E. Speight y Y. Xie, «Power and performance of read-write aware Hybrid Caches with non-volatile memories,» 2009.

[47] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony y Y. Xie, «Hybrid cache architecture with disparate memory technologies,» 2009.

[48] J. Li, C. Xue y Y. Xu, «STT-RAM based energy-efficiency hybrid cache for CMPs,» 2001.

[49] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi y M. Stan, «Relaxing non-volatility for fast and energy-efficient STT-RAM caches,» 2011.

[50] Z. Sun, X. Bi, H. (. Li, W.-F. Wong, Z.-L. Ong y X. W. W. Zhu, «Multi retention level STT-RAM cache designs with a dynamic refresh scheme,» 2011.

[51] P. Zhou, B. Zhao, J. Yang y Y. Zhang, «Energy reduction for STT-RAM using early write termination,» 2009.

[52] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan y a. C. R. Das, «Architecting On-Chip Interconnects for Stacked 3D STT-RAM Caches in CMPs,» 2011.

[53] P. Abad, P. Prieto, L. Menezo, A. Colaso, V. Puente y J.-A. Gregorio, «TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers,» 2012.

[54] A. Hansson, K. Goossens y A. Radulescu, «Avoiding Message-Dependent Deadlock in Network-Based Systems on Chip,» 2007.

[55] Y. Chen, X. Wang, H. Li, H. Liu y D. V. Dimitrov, «Design Margin Exploration of Spin-Torque Transfer RAM (SPRAM),» 2008.

[56] V. N. Johnson, J. Jozwiak and A. Moll, "Through Wafer Interconnects on Active pMOS Devices," 2004.

[57] A. V. Pohm, C. S. Comstock y A. T. Hurst, «Quadrupled nondestructive outputs from magnetoresistive memory cells using reversed word fields,» 1990.