UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament d'Arquitectura de Computadors

RECURSOS ANCHOS: UNA TÉCNICA DE BAJO COSTE PARA EXPLOTAR PARALELISMO AGRESIVO EN CÓDIGOS NUMÉRICOS

Autor: David López Alvarez
Directores: Mateo Valero Cortés
Josep Llosa i Espuny

Capítulo 5.
Rendimiento de las técnicas

5.1 Introducción

En el capítulo 3 estudiábamos diversas técnicas para mejorar el rendimiento de bucles planificados con segmentación software. Para aquellos bucles limitados por los recursos se presentaba una técnica utilizada en varios procesadores actuales (replicación) y una alternativa de menor coste (*widening*). Para bucles limitados por las recurrencias se proponía usar unidades funcionales complejas capaces de implementar la operación FMA (*Fussed Multiply-and-Add*). En este capítulo vamos a estudiar el rendimiento de dichas técnicas para procesadores VLIW y código numérico utilizando los bucles del *Perfect Club*.

Primero presentaremos un estudio del rendimiento teórico que se puede alcanzar con estas técnicas. Para ello supondremos una planificación perfecta, bancos de infinitos registros y una memoria perfecta (100% *hit ratio* en la cache). Estudiaremos el rendimiento de un amplio rango de configuraciones donde se aplica replicación, *widening* y ambas técnicas simultáneamente. Estudiaremos qué parte del incremento de rendimiento es debido a la

utilización de las mencionadas técnicas en las unidades de memoria y qué parte es provocado por la aplicación de las técnicas en las unidades funcionales de coma flotante. También veremos cómo afecta al rendimiento el uso de unidades funcionales capaces de implementar la operación FMA.

Hay ciertos factores que limitan el rendimiento teórico al que nos referíamos en los párrafos anteriores. Si consideramos el uso de un banco de registros limitado, nos podemos encontrar con que el número de registros necesarios para la ejecución del bucle sea superior al número de registros físicos disponibles. En este caso añadiremos *spill code*, lo que puede redundar en una pérdida de rendimiento.

Por otro lado, nos encontraremos con que una planificación real puede usar más ciclos del mínimo teórico (*Minimum Initiation Interval, MII*). Esto es debido a tres factores:

- el *MII* es una cota inferior, pero no siempre es alcanzable.
- el problema de la planificación es NP-completo, por lo que empleamos heurísticas para resolverlo. Estas heurísticas pueden no encontrar la planificación óptima.
- el *spill code* dificulta la planificación, lo que puede producir que el planificador se aleje más todavía de la planificación óptima.

Así pues, estudiaremos también el rendimiento cuando los bucles son planificados con restricciones en el número de registros disponibles, y veremos qué parte de la pérdida de rendimiento respecto al teórico es debida al *spill code* y qué parte es debida al factor planificador.

Para todas las configuraciones del capítulo hemos supuesto el siguiente modelo de tiempos: todas las operaciones (load, add, mul,...) tienen una latencia de 4 ciclos y están absolutamente segmentadas, excepto las operaciones store (latencia de 1 ciclo), división (latencia de 19 ciclos, no segmentada) y la operación raíz cuadrada (latencia de 27 ciclos, no segmentada).

5.2 Límites teóricos

5.2.1 Comportamiento de los buses anchos

Culture is the widening of the mind and of the spirit.

Jawahalal Nehru

Primero estudiaremos el efecto del uso de replicación y *widening* en los buses. Para ello tomaremos configuraciones donde el número de FPUs ha sido fijado a 2048 (lo que, a efectos prácticos es como si fueran infinitas FPUs) y veremos el rendimiento teórico de diversas configuraciones de buses (figura 5.1). Nuestra configuración base será una arquitectura con un 1 bus y 1 unidad funcional multifunción (FPU) ambos de ancho 1, y con las latencias descritas en la introducción del capítulo. En esta figura, las configuraciones estudiadas reciben el nombre de **XwY** donde **X** es el grado de replicación e **Y** es el grado de *widening* de los buses (es decir, tienen un ancho de banda de **X*Y** palabras). A cada configuración se le ha aplicado un grado de *unrolling* igual a su ancho de banda. Así pues, una configuración 4w1 tiene 4

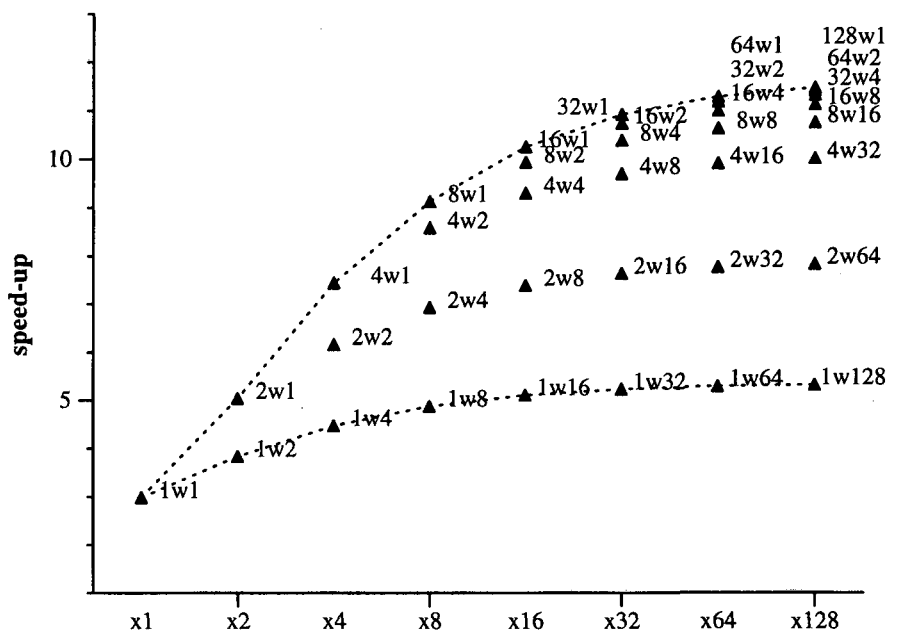

Figura 5.1: Rendimiento de diferentes configuraciones de bus. Todas las configuraciones tienen 2048 FPUs. La configuración base tiene 1 FPU y 1 bus, ambos de ancho 1

buses de ancho 1. Dicha configuración tiene un ancho de banda 4 veces superior al de la configuración base; otras configuraciones con el mismo ancho de banda son la 2w2 y la 1w4. De esta manera, las configuraciones han sido agrupadas según su ancho de banda (**X*Y**), desde configuraciones con ancho de banda igual al de la configuración base hasta configuraciones con ancho de banda 128 superior al de la base.

En la mencionada figura 5.1 se pueden observar ciertos efectos:

- en las configuraciones donde sólo se modifica el grado de replicación (configuraciones **Xw1**, la línea discontinua superior) el incremento de rendimiento va disminuyendo conforme aumenta el grado de replicación, especialmente a partir del momento en que disponemos de 16 buses. Este efecto es debido a las recurrencias. Teniendo en cuenta que tenemos virtualmente infinitas FPUs, no habrá ningún bucle *compute-bound.* Así, conforme aumentamos el número de buses habrá una tendencia a que el número de bucles *memory-bound* vayan disminuyendo, de manera que la mayoría pasen a estar limitados por las recurrencias. A partir de que un bucle es *recurrence-bound*, su rendimiento no mejorará aun disponiendo de infinitos recursos. Como hay bucles *memory-bound* que no tienen recurrencias, el rendimiento sigue aumentando, pero cada vez menos.

- las configuraciones donde sólo se modifica el grado de *widening* (configuraciones **1wY**, la línea discontinua inferior) tienen el mismo efecto, pero más acusado. Esto es debido a que sólo disponemos de un bus, aunque de anchura creciente, de manera que las operaciones de memoria no compactables no pueden ejecutarse en paralelo y sólo las operaciones compactables se benefician del incremento del ancho de banda. Se puede observar que configuraciones con grado de *widening* superior a 16 tienen un incremento de rendimiento prácticamente despreciable.

- por lo que respecta a las configuraciones intermedias, podemos observar como, con un grado de replicación fijo, el incremento de rendimiento de aplicar *widening* tiene una tendencia a decrecer. Sin embargo hay ciertos casos que conviene estudiar más a fondo,

como la configuración 4w4, que tiene un rendimiento teórico mayor que 8w1, por lo que habría que realizar un estudio del coste de ambas configuraciones para averiguar cuál es la mejor.

- otro efecto observado es que, en configuraciones con ancho de banda superior a 16 veces el de la base, todas las configuraciones con 16 buses o más tienen un rendimiento similar, independientemente del grado de replicación o de *widening*.

De las observaciones anteriores deducimos que, por lo que respecta al bus, configuraciones con ancho de banda superior a 16 veces el de la base tienen un incremento de rendimiento tan pequeño que no compensa el incremento de coste que podemos intuir que lleva asociado.

Un punto interesante es que la configuración 1w1 (que tiene 2048 FPUs) tiene un rendimiento tres veces superior al de la configuración base (que tiene el mismo bus, pero sólo

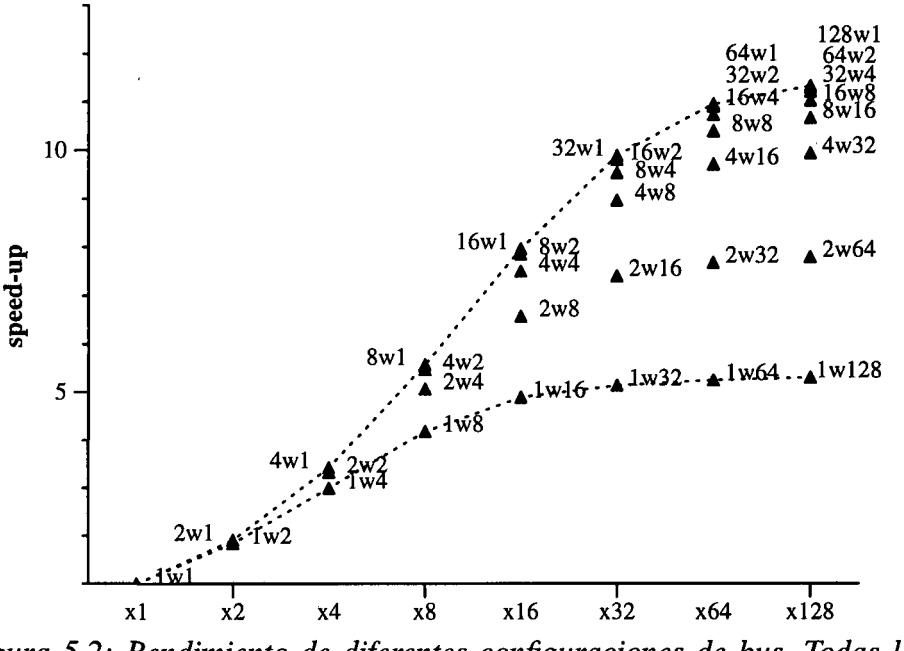

Figura 5.2: Rendimiento de diferentes configuraciones de bus. Todas las configuraciones tienen tantas FPUs como su ancho de banda total. La configuración base tiene 1 FPU y 1 bus, ambos de ancho 1

una FPU, de lo cual deducimos que, a igualdad de recursos de memoria y aritméticos, los bucles del *Perfect Club* son ligeramente *compute-bound.* Dado que en una configuración real no podemos disponer de infinitas FPUs, estudiaremos a continuación el rendimiento de aplicar replicación y *widening* a los buses, pero con FPUs finitas.

La figura 5.2 muestra el rendimiento teórico para varias configuraciones de bus **X**w**Y**, donde sólo se aplica a las FPUs la técnica de replicación. Una configuración **X**w**Y** tiene **X** buses de ancho **Y**, y **X*Y** FPUs. La figura 5.3 muestra el mismo estudio, pero en este caso hay una relación entre el número de FPUs y el ancho de banda de 2 a 1; es decir, una configuración **X**w**Y** tiene 2***X*****Y** FPUs.

Podemos observar cómo el comportamiento del rendimiento de las configuraciones es muy similar en los dos casos, y a su vez, muy similar al mostrado con infinitas FPUs (figura 5.1), por lo que las conclusiones anteriores también se aplican con un número de FPUs finito.

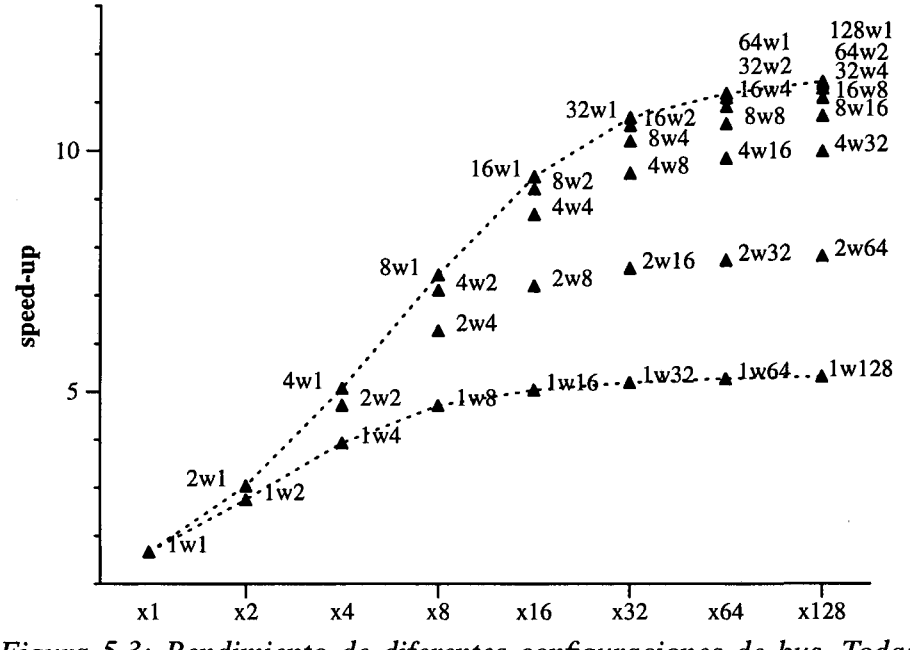

*Figura 5.3: Rendimiento de diferentes configuraciones de bus. Todas las configuraciones tienen un número de FPUs= 2*X*Y. La configuración base tiene 1 FPU y 1 bus, ambos de ancho 1*

Estudiando la figura 5.1 ya se observó que los bucles del *Perfect Club* son ligeramente *compute-bound*. Viendo los resultados de las figuras 5.2 y 5.3, podemos concluir que, para código numérico, una relación 2 a 1 entre el número de operaciones en coma flotante que se pueden lanzar por ciclo y el ancho de banda resulta una buena combinación: la configuración 1w1 con 2 FPUs tiene un rendimiento 1.66 superior al de 1w1 con 1 FPU, aunque conforme crece el grado de replicación y *widening* de la configuración, esta diferencia de rendimiento se va reduciendo.

5.2.2 Comportamiento de las unidades funcionales anchas

¿Servicio de habitaciones? Mándenme una habitación más grande.

Julius "Groucho" Marx

En este punto, vamos a repetir el estudio del punto anterior, pero para las FPUs. Así, lo primero que haremos será estudiar diversas configuraciones **X**w**Y**, donde **X** e **Y** siguen siendo los grados de replicación y widening, respectivamente, de las FPUs, pero con un número de buses de 2048 (lo que supone virtualmente un número infinito de buses). Este estudio se muestra en la figura 5.4.

Podemos observar que el rendimiento de las configuraciones **X**w**Y** depende más del número de operaciones que se pueden lanzar por ciclo que del grado de replicación y de *widening*. Esto es debido a que las restricciones a la hora de compactar operaciones aritméticas son mucho más suaves que a la hora de compactar operaciones de memoria (compactamos instanciaciones de la misma operación en iteraciones consecutivas siempre y cuando no haya dependencia entre ellas, mientras que en el caso de operaciones de memoria, además es necesario que los accesos tengan *stride* 1).

Asimismo, observamos que la configuración 1w1 y la base tienen prácticamente el mismo rendimiento, a pesar de que la base tiene infinitos buses y 1w1 un sólo bus. Esto es debido a que los bucles estudiados son en general *compute-bound*, con lo que, al disponer de una sola FPU, el rendimiento con 1 bus y con infinitos buses es muy similar. Debido a esta

característica, el estudio de rendimiento de configuraciones donde el número de buses crece proporcionalmente al número de operaciones aritméticas que se pueden lanzar por ciclo, da resultados prácticamente indistinguibles de los mostrados en la figura 5.4.

De los efectos observados en este punto y en el anterior, podemos extraer las siguientes conclusiones:

- las técnicas de replicación y *widening* ofrecen rendimientos similares cuando se aplican sobre las FPUs, mientras que cuando se aplican sobre los buses, los resultados son muy diferentes. Es, por tanto, el comportamiento sobre los buses de una técnica el que definirá su rendimiento teórico.

- los bucles numéricos (como los del *Perfect Club*) suelen ser ligeramente *compute-bound*, de manera que una configuración capaz de realizar dos operaciones aritméticas por cada operación de memoria resulta más balanceada que una con una

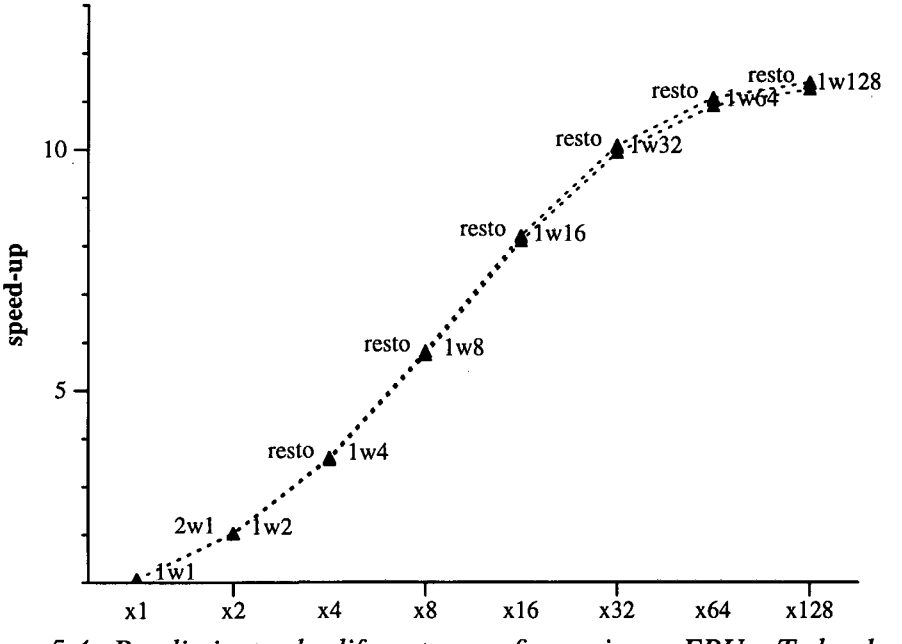

Figura 5.4: Rendimiento de diferentes configuraciones FPUs. Todas las configuraciones tienen un número de buses= 2048. La configuración base tiene 1 FPU y 1 bus, ambos de ancho 1

relación uno a uno. Dado que una configuración balanceada nos permitirá evaluar mejor la bondad de una técnica, a partir de este momento usaremos configuraciones con una relación de dos recursos aritméticos por cada recurso de memoria.

5.2.3 Límites de la técnica *widening*

Ponte un límite y, ciertamente, lo tendrás.

Anónimo

En los puntos anteriores vimos cómo afectaba la utilización de las técnicas de replicación y *widening* aplicadas a las FPUs o a los buses. A continuación veremos qué límites tiene la técnica de *widening* aplicada a FPUs y buses de manera simultánea.

La figura 5.5 muestra el rendimiento de aplicar grados de *widening* entre 1 y 128 a una configuración con 2 FPUs y 1 bus. Podemos ver cómo al principio el incremento de rendimiento es bastante elevado, mientras que hay un punto en que el rendimiento conseguido

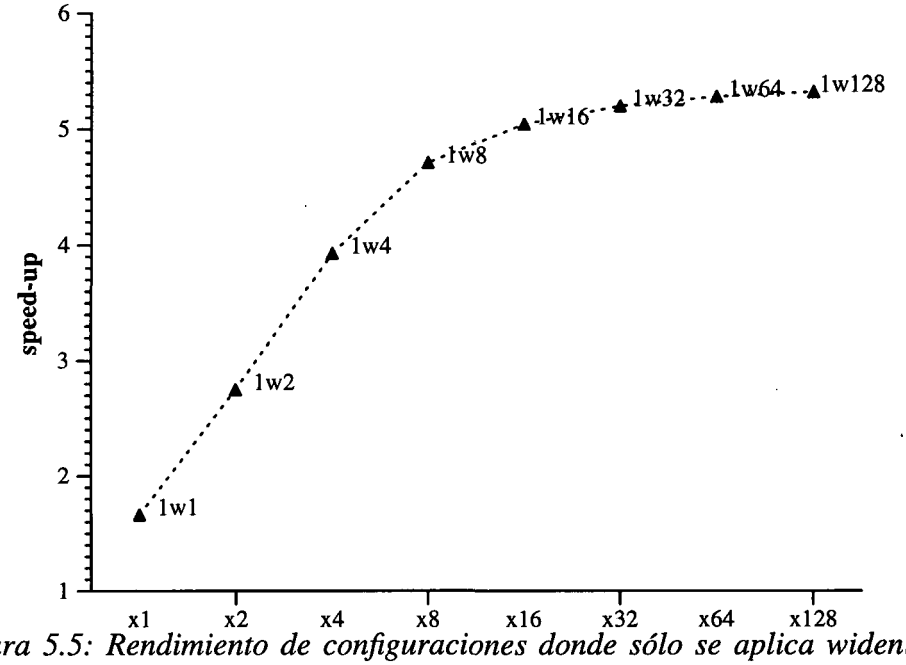

Figura 5.5: Rendimiento de configuraciones donde sólo se aplica widening (tanto a FPUs como a buses). La configuración base tiene 1 FPU y 1 bus, ambos de ancho 1

empieza a decrecer. Esta pérdida es debida al hecho de disponer de un sólo bus (como vimos en los puntos anteriores).

Viendo las características del estudio, nos podríamos preguntar si una arquitectura ancha sería similar a una arquitectura vectorial. Si analizamos ambas arquitecturas, veremos que hay ciertas diferencias entre ellas;

- primero, una arquitectura ancha tal y como la presentamos no aprovecha *strides* diferentes de 1, pues se basa en la contigüidad de los datos en memoria, mientras que una arquitectura vectorial puede sacar partido de *strides* diferentes de 1. Sin embargo, los datos a los que se accede con una instrucción ancha suelen estar en la misma línea de memoria cache, mientras que el sistema de memoria de un procesador vectorial tiene una gran complejidad (y alto coste).

- por otro lado, en una arquitectura ancha se pueden lanzar operaciones simples (de ancho 1) o complejas (de ancho n). Así pues, un bucle que *a priori* fuera no vectorizable pero tuviera alguna operación vectorizable podría ser planificado al completo en una arquitectura ancha. En una arquitectura vectorial se podría hacer *loop distribution* y aprovechar las unidades vectoriales, pero tendríamos varios bucles, mientras que en una arquitectura ancha tendríamos un único bucle con operaciones simples y complejas donde podríamos aplicar segmentación software y aprovechar sus ventajas.

Ilustraremos este último punto con un ejemplo: en la figura 5.6a se muestra un bucle con una recurrencia. En una arquitectura vectorial, el vectorizador podría extraer dos partes vectorizables (P1 y P3) y una no vectorizable (P2, que corresponde a la recurrencia). El procesador vectorial ejecutaría el bucle ejemplo como se muestra en la figura 5.6b: primero ejecutaría P1 de manera vectorial, a continuación P2 sobre las unidades escalares y finalmente P3 de manera vectorial. Lo primero que podemos observar es la cantidad de operaciones de memoria que han debido añadirse al código para hacer el *loop distribution*, con lo que la

cantidad de código a aumentado. Si suponemos que la latencia de las unidades funcionales escalares es de 4 ciclos, la ejecución de P2 necesitaría 8 ciclos por iteración. El tiempo de ejecución sería el número de iteraciones de P2 por 8 ciclos, más el tiempo de la ejecución vectorial de P1 y P3.

Si planificamos el mismo bucle sobre una arquitectura ancha, se aplicarían las técnicas de segmentación software y se solaparía la ejecución de las 3 partes. Así, dado que el bucle es *recurrence-bound*, el *II* es de 8 ciclos por iteración; es decir, el número de iteraciones del bucle (que coincide con el de P2) por 8, más el tiempo del prólogo y el epílogo que, si el número de

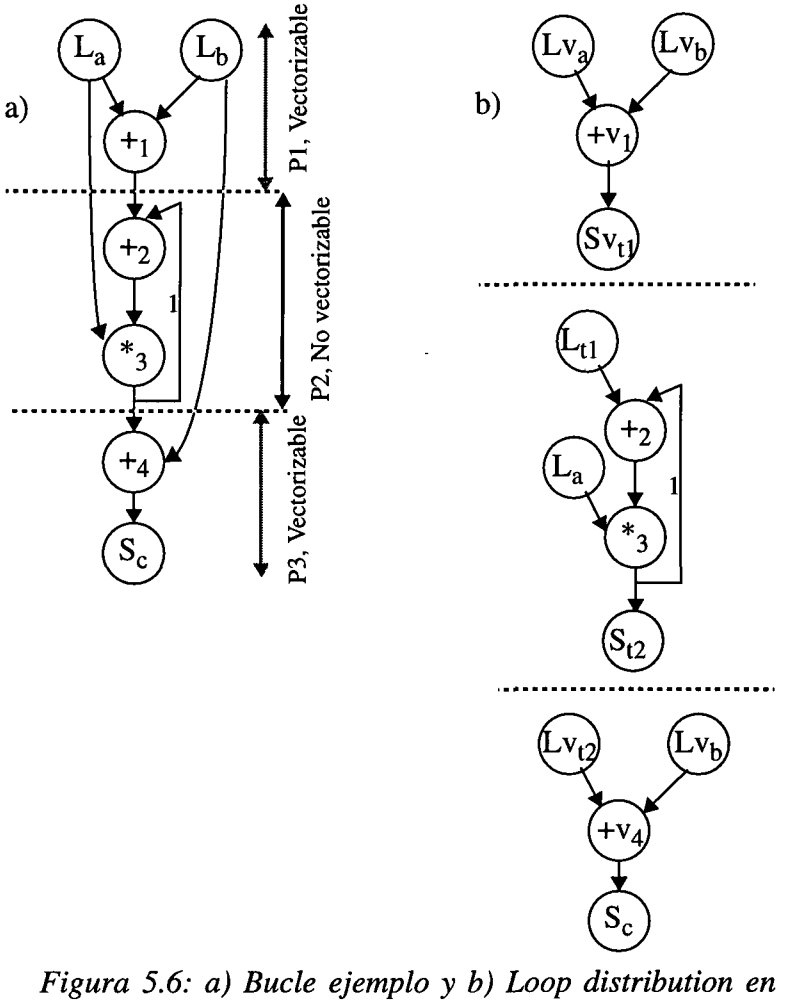

Figura 5.6: a) Bucle ejemplo y b) Loop distribution en un vectorial

iteraciones fuera suficientemente grande, sería despreciable. En este caso, la ejecución de P1 y P3 se solaparía con la ejecución de P2. Podemos ver, por tanto, que la técnica de *widening* puede ofrecer rendimientos para bucles con recurrencias que superen los de procesadores vectoriales.

La figura 5.7 muestra para cada configuración qué porcentaje de los ciclos ganados (respecto a la base) se han dado en bucles sin recurrencias y en bucles con recurrencias. Por ejemplo, en la configuración 1w1 el 75.54% de los ciclos ganados se dan en bucles con recurrencias, y el resto (24.46%) se dan en bucles sin recurrencias. Podemos observar que, conforme aumenta el ancho de las configuraciones, también aumenta el porcentaje de ciclos ganados en bucles sin recurrencias, cosa lógica, ya que estos bucles son los que aprovecharían mejor la técnica de *widening*.

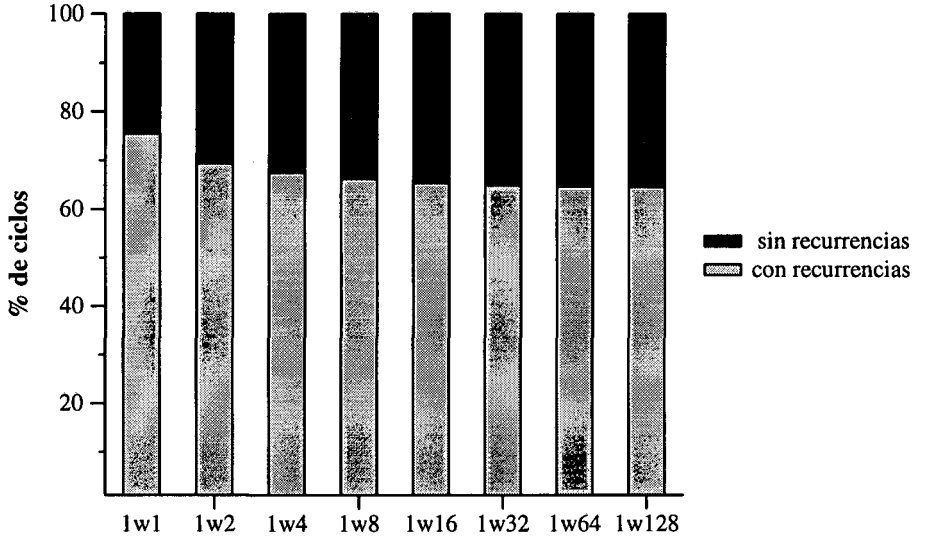

Figura 5.7: Porcentaje de ciclos ganados en bucles con y sin recurrencias para configuraciones donde sólo se aplica la técnica de widening

5.2.4 Técnicas mixtas

Luchemos por cosas lo bastante grandes para que nos importen,
y lo suficiente pequeñas para poder ganarlas.

Jonathan Kozo

En este punto estudiaremos cuál es el rendimiento teórico de diferentes configuraciones donde se aplican diversos grados de replicación y *widening* a los recursos bus y FPUs simultáneamente. Cada configuración **XwY** dispone de **X** buses de ancho **Y**, y 2***X** FPUs de ancho **Y**.

La figura 5.8 muestra los resultados de esta comparación, donde la configuración base tiene 1 FPU y 1 bus, ambos de ancho 1. En dicha figura se han marcado con líneas discontinuas los puntos donde se aplica sólo una de las técnicas. Como era de esperar, el incremento de rendimiento utilizando únicamente replicación es superior al de *widening* en solitario o de

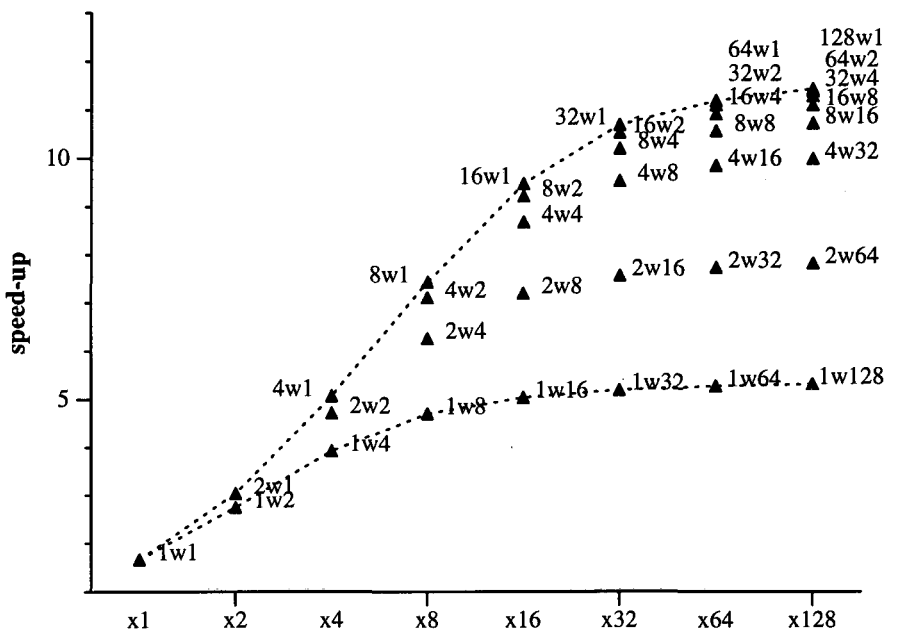

*Figura 5.8: Incremento de rendimiento para configuraciones XwY, donde cada configuración tiene X buses de ancho Y, y 2*X FPUs de ancho Y. Configuración base: 1 FPU + 1 bus, ambos de ancho 1)*

cualquier técnica mixta. Sin embargo, aun usando la técnica de replicación en solitario, el aumento del rendimiento no es proporcional a los recursos disponibles. Esto es debido a que cada vez más bucles se convierten en *recurrence-bound.*

Lo más destacable de esta gráfica es el alto rendimiento de ciertas configuraciones mixtas. Por ejemplo, las configuraciones Xw2 tienen un rendimiento muy similar a las configuraciones (2*X)w1, sin embargo, las segundas tienen el doble de buses y FPUs que las primeras, lo que resultará en un incremento de coste.

5.2.5 Añadiendo unidades funcionales FMA

Mastropiero observó que un 43% de las ovejas emitian el sonido "beeeeee"
y un 57% emitian un balido similar a "meeeeee",
aunque un 10% de las mismas también podían emitir "beeeeeeee".
A estas llamó ovejas ambibalantes, o de balido mixto.

Les Luthiers. Viegésimo aniversario.

Tener una configuración con FPUs capaces de implementar la operación FMA (*Fussed Multiply-and-Add*) permite incrementar el rendimiento de dicha configuración dado que estas FPUs incrementan el rendimiento en bucles *compute-bound* (si hay operaciones aritméticas que puedan fusionar algún par multiplicación-suma asociadas), así como de los bucles *recurrence-bound* (si la operación fusionada formaba parte de la recurrencia que limitaba el rendimiento del bucle).

La figura 5.9 muestra la diferencia de rendimiento entre utilizar FPUs capaces de implementar la operación FMA, y FPUs que no lo hacen. La diferencia de rendimiento cuando se aplica la técnica de replicación en solitario (las dos líneas discontinuas superiores) tiende a reducirse conforme aumenta el número de FPUs. Este efecto se debe a que, conforme aumentamos el número de FPUs de una configuración, el número de bucles *compute-bound* se va reduciendo, de manera que toda la diferencia de rendimiento se va centrando en los bucles *recurrence-bound.*

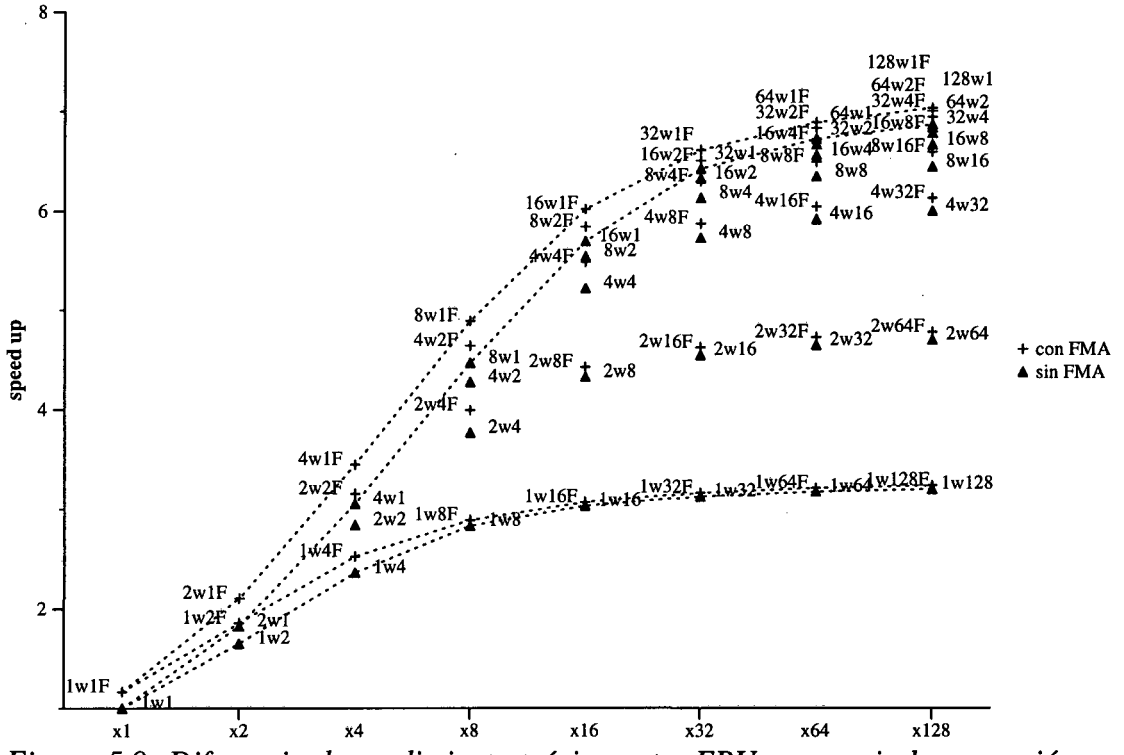

Figura 5.9: Diferencia de rendimiento teórico entre FPUs con y sin la operación fussed multiply-and-add

Respecto a la aplicación de la técnica de *widening* en solitario (las dos líneas discontinuas inferiores), podemos ver como el rendimiento de ambos tipos de FPU tiende a converger al mismo punto rápidamente. Esto se debe a que, si se aplica sólo *widening*, las operaciones no compactables tienen un peso cada vez mayor en la planificación, y dado que podemos ejecutar hasta 4 operaciones aritméticas por cada operación de memoria, estos bucles se irán convirtiendo en *memory-bound.*

El efecto más interesante se produce en las configuraciones intermedias. Algunas configuraciones que usan FMA tienen un rendimiento mejor que configuraciones sin FMA, pero más agresivas. Por ejemplo, la configuración 4w2 con FMA tiene mejor rendimiento que 8w1 sin FMA, siendo esta última más agresiva (tendría el doble de buses y FPUs).

5.3 Añadiendo *spill code*

An affair wants to spill, to share its glory with the world.
No act is so private it does not seek applause.

John Updike

El objetivo de incrementar el número de operaciones que pueden realizarse por ciclo (empleando replicación o *widening*, indistintamente) es reducir el número de ciclos necesarios para ejecutar una iteración de un bucle (*Initiation Interval, II*). Desgraciadamente, reducir el *II* de un bucle puede incrementar el número de registros necesarios en su planificación [LAV98]. Si los registros necesarios para la planificación superan el número de registros físicos disponibles en la arquitectura entonces se debe introducir *spill code* para liberar registros. El *spill code* añadido incrementa el tráfico con memoria y puede dar como resultado un aumento del *II*, con la consiguiente degradación sobre el rendimiento teórico de las configuraciones.

En nuestra propuesta, la técnica de *widening* no sólo se aplica a los buses y a las FPUs, sino también a los registros. Es decir, una configuración XwY (X buses y X*2 FPUs, todo de ancho Y) tiene los registros de un ancho de Y palabras (en nuestras pruebas, palabras de 64 bits). Por ejemplo, una configuración 4w1 con un banco de 32 registros puede acceder a 32 registros de ancho igual a 1 palabra, mientras que una configuración 2w2 con un banco de 32 registros puede acceder a 32 registros de ancho igual a 2 palabras. Si planificamos un bucle en una configuración 2w2, las operaciones compactadas son capaces de almacenar 2 resultados en un único registro de ancho 2, de manera que tenemos una capacidad adicional de almacenamiento respecto a la configuración 4w1. Por otra parte, si las operaciones son no compactables, no podemos beneficiarnos de dicha capacidad de almacenamiento adicional. Este incremento en la capacidad de almacenamiento del banco de registros producida cuando se aplica la técnica de *widening* resulta en un decremento de las necesidades de *spill code* en configuraciones que utilizan dicha técnica.

La figura 5.10 muestra el rendimiento de varias configuraciones cuando se tiene en cuenta el *spill code*. La configuración base con rendimiento 1 es la configuración 1w1 con un

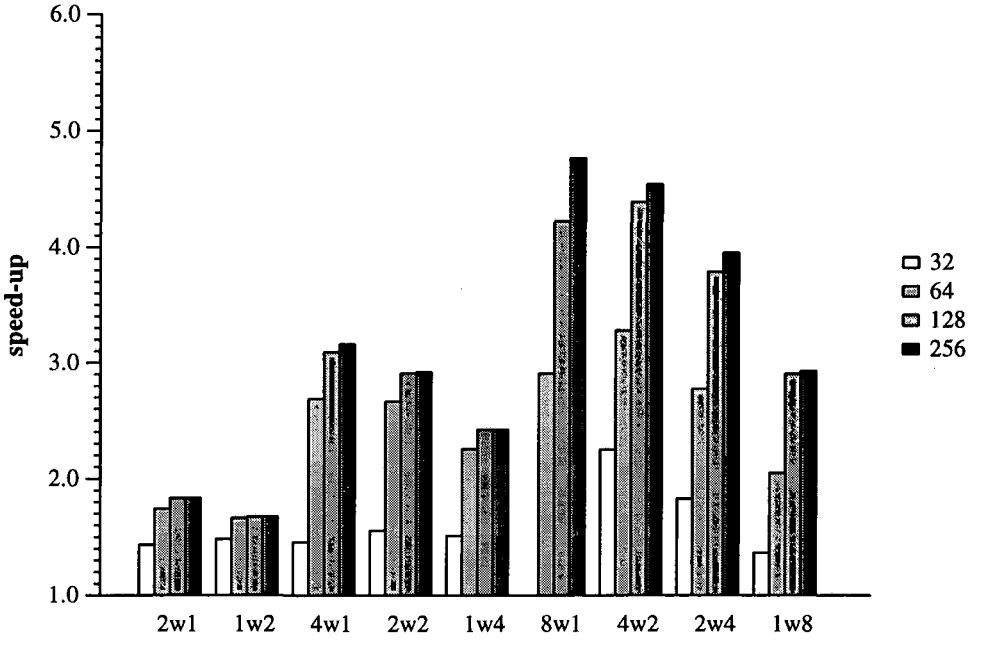

Figura 5.10: Rendimiento de diversas configuraciones XwY teniendo en cuenta el tamaño del banco de registros y el spill code. Configuración base: 1w1 con un banco de 256 registros

banco de 256 registros. Para cada configuración, los bucles han sido planificados usando bancos de 32, 64, 128 y 256 registros y se ha añadido *spill code* cuando era necesario.

Nótese que la configuración 8w1 no tiene la barra que marca el rendimiento de un banco de 32 registros. Esto es debido a que la configuración 8w1 es capaz de producir 24 resultados por ciclo (8 de memoria y 16 de las FPUs), y que consideramos una latencia de 4 ciclos. En este caso, la presión sobre los registros es tan grande que el planificador no es capaz de producir una planificación correcta con los registros disponibles. Esta es una de las razones por la que no se han considerado configuraciones capaces de incrementar el número de operaciones por ciclo de la configuración base por un factor mayor que 8.

Los resultados de esta figura muestran que, cuanto más agresiva es una configuración, mayor cantidad de *spill code* necesita, con la consiguiente degradación de rendimiento. Por ejemplo, la configuración 4w2 tiene un rendimiento relativo a la base de 2.25 con un banco de 32 registros, de 3.28 con 64 registros, de 4.39 con 128 registros y de 4.76 con 256 registros. Sin embargo, la configuración 1w2 consigue prácticamente el máximo rendimiento con un banco de 64 registros.

La capacidad adicional de almacenamiento de la técnica de *widening* reduce la necesidad de *spill code*. Por ejemplo, la configuración 8w1 tiene un rendimiento teórico mayor que 4w2 (véase el punto 5.5); sin embargo, se puede observar que, cuando ambas configuraciones tienen bancos de 64 ó 128 registros, la configuración 4w2 obtiene mejor rendimiento que 8w1, y sólo cuando comparamos las configuraciones con bancos de 256 registros, 8w1 tiene el mejor rendimiento. Así, podemos concluir que la capacidad adicional de almacenamiento de la técnica de *widening* tiene un impacto muy importante en el rendimiento final cuando tenemos en cuenta el tamaño del banco de registros y la necesidad de *spill code*.

La figura 5.11 muestra la misma comparación que la figura 5.10, pero con configuraciones donde las FPUs implementan FMA. Cada columna representa una configuración de máquina y banco de registros, donde la parte de color muestra el rendimiento con FPUs normales y la parte blanca de la columna representa el rendimiento con FPUs que implementan FMA.

Observando los rendimientos de dicha figura, podemos ver cómo, para cada configuración, el hecho de tener un banco de registros limitado tiene un efecto muy similar se apliquen o no FPUS con FMA. Por ejemplo, en la configuración 4w1 con 256 registros se obtiene un rendimiento de 3.16 sin FMA y de 3.50 con FMA. Si comparamos el rendimiento de esta configuración con 32, 64 y 128 registros respecto al rendimiento con 256 registros, observamos que es del 45.8% (32 registros), el 84.8% (64 registros) y el 97.8% (128 registros) con FPUs sin FMA, y del 51.1% (32 registros), 86.2%(64 registros) y 98.3% (128 registros) para FPUs con FMA.

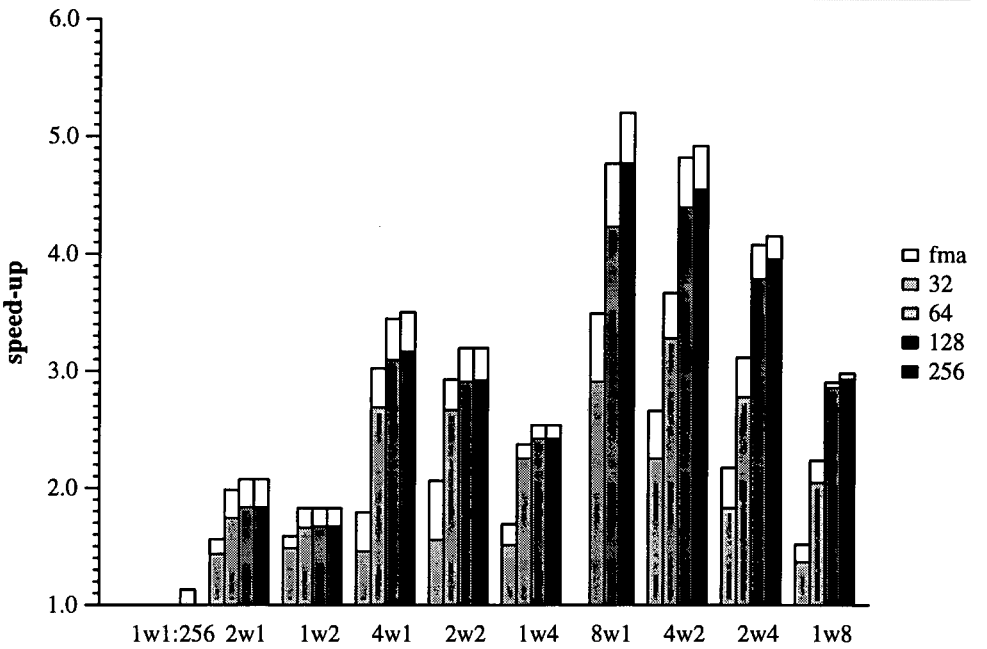

Figura 5.11: Comparativa de rendimiento teniendo en cuenta el spill code para bancos de 32, 64, 128 y 256 registros, para FPUs sin FMA y con FMA. Configuración base: 1w1 con 256 registros

Otro efecto que podemos observar es que, cuanto más agresiva es una configuración, mayor es el rendimiento que se observa cuando se aplica FMA. Por ejemplo, en la configuración 8w1 con 256 registros, el rendimiento es un 9% superior usando FMA que sin usarlo. Este rendimiento baja al 8.1% en la configuración 4w2, al 5% en la 2w4 y al 1.7% en la 1w8. Este efecto se debe a las operaciones no compactables, especialmente las de memoria. Por ejemplo, en la configuración 1w8 disponemos de un sólo bus de ancho 8. Toda operación de memoria no compactable tiene una gran penalización debido a que se deben ejecutar secuencialmente. Por tanto, en una configuración 1w8 muchos de los bucles serán *memory-bound*, por lo que disponer de FPUs con FMA apenas afectará al rendimiento. Este efecto se va suavizando conforme aumentamos el número de buses (configuraciones 2w4, 4w2, 8w1).

Los resultados de las figuras 5.10 y 5.11 nos dan una idea de las diferencias de rendimiento entre diferentes configuraciones. Sin embargo, estas diferencias dependen de varios factores:

- el efecto FMA, que se da entre las configuraciones con FPUs que implementan dicha operación y FPUs que no la implementan. Se puede estudiar comparando el *MII* de dos configuraciones que sólo varíen en el tipo de FPUs de que disponen.

- el efecto planificador, que se debe al hecho de que la planificación se realiza mediante una heurística, que no siempre alcanza el *MII*.

- el efecto *spill*, que se produce cuando los registros requeridos superan los registros disponibles.

A la hora de encontrar el rendimiento final, miramos los ciclos requeridos por la planificación (*II*). Los pasos que se han seguido para disponer de esta información son: primero se calcula el *MII* alcanzable por el par bucle-arquitectura. A continuación se intenta hacer una planificación en los ciclos marcados por el *MII*. Si se fracasa, se incrementa el *II* y se replanifica. Por último, se mira si los requerimientos de registros superan los recursos disponibles. En caso afirmativo, se añade *spill code* y se vuelve a calcular el *MII*.

Las gráficas de la figura 5.12 muestran, para cada configuración, el desglose de los ciclos debidos a cada uno de los factores descritos anteriormente. Las ocho columnas agrupadas de dos en dos indican los ciclos necesarios para la ejecución de todos los bucles del juego de pruebas, para bancos de 32, 64, 128 y 256 registros (leyenda inferior). Para cada banco de registros hay dos columnas, una para indicar los ciclos en la configuración con FPUs capaces de implementar la operación *Fussed Multiply-and-Add* (columna con la leyenda FMA) y la otra para indicar los ciclos de FPUs que no la implementan (columna sin leyenda). A su vez, cada columna indica qué parte de los ciclos es debida a la combinación bucle-arquitectura (parte gris, *MII*), cuál es debida al factor planificador (parte negra) y cuál al *spill code* añadido (parte blanca).

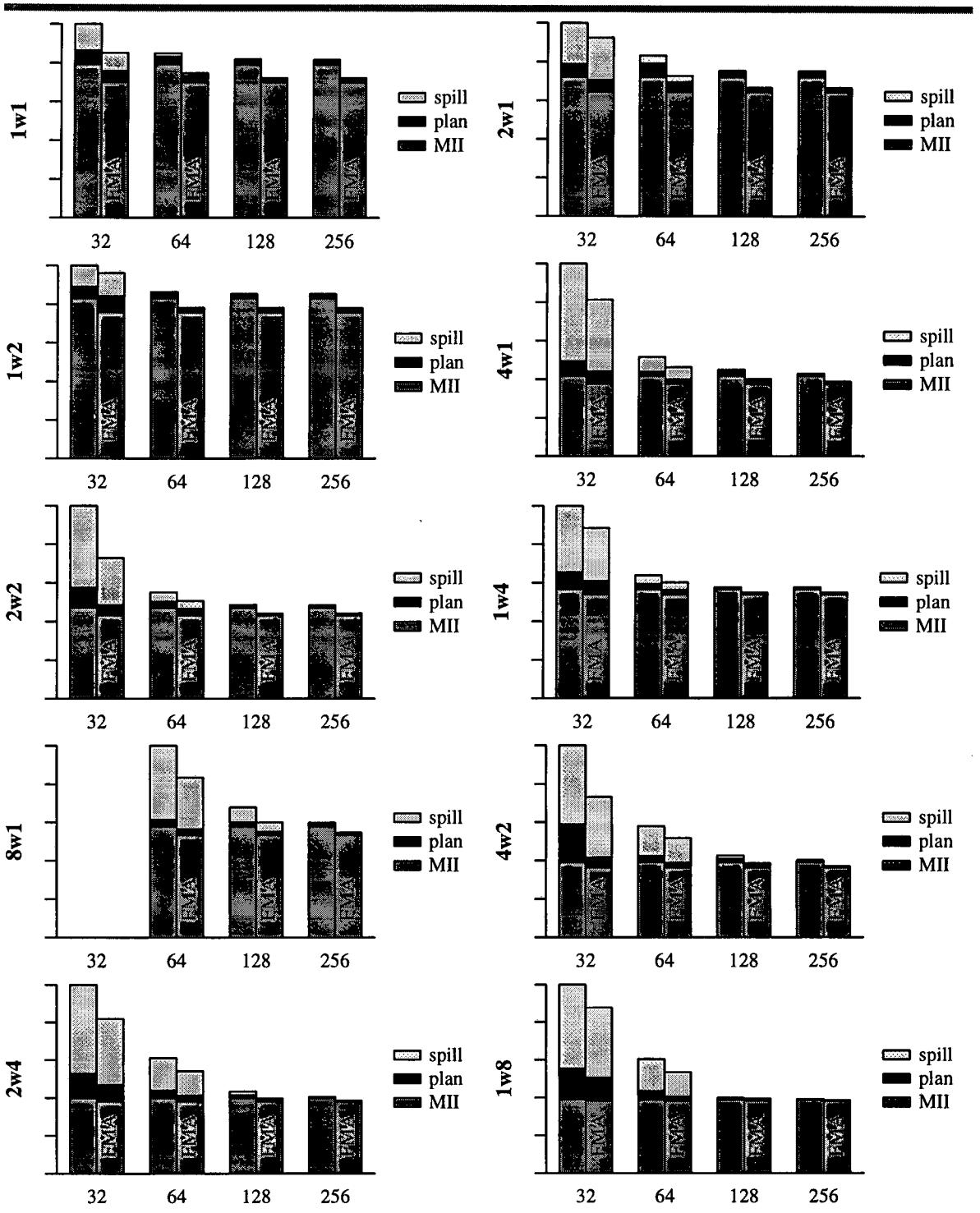

Figura 5.12: Desglose de ciclos empleados para cada configuración entre MII, factor planificador y spill code

La parte gris (*MII*) de una configuración difiere de usar FPUs con FMA a no usarlas, pero es independiente del tamaño del banco de registros, ya que es el número de ciclos mínimo teórico, antes de la planificación y la asignación de registros. Por ejemplo, en la configuración 1w1, usar FMA puede ahorrarnos alrededor de un 12% de los ciclos teóricos.

La parte negra (el factor planificador) es el número de ciclos que ha introducido el planificador sobre el teórico, antes de encontrar una planificación correcta. Estos ciclos son debidos a la heurística empleada y serían diferentes con otra heurística. Se puede observar que dicho factor suele ser bastante pequeño respecto al total (por ejemplo, en la configuración 1w1 con 256 registros y FMA es de menos de un 2%).

La parte blanca (el factor *spill*) crece conforme disminuye el número de registros disponible. En esta parte hemos puesto sólo los ciclos debidos al *spill*, es decir, la diferencia entre el *MII* teórico inicial y el *MII* teórico una vez añadidas las operaciones de memoria que conforman el *spill code*. Podemos observar que, para la configuración 1w1, con 128 y 256 registros no es necesario incluir este tipo de código adicional. Con 64 registros, hay que añadir, pero muy poco, y ya con 32 registros el coste en ciclos es considerable.

Analizaremos a continuación algunos efectos observados en estas gráficas.

Como era de esperar, cuanto menor es el banco de registros, más ciclos se pierden debidos al *spill code*. Sin embargo, podemos observar cómo ninguna de las configuraciones necesita más de 256 registros, y que sólo unas pocas necesitan más de 128 registros (de hecho, la única que tiene un número de ciclos significativo en *spill code* con 128 registros es la configuración 8w1, que es la más agresiva). El uso de bancos de 32 registros en configuraciones agresivas da los malos resultados esperables, pudiendo en algunas ocasiones necesitar el doble de ciclos del mínimo teórico (como en las configuraciones 4w1, 2w2, 4w2, 2w4 y 1w8). En la configuración 8w1, la presión sobre el planificador es tan grande, que ni siquiera puede obtener una planificación correcta.

Respecto al planificador, podemos observar cómo hay una relación entre el número de ciclos añadido por el mismo y la dificultad de la planificación. Dicha dificultad depende del número de nodos añadidos como *spill code*: dado que este código se añade para reducir la presión sobre los registros, se fuerza que las nuevas instrucciones de store y load añadidas se planifiquen lo suficientemente alejadas para maximizar el tiempo en que este dato esté en memoria en lugar de estar usando un registro. Esto explica por qué el número de ciclos debidos al planificador suele ser insignificante con configuraciones con 256 registros (que no requieren *spill code*), pero crece conforme el número de registros disponible se va haciendo más pequeño. Esto también explica una ventaja de las unidades funcionales con FMA: al reducir el número de nodos (dos nodos multiplicación y suma se convierten en uno sólo) se reduce el número de ciclos ocupados y se facilita la labor del planificador cuando hay *spill code* (además, utilizando este tipo de unidades aritméticas, se produce menos *spill code*)

Si miramos la diferencia entre dos configuraciones con las mismas características, pero una con FPUs que implementan la operación FMA y otra con FPUs que no la implementan, podemos ver que también hay una diferencia de ciclos. Una parte de estos ciclos será debida a mejoras en las recurrencias (cuando fusionamos dos operaciones que están dentro de la recurrencia) o debida a los recursos (una operación FMA ejecuta dos operaciones ocupando un sólo recurso). La figura 5.13 muestra qué parte de los ciclos ganados pertenecen a cada uno de estos casos, para cada configuración. En la configuración 1w1, casi todo el incremento de rendimiento se debe a los recursos. Esto es lógico, ya que dicha configuración es muy poco agresiva, de manera que la mayoría de los bucles estarán limitados por los recursos. Sin embargo, conforme aumenta el número de recursos, cada vez habrá más bucles limitados por las recurrencias. Así, en las configuraciones capaces de lanzar 8 veces más operaciones que la configuración base (configuraciones 8w1, 4w2, 2w4 y 1w8) el porcentaje de ciclos ganados por recurrencias es mayor que en las configuraciones capaces de lanzar 4 veces más operaciones (configuraciones 4w1, 2w2 y 1w4).

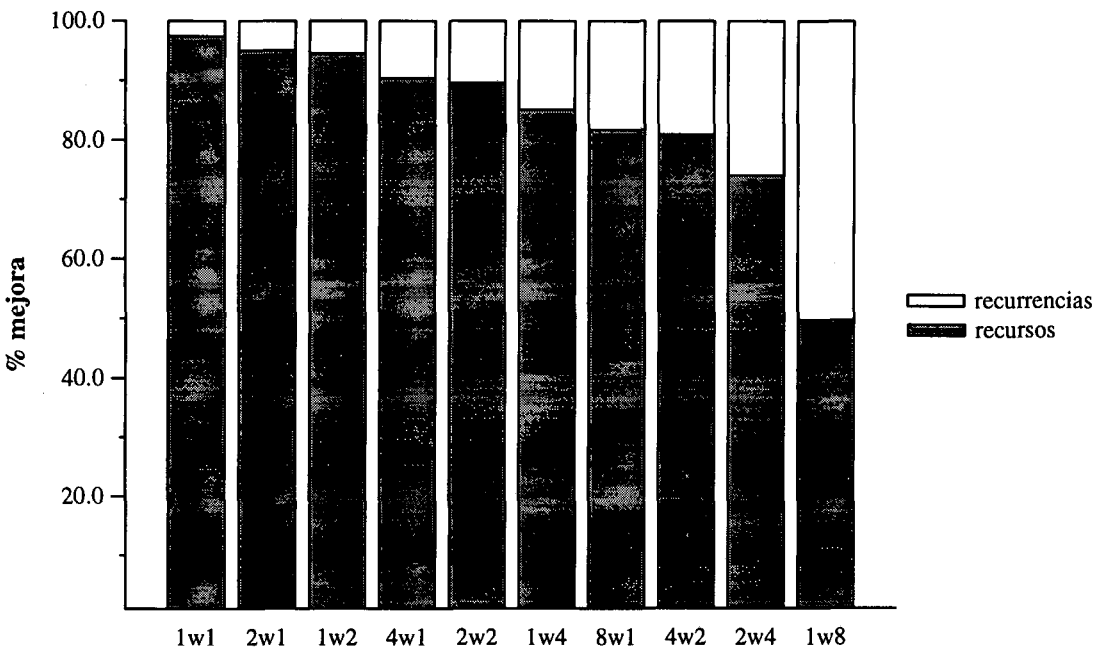

Figura 5.13: Desglose del incremento de rendimiento debido a unidades funcionales con FMA

Como resumen de este punto, diremos que, del estudio del rendimiento cuando se efectúa una planificación con un número de registros limitado, se pueden extraer las siguientes conclusiones:

- la técnica de *widening* obtiene una ventaja de su capacidad adicional de almacenamiento en el banco de registros, lo que provoca que algunas configuraciones muy agresivas tengan un rendimiento real (en ciclos) menor que otras configuraciones menos agresivas y con un rendimiento teórico menor. Por ejemplo, la configuración 1w2 con 32 registros obtiene un mejor rendimiento una vez planificado que 2w1 con 32 registros, siendo esta última la que tenía mejor rendimiento teórico.

- dados los rendimientos en ciclos de diferentes combinaciones, se puede ver la necesidad de un estudio detallado del coste de una configuración donde se aplican diferentes grados de replicación y de *widening*. Por ejemplo: la configuración 4w2 tiene

mayor rendimiento teórico y mayor coste que la 2w4. Sin embargo, 2w4 con un banco de 128 registros consigue un incremento de rendimiento 2 veces superior al de la configuración 4w2 con 32 registros y un incremento un 20% superior al de 4w2 con 64 registros. Es por ello que hemos realizado un estudio rendimiento-coste hardware, que se explicará en los siguientes capítulos.

- las ventajas de usar FPUs capaces de implementar la operación FMA ofrece un incremento de rendimiento debido a tres factores: reducción del *MII*, debido tanto a recurrencias como a recursos, reducción de la necesidad de *spill code*, y reducción del número de nodos a planificar, lo que facilita la labor del planificador.

5.4 Sumario y contribuciones

En este capítulo se ha presentado un estudio de los factores que limitan el rendimiento de la técnica de *widening*. Para ello se ha estudiado el efecto sobre el rendimiento teórico de dicha técnica en buses, en FPUs y en ambos recursos simultáneamente, aplicada en solitario o en combinación con la técnica de replicación. También se ha estudiado el efecto de las técnicas sobre el rendimiento cuando se efectúa la planificación y se limita el tamaño del banco de registros. Todos los estudios se han realizado para FPUs capaces de implementar la operación *Fussed Multiply-and-Add* y para FPUs que no la implementaban.

De estos estudios [LLVA98b] (que configuran la contribución de este capítulo) se puede extraer una conclusión: el estudio del ILP teórico no indica cuál es la mejor configuración. Hemos visto casos en que, entre dos configuraciones donde la primera tiene un rendimiento teórico superior a la segunda, es la segunda la que ofrece el mejor rendimiento cuando se realiza una planificación con un banco de registros limitado.

De esta conclusión nace la necesidad de evaluar el coste de las propuestas (veremos cómo hacerlo en el siguiente capítulo) y efectuar un estudio del rendimiento teniendo en cuenta dichos costes (que veremos en el capítulo 7).