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 6.
Coste de una arquitectura

6.1 Introducción

En el capítulo anterior analizamos el rendimiento de las técnicas de replicación y *widening*. Vimos cómo podía alcanzarse un cierto rendimiento máximo teórico usando sólo replicación, sólo *widening* o una mezcla de ambas técnicas. Asimismo se vio cómo entre arquitecturas con el mismo rendimiento máximo teórico, aquella que aplicaba solamente la técnica de replicación producía el mejor rendimiento (bajo ciertas condiciones ideales: número infinito de registros, memoria perfecta). Sin embargo, un efecto que se podía observar era que con una arquitectura donde se mezclaran las técnicas de replicación y *widening* se podía obtener un rendimiento muy cercano al de una arquitectura donde se aplicara sólo la técnica de replicación. Estos rendimientos también variaban cuando se limitaba el número de registros físicos disponibles y surgía la necesidad de *spill code*. Todas estas consideraciones nos llevan a la necesidad de estudiar el coste de las técnicas propuestas. En este capítulo veremos cómo calcular el coste de una arquitectura, a nivel de área del procesador (FPUs y banco de registros), tiempo de acceso al banco de registros, memoria y lenguaje máquina.

6.2 Coste en área

Joy, has no cost.

Marianne Williamson

Previsiones de la tecnología

La capacidad de los chips aumenta año tras año; cada vez se pueden introducir más transistores por milímetro cuadrado, y de hacer los *chips* más grandes en área. En la tabla 6.1 se puede observar las previsiones de λ y tamaño del *chip* de las próximas generaciones, según un estudio de la *Semiconductor Industry Association* [SIA94].

	1998	2001	2004	2007	2010
λ (μm)	0.25	0.18	0.13	0.10	0.07
Tamaño (mm^2)	300	360	430	520	620
λ^2 por chip ($x10^6$)	4800	11111	25443	52000	126530
λ^2 / mm^2 ($x10^6$)	16	30.86	59.17	100	204.08

Tabla 6.1: Predicciones de la *Semiconductor Industry Association* (SIA) en 1994

En la tabla 6.1 se puede observar que las previsiones de λ para el año 1998 son de 0.25 micras (μm) y un tamaño de chip de 300 mm^2, mientras que para el año 2001 se prevé una λ de 0.18 micras y un tamaño de 360 mm^2, de manera que la capacidad de λ^2 por *chip* se ha multiplicado por 2.3 entre estas dos generaciones. Así pues, nuestro estudio presenta el coste en área en función de la cantidad de transistores (o λ^2) requerido por cada elemento de la CPU estudiado, de manera que es independiente de la tecnología usada.

Área de una unidad funcional multifunción

Hemos calculado el área de una unidad funcional multifunción usando como referencia la FPU de MIPS R10000. Dicha FPU incluye un multiplicador, un sumador y un divisor, que son los elementos básicos que consideramos en una unidad funcional multifunción. Con una

tecnología de 0.25 μm, la FPU del R10000 requiere 12 mm^2 de área [ONH+96], así que asumimos el área de una FPU como 12 mm^2 x $16x10^6$ λ^2/mm^2 = $192x10^6$ λ^2.

Área del banco de registros

El área de un banco de registros viene determinada principalmente por el tamaño de cada una de las celdas de memoria, que son la parte más replicada del banco de registros. Los otros componentes necesarios para acceder al banco de registros, como los decodificadores, suelen representar menos del 5% del área total requerida [Lee92]. Así pues nos centraremos en el estudio del coste necesario para cada celda.

La figura 6.1 muestra una celda de memoria de un banco de registros con un puerto de lectura y uno de escritura. La celda ha sido implementada usando CMOS escalable [Lee92]. Para acceder a la celda de memoria, cada puerto requiere un transistor, una línea de selección y una línea de datos. Cada puerto de lectura requiere, además, un segundo transistor de acceso y una segunda línea de datos. El área de la celda de un registro crece aproximadamente como el cuadrado del número de puerto añadidos, dado que cada puerto fuerza el crecimiento de la celda tanto a lo ancho como a lo alto.

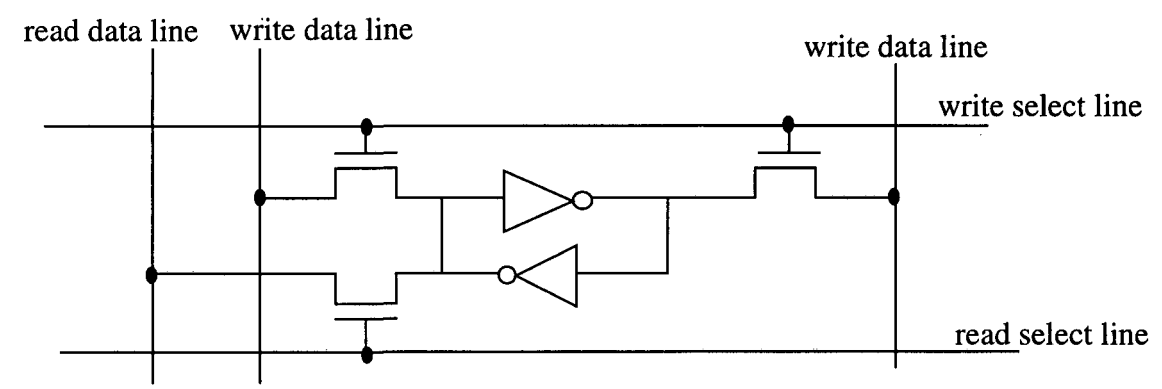

Figura 6.1: Esquema de una celda de memoria de un banco de registros con un puerto de lectura y un puerto de escritura, usando CMOS escalable

La parte de memoria de la celda consiste en un par de inversores cruzados (*cross-coupled inverters*) que tienen 4 transistores y que fuerzan una altura mínima de 41λ. La celda puede acomodar 3 líneas de selección que crucen a lo ancho de la celda. Así, la altura de la celda no crece hasta que se dispone de más de 3 puertos. Cada puerto adicional añade 8 λ a la altura de la celda.

El ancho de la celda descrita es de 50λ. Cada puerto de lectura adicional añade 14λ a la anchura: 8λ por la línea de datos y 6λ por el transistor de acceso. Cada puerto de escritura adicional añade 28λ a la anchura, porque necesita 2 líneas y 2 transistores. La tabla 6.2 muestra las dimensiones de varias celdas de registros multipuerto, con el área relativa a una celda base con 1 puerto de lectura y 1 de escritura (1R, 1W).

Puertos	1R, 1W	2R, 1W	4R, 2W	5R, 3W	10R, 6W	20R, 12W
ancho x alto	50x41	64x41	120x65	162x81	316x145	568x257
Área (λ^2)	2050	2624	7800	13122	45820	145976
Área relativa	1	1.28	3.80	6.4	22.35	71.21

Tabla 6.2: Comparación de las dimensiones de varias celdas de bancos de registros multipuerto.

Coste de las técnicas de replicación y *widening*

Para comparar el coste de ambas técnicas, tomemos una configuración básica con 1 bus y 2 FPUs, que llamaremos configuración 1w1 (grado de replicación 1 y grado de *widening* 1).

Los requerimientos del banco de registros de la configuración 1w1 son los siguientes: el bus con la memoria cache requiere 1 puerto de lectura y 1 puerto de escritura. Cada FPU requiere 2 puertos de lectura y 1 de escritura. Por tanto 1w1 necesita 5 puertos de lectura y 3 de escritura. En la tabla 6.2 se observa que este número de puertos necesita 13122 λ^2 por bit. Si asumimos un banco de 64 registros de 64 bits cada uno, la configuración 1w1 necesita

13122x64x64 = 53.75x$10^6\lambda^2$ para el banco de registros, más 192x10^6x2 = 384x$10^6\lambda^2$ para las 2 FPUs.

La configuración 1w1 puede ejecutar 2 instrucciones aritméticas y 1 de memoria por ciclo. Podemos doblar la capacidad teórica aplicando replicación (configuración 2w1) o *widening* (configuración 1w2). En ambos casos, el área de las FPUs es el doble que en la configuración 1w1 (es decir, 768x$10^6\lambda^2$). La diferencia de área entre las configuraciones 1w2 y 2w1 radica en el banco de registros.

Si aplicamos replicación necesitaremos un banco de 64 registros de 64 bits con 10 puertos de lectura y 6 de escritura por celda de memoria (ya que tenemos 2 buses y 4 FPUs). En la tabla 6.2 podemos ver que este banco de registros necesita 45820λ^2 por bit. Por tanto el coste es de 45820x64x64 = 187.7x$10^6\lambda^2$ (3.49 veces el área del banco de registros de la configuración 1w1).

Si aplicamos *widening* necesitaremos un banco de 64 registros de 128 bits (la técnica *widening* se aplica también a los registros) con 5 puertos de lectura y 3 de escritura. El banco resultante tiene un área de 13122x64x128 = 107.5x$10^6\lambda^2$; es decir, 2 veces el área del banco de la configuración 1w1 y un 57.3% del área de la configuración 2w1 (además 1w2 tiene el doble de capacidad de almacenamiento que 2w1).

En el ejemplo anterior hemos considerado configuraciones no muy agresivas, donde el factor dominante del área eran las FPUs y no el banco de registros. Considerando ambos, la configuración 1w1 tiene un área de 474x$10^6\lambda^2$, 2w1 tiene un área de 965x$10^6\lambda^2$ y 1w2 un área de 875x$10^6\lambda^2$ (es decir, el 91.5% del área de 2w1). Sin embargo, en configuraciones más agresivas en términos de número de unidades funcionales, la parte dominante del área será la perteneciente al banco de registros ya que crece cuadráticamente respecto al número de puertos. Por ejemplo, la configuración 16w1 (grado de replicación 16, grado de *widening* 1) con un banco de 256 registros de 64 bits, necesita 48306x$10^6\lambda^2$, mientras que la configuración

1w16 (grado de replicación 1, grado de *widening* 16) con un banco de 256 registros de 1024 bits (16 veces 64 bits) necesita sólo $9584x10^6\lambda^2$. En otras palabras, la configuración 1w16 tiene el mismo número de FPUs que la configuración 16w1 (y por tanto la misma área, $3072x10^6\lambda^2$), pero 1w16 tiene 16 veces más capacidad de almacenamiento ya que cada registro es de ancho 16 palabras, y requiere sólo el 19.8% del área del banco de registros de la configuración 16w1.

La figura 6.2 muestra el coste total en área de las técnicas de replicación y *widening* para bancos de 32, 64, 128 y 256 registros. El eje horizontal representa el ancho de memoria **b** para una configuración XwY (**b**=X*Y). El ancho de los registros varía con el ancho de las unidades funcionales, por tanto la capacidad de almacenamiento es proporcional al grado de *widening*.

La línea inferior (marcada con diamantes) representa el área debida a las FPUs. La línea continua marca la aplicación de la técnica de *widening* en solitario: todas las configuraciones tienen 1 bus y 2 FPUs, aunque varía su anchura (configuraciones 1wb). Nótese que el área crece linealmente respecto al grado de *widening*. La línea superior (de puntos, marcada con triángulos) presenta el coste en área de la técnica de replicación en solitario: todas las configuraciones tienen un ancho de 1 (configuraciones bw1). Los puntos intermedios entre las dos líneas anteriores reflejan el impacto de aplicar ambas técnicas. Se puede observar que cuanto más agresiva es la configuración (más unidades funcionales y registros) y se aplica replicación, mayor es el impacto del banco de registros en el área total (nótese que el eje vertical está en escala logarítmica).

Asumimos que es razonable dedicar entre el 10% y el 20% del área total del chip a las FPUs y el banco de registros en coma flotante. Las bandas horizontales de la figura 6.2 representan la cuña entre el 10 y el 20% del total del área del chip para cada una de las cinco generaciones de las predicciones de la SIA mostradas en la tabla 6.1. Estas bandas representan qué puede ser implementado bajo unas determinadas restricciones tecnológicas. Por ejemplo, en el año 2007 (previsión de tecnología de 0.10μ) será posible implementar las configuraciones 2w8 y 8w1 con un banco de 128 registros si estamos dispuestos a dedicarle

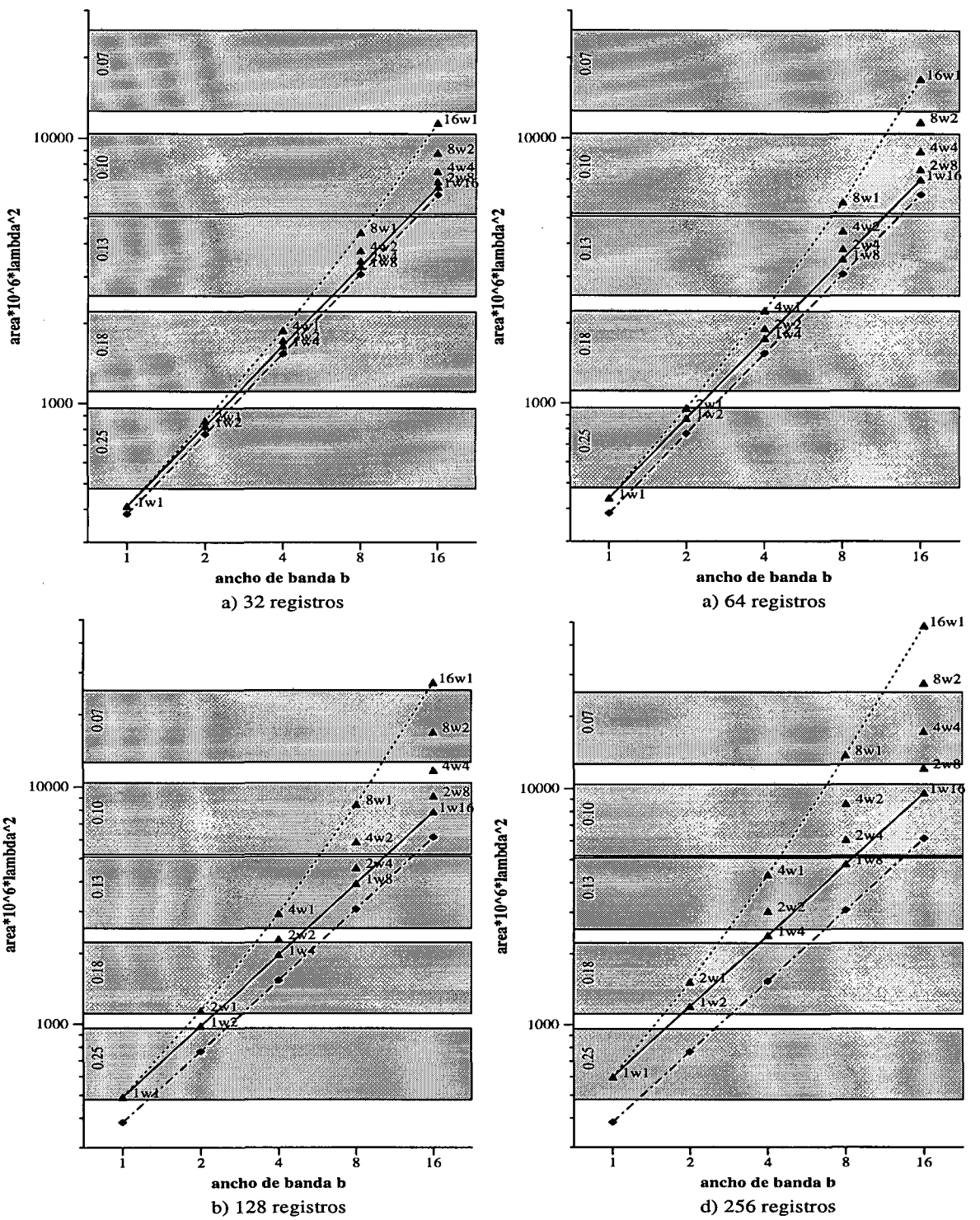

Figura 6.2: Tamaño del banco de registros y las FPUs para diversas configuraciones xwy, *asumiendo bancos de a) 32 b) 64 c) 128 y d) 256 registros*

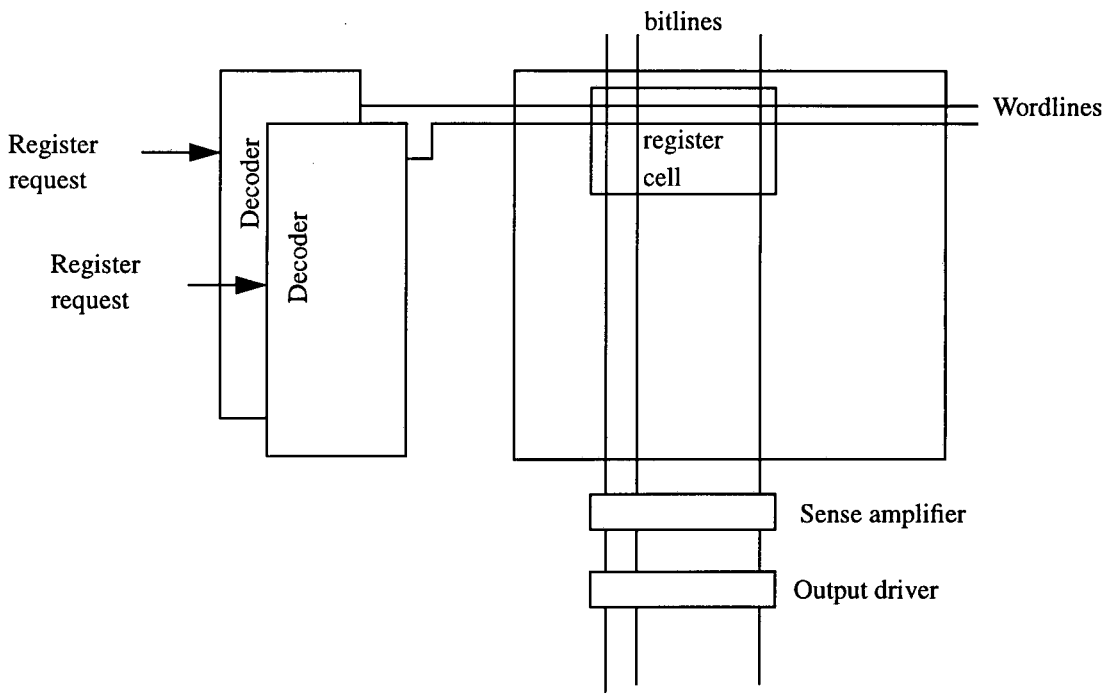

Figura 6.3: Estructura de un banco de registros multipuerto

casi el 20% del área total del *chip*, mientras que si dedicamos el 10%, será posible implementar las configuraciones 4w1 y 2w4, pero no 2w8 y 8w1.

6.3 Tiempo de ciclo

El tiempo es relativo.

Albert Einstein

Un banco de registros multipuerto sigue el esquema propuesto en la figura 6.3. El modelo de tiempo de acceso utilizado está basado en una adaptación del modelo de tiempo de acceso a un banco de registros usado por Keith Farkas en su tesis, que a su vez está basado en el modelo CACTI de memoria cache propuesto por Wilton y Jouppi.

El modelo CACTI [WiJo96] de memoria cache se basa en ecuaciones relativamente simples que permiten modelizar el tiempo de acceso y ciclo en función de varios parámetros de la cache. El modelo *register file cycle time* (RFCT) de Jouppi usado y descrito por Farkas en su tesis [Far97] es una adaptación de CACTI donde se inhiben ciertas partes del modelo y se modifican algunos parámetros.

A diferencia de una memoria cache, un banco de registros no necesita una tabla de *tags* ya que existe una correspondencia uno a uno entre el identificador de un registro y su localización en memoria. Así, la tabla de *tags* de CACTI es inhibida y sólo se usa la tabla de datos. El modelo RFCT asume una organización en que dos registros no pueden compartir la misma línea de activación (*wordline*). Por tanto, se considera un banco de registros como una matriz de datos con tantas columnas como bits tenga el registro y tantas filas como registros tenga el banco.

El modelo CACTI no es multipuerto, sin embargo un banco de registros requiere un número de puertos mínimo para trabajar (por ejemplo, es necesario que el banco de registros pueda ser leído y escrito simultáneamente). Para calcular el tiempo de un banco multipuerto, RFCT escala las capacitancias en las direcciones del *bitline* y el *wordline* debido al metal añadido para cablear las nuevas líneas necesarias por el incremento de puertos. Este escalamiento es directamente proporcional al número de puertos, ya que RFCT considera que las celdas crecen proporcionalmente al número de puertos.

En nuestro trabajo se modificó el modelo CACTI siguiendo los parámetros descritos en los dos últimos párrafos, excepto por lo que respecta al escalamiento de la capacitancia del *wordline* y del *bitline* ya que, como vimos en el punto anterior, la celda no crece proporcionalmente respecto al número de puertos. En nuestro caso, el escalamiento ha sido proporcional al incremento en anchura (para el *wordline*) y en altura (para el *bitline*) de cada celda, lo que ofrece unos resultados mucho más realistas.

El modelo se asume gobernado por el tiempo de lectura, y se puede expresar como la suma de los siguientes términos:

- Tiempo de decodificación: es el tiempo necesario para decodificar a qué registro se quiere acceder y activar su línea de selección de registro (*wordline*). Este tiempo depende principalmente del número de registros disponible.

- Tiempo de línea de selección (*wordline*): es el tiempo necesario para que se activen todas las celdas del registro por medio de la señal que pasa por el *wordline*. Depende principalmente del número de bits del registro y del ancho de cada celda de memoria.

- Tiempo de línea de datos (*bitline*): es el tiempo que transcurre entre la activación del *wordline* y la detección del estado de la celda de memoria por parte del amplificador de señal. Depende principalmente de la altura de cada celda y del número de registros.

- Tiempo de amplificación de señal: el tiempo necesario para que la señal atraviese el amplificador.

- Tiempo de volcado de señal: tiempo necesario para cargar el dato leído en el bus interno que conecta el banco de registros con la ALU.

- Tiempo de precarga: el tiempo necesario para la precarga de las líneas de datos, los comparadores y el bus interno de decodificación.

En resumen, el tiempo de acceso está gobernado por el número de registros, el ancho de los mismos y el tamaño de la celda de memoria (que depende del número de puertos). La tabla 6.3 muestra el tiempo de acceso de diferentes configuraciones variando el número de registros (32, 64, 128 y 256), el ancho de cada registro (64 bits x el grado de *widening*) y el tamaño de la celda de memoria (que depende del número de puertos). Para considerar el tiempo de manera independiente de la tecnología usada, todos los tiempos han sido normalizados respecto a una configuración base (1w1 con un banco de 32 registros).

Configuración	Tamaño del banco de registros			
	32	64	128	256
1w1	1	1.05	1.18	1.34
2w1	1.49	1.54	1.70	1.87
1w2	1.10	1.15	1.29	1.45
4w1	2.44	2.51	2.69	2.90
2w2	1.65	1.72	1.87	2.06
1w4	1.22	1.27	1.43	1.60
8w1	4.32	4.41	4.61	4.87
4w2	2.75	2.82	3.00	3.23
2w4	1.85	1.92	2.09	2.29
1w8	1.39	1.45	1.62	1.80
16w1	8.04	8.15	8.39	8.72
8w2	4.89	4.99	5.20	5.48
4w4	3.10	3.18	3.38	3.61
2w8	2.12	2.20	2.38	2.60
1w16	1.68	1.75	1.93	2.14

Tabla 6.3: Tiempo de acceso relativo de varios bancos de registros (1w1 con 32 registros, tiempo de acceso = 1)

Para reducir el tiempo de acceso, un banco de registros puede ser particionado en diversos bancos, manteniendo copias de todos los datos [CDN92]. Por ejemplo, el banco de registros de la configuración 8w1 puede ser implementado como un único banco donde cada celda necesita 40 puertos de lectura y 24 de escritura (8R + 8W para los 8 buses y 32R + 16W para las 16 FPUs). El mismo banco puede ser implementado como dos copias idénticas, donde todas las unidades funcionales pueden escribir en ambos bancos, pero sólo 4 buses y 8 FPUs leen de cada una de las dos copias. De esta manera, cada celda requiere 20 puertos de lectura y 24 de escritura, lo que significa un incremento del área del banco, pero también un decremento de su tiempo de acceso.

La configuración 8w1 puede ser particionada en 1, 2, 4 u 8 bloques, resultando en el incremento relativo de área y decremento relativo de tiempo de acceso presentado en la figura 6.4.

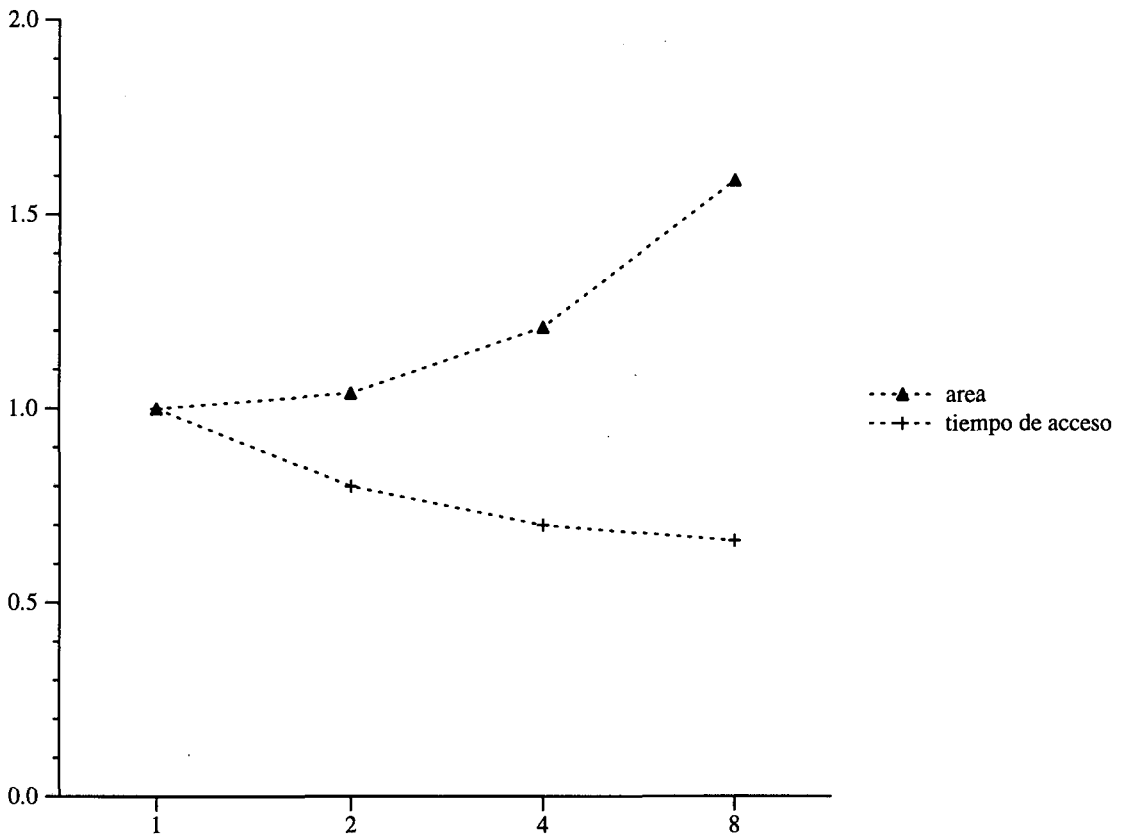

Figura 6.4: Comportamiento del área y tiempo de ciclo de un banco de 64 registros para una configuración 8w1, con particionado 1, 2, 4 y 8.

En la figura podemos observar cómo el incremento en área crece de manera exponencial, mientras que el decremento en tiempo de ciclo disminuye de manera logarítmica. Un particionado pequeño, como por ejemplo partir el banco de registros en 2 bancos, tiene un pequeño incremento en coste y un decremento en tiempo de acceso muy importante. La tabla 6.4 muestra cómo varía el tiempo de ciclo con el particionado del banco de registros, mientras que la tabla 6.5 muestra la variación en área. El compromiso del particionado será utilizado en el capítulo 7 para el estudio de las configuraciones que ofrecen mejor rendimiento/coste.

Configuración	Particio-nado	Banco de registros			
		32	64	128	256
1w1	1	1	1.0457	1.18137	1.33777
2w1	1	1.48847	1.54401	1.69512	1.87494
	2	1.26781	1.31933	1.46408	1.63422
1w2	1	1.09762	1.14929	1.28992	1.45131
4w1	1	2.44313	2.51236	2.68528	2.89844
	2	2.01084	2.07458	2.23864	2.43813
	4	1.79353	1.85416	2.01326	2.20513
2w2	1	1.65493	1.71714	1.87432	2.06079
	2	1.40308	1.46098	1.61135	1.78745
1w4	1	1.219	1.27434	1.43005	1.59554
8w1	1	4.32185	4.40941	4.6123	4.87204
	2	3.46927	3.54973	3.7407	3.98159
	4	3.04152	3.11788	3.30207	3.53237
	8	2.82716	2.90129	3.08184	3.30646
4w2	1	2.7457	2.82256	3.0031	3.22528
	2	2.25166	2.32265	2.4937	2.70124
	4	2.00339	2.07106	2.23679	2.43617
2w4	1	1.854	1.9206	2.09402	2.28631
	2	1.56694	1.62894	1.79507	1.97629
1w8	1	1.38762	1.44771	1.61556	1.80409
16w1	1	8.03965	8.1504	8.39283	8.71594
	2	6.43897	6.54194	6.77031	7.06996
	4	5.50232	5.59962	5.81816	6.10193
	8	5.07834	5.17287	5.38662	5.66268
	16	4.86619	4.95925	5.1705	5.44254
8w2	1	4.89442	4.99093	5.20357	5.47565
	2	3.91921	4.0081	4.20798	4.45986
	4	3.43004	3.51454	3.70716	3.94769
	8	3.18493	3.26705	3.45576	3.69022
4w4	1	3.09841	3.18068	3.37911	3.6096
	2	2.53465	2.61064	2.79891	3.01374
	4	2.25144	2.32388	2.50645	2.71255
2w8	1	2.12065	2.19284	2.37979	2.59715
	2	1.78916	1.85641	2.03551	2.24094
1w16	1	1.68225	1.74965	1.93444	2.14082

Tabla 6.4: Variación del tiempo de ciclo del banco de registros debida al particionado

Configuración	Particio-nado	Banco de registros			
		32	64	128	256
1w1	1	410	437	491	598
2w1	1	861	955	1143	1518
	2	873	979	1191	1614
1w2	1	821	875	982	1197
4w1	1	1884	2233	2931	4327
	2	1918	2301	3066	4596
	4	2054	2573	3611	5687
2w2	1	1723	1911	2286	3037
	2	1747	1959	2382	3228
1w4	1	1643	1750	1965	2395
8w1	1	4415	5758	8445	13819
	2	4522	5973	8875	14679
	4	5013	6954	10837	18602
	8	6131	9191	15310	27549
4w2	1	3769	4467	5863	8654
	2	3837	4602	6132	9193
	4	4109	5147	7223	11374
2w4	1	3447	3822	4573	6074
	2	3495	3918	4764	6457
1w8	1	3286	3501	3931	4791
16w1	1	11414	16684	27224	48305
	2	11879	17614	29084	52025
	4	13644	21144	36144	66145
	8	17902	29660	53176	100208
	16	26692	47241	88339	170535
8w2	1	8830	11517	16891	27638
	2	9045	11947	17751	29359
	4	10026	13909	21674	37205
	8	12263	18382	30621	55099
4w4	1	7539	8935	11726	17308
	2	7674	9204	12265	18387
	4	8219	10295	14446	22748
2w8	1	6894	7645	9146	12149
	2	6990	7836	9529	12915
1w16	1	6573	7003	7863	9583

Tabla 6.5: Variación del área del banco de registros debida al particionado (λ^2x10^6)

6.4 El lenguaje máquina

El lenguaje es un proceso de libre creación; sus leyes y principios son fijos, pero la manera de usar esos principios es libre e infinitamente variada. Incluso la interpretación y uso de las palabras conlleva un proceso de libre creación.

Noam Chomsky

En una arquitectura VLIW, una instrucción codifica varias operaciones individuales (load, add,...). Usando la técnica de *widening*, cada operación individual puede especificar más de una operación elemental. Por ejemplo, la configuración 4w1 necesita una palabra de instrucción suficientemente larga para codificar hasta 4 accesos a memoria y 8 operaciones en coma flotante, mientras que en la configuración 2w2, la palabra de instrucción debe codificar hasta 2 accesos a memoria y 4 operaciones en coma flotante (pero, en el mejor caso, ambas configuraciones pueden ejecutar el mismo número de operaciones básicas). Así pues, la longitud de la palabra de instrucción de la configuración 4w1 es el doble de la longitud de la palabra de instrucción requerida por la configuración 2w2, y el cuádruple de la longitud requerida por la configuración 1w4. Sin embargo, la configuración 2w2 puede necesitar más instrucciones para ejecutar un bucle que la configuración 4w1, ya que la segunda es más versátil.

La figura 6.5 muestra una comparación entre el tamaño de código de diferentes configuraciones, agrupando aquellas que tienen el mismo rendimiento máximo teórico (la gráfica muestra la suma del tamaño del código de todos los grafos probados). El tamaño de cada grupo de configuraciones ha sido escalado de manera que consideramos que la configuración con el código mayor (aquella que tiene un grado de *widening* igual a 1) hace de base y tiene un tamaño igual a 1. Puede observarse que el incremento en el número de instrucciones de una configuración que usa *widening* es ampliamente compensado por la reducción de código debido a la compactación de operaciones. Ésta es una ventaja adicional de la técnica de *widening*, que al reducir el tamaño del código puede reducir el número de fallos de la cache de instrucciones. Aunque para el estudio presentado en el capítulo 7 no afectará, dado que se asumirá una memoria perfecta (*hit ratio* del 100%).

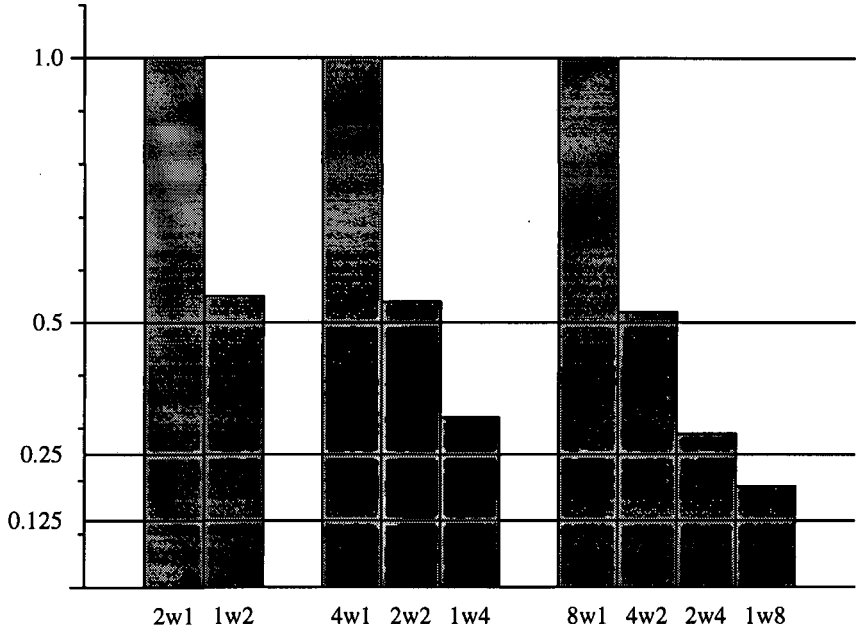

Figura 6.5: Comparación del tamaño del código generado para diferentes configuraciones

6.5 La memoria cache

Memory is like an orgasm. It's a lot better if you don't have to fake it.

Seymore Cray

Una ventaja de la técnica de *widening* es que requiere menos puertos de conexión con la memoria cache de primer nivel que la técnica replicación. A continuación discutiremos diversas soluciones utilizadas para implementar memorias caches multipuerto. Este punto no es original de la tesis y por tanto no supone ninguna aportación; simplemente quiere enfatizar las ventajas que se obtienen al reducir el número de puertos entre el banco de registros y la memoria cache, como hace la técnica de *widening*. Un estudio en profundidad de las necesidades de memoria cache puede encontrarse en la tesis de Toni Juan [Jua98]. Estudiar cómo afecta el uso de *widening* en la memoria forma parte del trabajo futuro de esta tesis (véase capítulo 8).

Implementar una memoria cache de datos con múltiples accesos es difícil ya que las peticiones de load/store que se reciben en paralelo no tienen por qué seguir propiedades de localidad. La dificultad radica en proponer un diseño capaz de dar cabida a las peticiones sin un gasto excesivo de área, y con un impacto no demasiado grande en el tiempo de ciclo del procesador. Las soluciones implementadas en los procesadores actuales pueden ser divididas en:

- *True multi-porting*. Con respecto al rendimiento, la mejor solución es implementar completamente el multiporting, pero es muy caro en términos de área por las mismas razones descritas para el banco de registros. El coste puede ser reducido a cambio de una degradación del tiempo de acceso, que puede afectar negativamente al rendimiento final del sistema.

- *Multiple Cache Copies*. Para **n** accesos, la cache puede ser replicada **n** veces, siguiendo un esquema similar al del particionado del banco de registros descrito anteriormente. Sin embargo al enviarse un store (que debe escribirse en todas las copias de la cache), por razones de coherencia no se deben enviar otras peticiones de cache en paralelo. Como aproximadamente el 30% de los accesos a memoria son stores [HePa96], esta solución está muy limitada. El DEC Alpha 21164 [DEC94] implementa una cache de dos puertos con esta técnica.

- *Virtual Multiporting*. En el IBM POWER2 [WhDh94], se usa un sistema de dos puertos virtuales: la SRAM cache es dos veces más rápida que la velocidad de reloj del procesador, de manera que es capaz de atender dos peticiones por cada ciclo del procesador. El nuevo DEC Alpha 21264 [MR96] también usa esta técnica. Sin embargo, esta solución no es factible si se requieren **n** accesos a la memoria (lo que implica una cache **n** veces más rápida que la velocidad de ciclo del procesador).

- *Multi-banking*. En esta solución, la cache se divide en varios bancos y cada banco puede ser accedido de manera independiente de los otros bancos. El mayor coste del multibanco es el *crossbar* necesario para encarrilar las peticiones de load/store al banco

correspondiente, y otro *crossbar* desde los bancos hasta los puertos con el banco de registros. El área del *crossbar*, así como el tiempo de acceso a la cache se incrementa conforme aumenta el número de bancos. Si embargo, el principal problema son los conflictos de los bancos, como demuestra la experiencia con memorias multibanco en procesadores vectoriales [Pei96]. Un conflicto en un banco se da cuando dos o más peticiones tratan de acceder al mismo banco simultáneamente. Estos conflictos son parcialmente eliminados en procesadores vectoriales utilizando un número grande de bancos. Sin embargo, esta solución no es práctica para memorias cache, y además, el hecho de incrementar el número de bancos implica un *crossbar* mayor, de manera que se incrementa el tiempo de acceso. Entre las máquinas que usan está técnica está el MIPS R8000 [MIPS94], con 2 bancos, y la cache de datos del Pentium[Int93], con 8 bancos.

En este trabajo no hemos tenido en cuenta el coste de la memoria cache, ya que hemos estado suponiendo una memoria perfecta. Sin embargo, hemos visto que la mayoría de los procesadores actuales usan métodos para tener una cache multipuerto que son difícilmente escalables para altos grados de ILP, cuando no muy costosos en términos de área y tiempo de acceso. Por esta razón consideramos que el uso de la técnica de *widening* tendría una ventaja adicional si tuviésemos en cuenta el acceso a cache.

6.6 Sumario y contribuciones

En este capítulo se han presentado métodos para evaluar el coste de las técnicas de replicación y *widening*. Respecto al área, se ha presentado cómo evaluar el coste de las FPUs y del banco de registros de manera independiente de la tecnología (en función de λ), observándose que el coste de dicho banco crecía linealmente cuando se aplica la técnica de *widening* y cuadráticamente cuando se aplica replicación.

También se ha evaluado el tiempo de acceso al banco de registros, haciendo una adaptación del método RFCT [Far97]. Se ha observado cómo *widening* incrementa el tiempo de acceso al banco de registros en menor medida que replicación.

Se ha apuntado el coste de ambas técnicas respecto a memoria cache, evaluando la reducción del tamaño de código producido por las operaciones anchas (respecto a la cache de instrucciones) y se han señalado los problemas de una cache de datos multipuerto, así como las soluciones utilizadas en procesadores actuales.

Las aportaciones de este capítulo son las siguientes:

- Evaluación independiente de la tecnología y comparación del coste en área del banco de registros, usando las técnicas de replicación, *widening* y ambas simultáneamente para un amplio espectro de configuraciones.
- Adaptación de un método de evaluación de tiempo de acceso al banco de registros, así como el estudio del tiempo de ciclo de bancos de registros donde se han aplicado las técnicas de replicación y *widening*.
- Estudio del tamaño del código producido por ambas técnicas.