UNIVERSIDAD DE CANTABRIA

DPTO. DE ELECTRÓNICA Y COMPUTADORES.

IMPACTO DEL SUBSISTEMA DE COMUNICACIÓN EN EL RENDIMIENTO DE LOS COMPUTADORES PARALELOS: DESDE EL HARDWARE HASTA LAS APLICACIONES.

Presentada por:

Valentin Puente Varona

Dirigida por:

Ramón Beivide Palacio.

SANTANDER, OCTUBRE DE 1999

Capítulo 6

Alternativas Estructurales.

En este capítulo nos detendremos en el análisis del impacto que tienen, en el rendimiento de las redes de interconexión, determinados planteamientos a nivel de la organización interna del encaminador. El objetivo fundamental es conocer de qué manera influye la combinación de diferentes aspectos estructurales y algorítmicos. En concreto, analizaremos varias propuestas que permiten evitar el problema de parada por bloqueo de la cabeza en las colas de almacenamiento de forma combinada con el empleo de mecanismos de encaminamiento adaptativos y deterministas.

6.1 Introducción.

En este capítulo se estudiarán soluciones que permitan evitar el problema asociado al bloqueo causado por la parada de la cabeza (*head-of-line blocking)* o HLB en el contexto de redes de interconexión para sistemas multiprocesador. Además, se analizarán los efectos combinados de aunar estrategias dirigidas a la evitación del bloqueo en la cabeza con mecanismos de encaminamiento adaptativo. Dentro del amplio número de grados de libertad que exhibe un problema de este naturaleza, el estudio sólo incorpora un análisis detallado en el que se combinan y comparan los resultados de rendimiento ofrecidos por ambas estrategias sobre el caso de redes *k-ary n-cube*.

En primer lugar, la mayor parte de las aplicaciones paralelas presentan un patrón de comunicaciones específico respecto a los destinos y que en general, se aleja de los patrones uniformes. Esto hace que los mecanismos de encaminamiento deterministas limiten la productividad máxima que puede alcanzar la red. La causa básica de esta degradación de rendimiento es la descompensación en la utilización que se hace de parte de los recursos físicos de la red, debido a las características intrínsecas del tráfico asociado (Ver Capítulo 3). Una de las soluciones adoptadas frente a este problema es proveer a la red de un cierto grado de adaptatividad a la hora de encaminar los mensajes desde su origen al destino. Con este tipo de sistemas se permite la multiplicidad de caminos, logrando así aprovechar mejor los recursos que ofrece la red. Además, de forma colateral, el routing adaptativo ofrece una mayor tolerancia a fallos que el routing determinista.

En segundo lugar, es conocido que un esquema de encaminamiento con almacenamiento en la entrada (*Input Queuing)* y política de almacenamiento FIFO es el más simple, pero también el que exhibe un rendimiento más pobre. En cada instante, solo el primer paquete de cada una de las colas de entrada es tomado en cuenta para su encaminamiento hacia el canal de salida correspondiente; en el caso de que este paquete colisione con otro dirigido al mismo puerto de salida o que el puerto de salida deseado no sea utilizable (porque el buffer en el router remoto se encuentra lleno) el resto de paquetes que le siguen en la misma cola se quedan parados (notar que esto es cierto únicamente en el contexto de los mecanismos de encaminamiento deterministas). Sin embargo, es muy probable que los diferentes paquetes en la cola deseen optar por otro puerto de salida diferente al del primer paquete y que en ese instante se encuentre libre. Este bloqueo innecesario es lo que se denomina bloqueo o parada de la cabeza. Como consecuencia de esto, un encaminador aislado, donde el patrón de destinos de cada uno de los puertos de entrada es aleatorio y con longitudes de paquete fija, se satura entorno al 60% de su capacidad total (*link capacity)*[71].

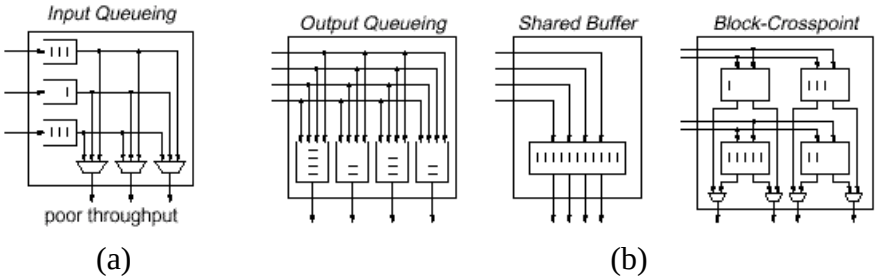

Figura 6-1. Diferentes Arquitecturas de Almacenamiento.

Frente a este problema se pueden emplear diversas soluciones, como utilizar espacios de almacenamiento más eficaces y adaptables a las características del tráfico. En este tipo de estrategias con colas no-FIFO, los buffers siguen teniendo un solo puerto de escritura. Esta organización proporciona bastante mejor rendimiento que la basada en almacenamiento en la entrada con colas FIFO [2][125][126]. Otro método para mejorar el rendimiento es proponer una arquitectura interna para el router que permita alcanzar un throughput mas alto. En la Figura 6-1.(b) se muestran algunas de las posibles soluciones a considerar.

Cualquiera de las dos alternativas arquitectónicas citadas previamente no está exenta de problemas. La primera solución implica la incorporación de una gestión substancialmente más complicada lo que afectará a la velocidad de reloj del encaminador. Por otro lado, recurrir a un espacio de almacenamiento en la salida puede implicar la necesidad de utilizar un espacio de almacenamiento más elevado que su contrapartida con bufferes en la entrada al ser necesaria la utilización de estructuras de almacenamiento multipuerto. Este *overhead* puede afectar notablemente al área del encaminador y dependiendo de la implementación de los buffers, puede venir acompañado de un aumento en su tiempo de ciclo. No obstante, esta última opción puede ser una alternativa factible en el entorno de las actuales tecnologías de implementación *hardware* ya que las perdidas de rendimiento en velocidad pueden ser inferiores a las asociadas a implementaciones que hagan uso de sistemas de almacenamiento en la entrada no-FIFO. Por este tipo de razones, nos centraremos, en principio, en analizar la segunda opción aunque posteriormente propondremos una arquitectura que sigue empleando espacios de almacenamiento en la entrada sin *HLB* empleando múltiples canales virtuales.

La primera parte de este estudio se centra en realizar un análisis comparativo entre los diferentes esquemas de almacenamiento y sistemas de encaminamiento sobre redes de interconexión de multiprocesadores en todas sus posibles combinaciones (para los pares encaminamiento determinista - adaptativo y almacenamiento en la entrada - en la salida). En nuestro estudio únicamente se considera el caso de redes *k-ary n-cube*. Fundamentalmente, se pretende evaluar, de

forma precisa, como se ha hecho en capítulos anteriores, el coste temporal que introducen planteamientos claramente orientados a mejorar la productividad de la red. Estos aspectos de bajo nivel pueden llegar a ser bastante más críticos que la productividad neta de la red, como hemos podido constatar en capítulos precedentes. Para efectuar un análisis adecuado y poder tener en cuenta estos factores, como anteriormente, se ha seguido una metodología que parte del diseño *hardware* de cada propuesta y llega hasta el análisis de rendimiento con el tráfico ofrecido por aplicaciones paralelas sobre una arquitectura ccNUMA pasando por la evaluación de rendimiento de cada una de las redes bajo condiciones de tráfico sintético.

Por último, el análisis se ha complementado analizando arquitecturas que emplean un número elevado de canales virtuales por línea física como método de evitación del *HLB*. Con este tipo de sistemas se mantiene el esquema de almacenamiento en la entrada con colas de almacenamiento tipo FIFO, pero se amplia el número de canales virtuales, manteniendo la capacidad total del canal físico asociado. De esta forma se logra reducir considerablemente el *HLB* al emplear colas de profundidad mucho menor. Como en el resto de propuestas, evaluaremos el coste real de estas estrategias considerando las restricciones tecnológicas.

6.2 Propuestas arquitectónicas para redes *k-ary n-cube*.

Con el fin de probar la viabilidad de cada uno de las diferentes alternativas se ha procedido a emplear cada uno de los mecanismos propuestos sobre dos esquemas de encaminamiento concretos. Se ha elegido el routing determinista y un routing adaptativo mínimo para redes *k-ary 2-cube*, introducidos en el Capítulo 3. Por lo tanto, en todos los routers se ha empleado el método de la *Burbuja* como sistema de evitación de *deadlock*. Con este esquema de encaminamiento son necesarios, únicamente, dos canales virtuales por línea física para asegurar la ausencia de *deadlock* en el caso adaptativo y no es necesario su uso en el caso determinista. Todos los encaminadores analizados emplean control de flujo VCT.

A continuación se presentan las diferentes estructuras de los cuatro primeros encaminadores cuyo rendimiento se comparará posteriormente.

6.2.1 Encaminadores con Buffer en la Entrada.

Este esquema de almacenamiento ha sido, con diferencia, el más empleado, existiendo innumerables encaminadores de sistemas reales que han recurrido a esta aproximación. Sin duda su mayor aceptación esta motivada por un menor coste *hardware* que otras alternativas de almacenamiento. El esquema básico para un encaminamiento determinista y adaptativo, basados en el algoritmo de la *Burbuja*, para redes *k-ary 2-cube* se muestran en las Figuras 6-2 y 6-3.

Como se puede apreciar en el caso del encaminador determinista el espacio de almacenamiento de cada canal físico se encuentra situado en la entrada. Dado que pretendemos comparar el rendimiento ofrecido por la red con y sin bloqueo por parada de cabeza, en las dos propuestas se emplea una estructura FIFO para los buffers de entrada. El crossbar empleado en el encaminador maneja los canales de tránsito con los routers vecinos y las líneas de inyección y consumo desde y hacia el nodo local. Tal y como está representado el router, se incorpora un elemento de sincronización por cada uno de los canales de entrada físicos. Como en encaminadores previos, esto es debido a que, tanto este encaminador como el resto, funcionan internamente de forma síncrona y de forma asíncrona en las comunicaciones con los encaminadores vecinos y el nodo de proceso local. De la misma forma que en el caso del encaminador determinista, en el encaminador adaptativo cada uno de los espacios de almacenamiento de los diferentes canales virtuales del encaminador se basa en una cola FIFO, con una profundidad diferente entre el canal adaptativo y el canal de escape.

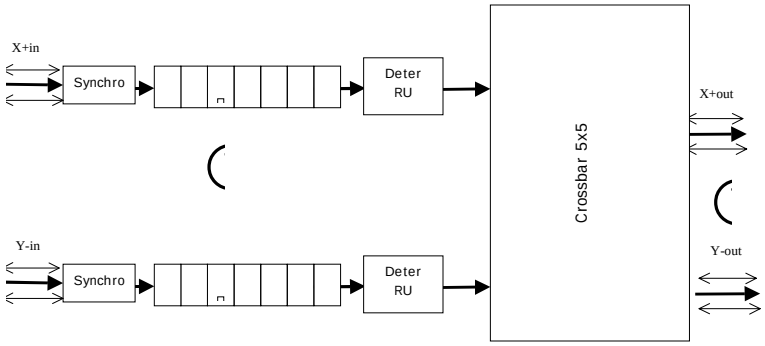

Figura 6-2. Esquema del router determinista, basado en el algoritmo de la *Burbuja,* con almacenamiento en la entrada.

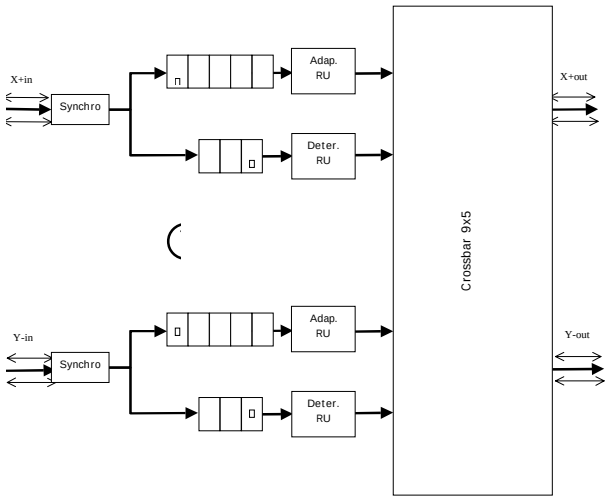

Figura 6-3. Esquema del router adaptativo, basado en el algoritmo de la *Burbuja,* con almacenamiento en la entrada.

Otro aspecto a señalar en este ultimo router es el sistema de arbitraje. Básicamente, el crossbar incorpora un arbitro por cada uno de los puertos de salida, completamente independiente con respecto al resto de puertos. Al no poseer un gestor global para todos los puertos de salida, es necesario que la unidad de routing de cada uno de los canales de entrada realice secuencialmente todas las peticiones correspondientes a todos los posibles canales de salida que acercan el paquete, almacenado en la cabeza de la cola, a su destino. Este sistema es un derivación del esquema de arbitraje de los routers deterministas, y pese a parecer intuitivamente que va a provocar una degradación de la productividad, en realidad al tomar en cuenta las dependencias tecnológicas logra mejor rendimiento que otras alternativas de arbitraje más complejas, como se demostró en el en el Capítulo 4.

La función de selección empleada a la hora de elegir la prioridad y orden de las peticiones que se efectúan hacia el crossbar está basada en X/Y dinámica. Esto implica que un paquete viaja siempre en orden creciente de dimensión hasta encontrar un bloqueo o el canal de destino ocupado por otro paquete. En este caso pasa a viajar por la dimensión inmediatamente superior y no la abandona hasta que quede agotada o encuentre una nueva congestión en el router siguiente.

6.2.2 Encaminadores con Buffer en la Salida.

Frente a las estructuras propuestas en el punto anterior, pasaremos a describir los dos encaminadores con almacenamiento en la salida que han sido empleados en este análisis.

6.2.2.1 Encaminador Determinista.

Como en el resto de encaminadores, la ausencia de ciclos de dependencia entre canales se logra utilizando un control de flujo basado en la *Burbuja*. Recordando este mecanismo, en el caso de buffers en la entrada, es suficiente con que el buffer asociado a la dimensión y dirección correspondiente tenga al menos dos huecos libres para permitir la inyección, además de que exista espacio para un paquete en el buffer remoto (condición impuesta por el propio control de flujo VCT). Por ello, en la práctica, con almacenamiento en la entrada se requieren al menos 3 huecos para lograr la inyección o cambio de dimensión. En el caso de buffers en la salida las restricciones de inyección con información local pueden ser rebajadas. De hecho para permitir la inyección del nuevo paquete basta con la existencia de 2 huecos en el buffer de salida. Por lo tanto, el esquema de almacenamiento en la salida reduce las restricciones impuestas por el algoritmo de la *Burbuja* con respecto al esquema con almacenamiento en la entrada. Sin embargo, es necesario tomar en consideración el hecho de que no existe ningún mecanismo de arbitraje de acceso a los puertos de salida del encaminador, por lo que hay que recurrir a la utilización de memorias multipuerto. Bajo estas condiciones, para asegurar el correcto funcionamiento de la *Burbuja*, es

necesario restringir la inyección en aquellos casos en que se supere un nivel de ocupación requerido por la implementación de la cola multipuerto para la aplicación correcta del control de flujo VCT, más un paquete adicional. Teniendo en cuenta estos aspectos, la estructura general del router determinista se muestra en la Figura 6-4. Como se puede apreciar, el número de puertos de escritura requeridos para las memorias multipuerto de la dimensión *X* es 2 y 4 para los de la dimensión *Y*. La FIFO ubicada en la entrada permite aplicar el control de flujo sin recurrir a el estado de ocupación de las colas de salida. De no incorporar este elemento la gestión de las señales de *Stop* se complicaría substancíalmente.

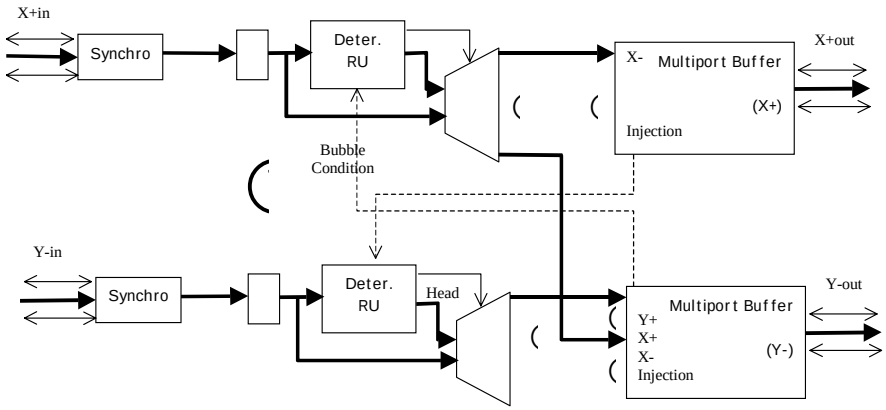

Figura 6-4. Encaminador determinista con método de evitación de deadlock basado en la *Burbuja* y con buffers en la salida.

6.2.2.2 Encaminador Adaptativo.

De forma similar al encaminador adaptativo con almacenamiento a la entrada, este encaminador toma como sistema de evitación de bloqueo el método de la *Burbuja*. Este método solamente requiere dos canales virtuales por línea física. A la vista de los comentarios realizados en el encaminador determinista con buffers en la salida, la estructura resultante es la que se muestra en la Figura 6-5.

En este caso es claro que la incorporación del canal adaptativo introduce un coste añadido con respecto a la propuesta previa en el sentido de que es necesario replicar recursos por cada uno de los dos canales físicos empleados. Además, es necesaria una cantidad superior en las interconexiones de los módulos internos del router y *fundamentalmente* es imprescindible un número considerable de memorias multipuerto con un número más elevado de canales de escritura que en el caso determinista. De hecho, en este caso se requiere que los buffers multipuerto incorporen hasta 7 canales de entrada para las memorias asociadas a las salidas adaptativas y deterministas en la dimensión *Y* y 5 canales para las deterministas en la dimensión *X*. Además, con el fin de no ampliar hasta un límite no sostenible el número de puertos de las memorias, es necesario tener memorias independientes para los puertos de salida adaptativos y de escape, lo

que implica que sea preciso incorporar un controlador externo para la multiplexación de los canales virtuales, resultando así que el pipeline tenga una etapa más que el caso determinista. A la vista de estas consideraciones, se considera esta propuesta excesivamente costosa su implementación en estos términos.

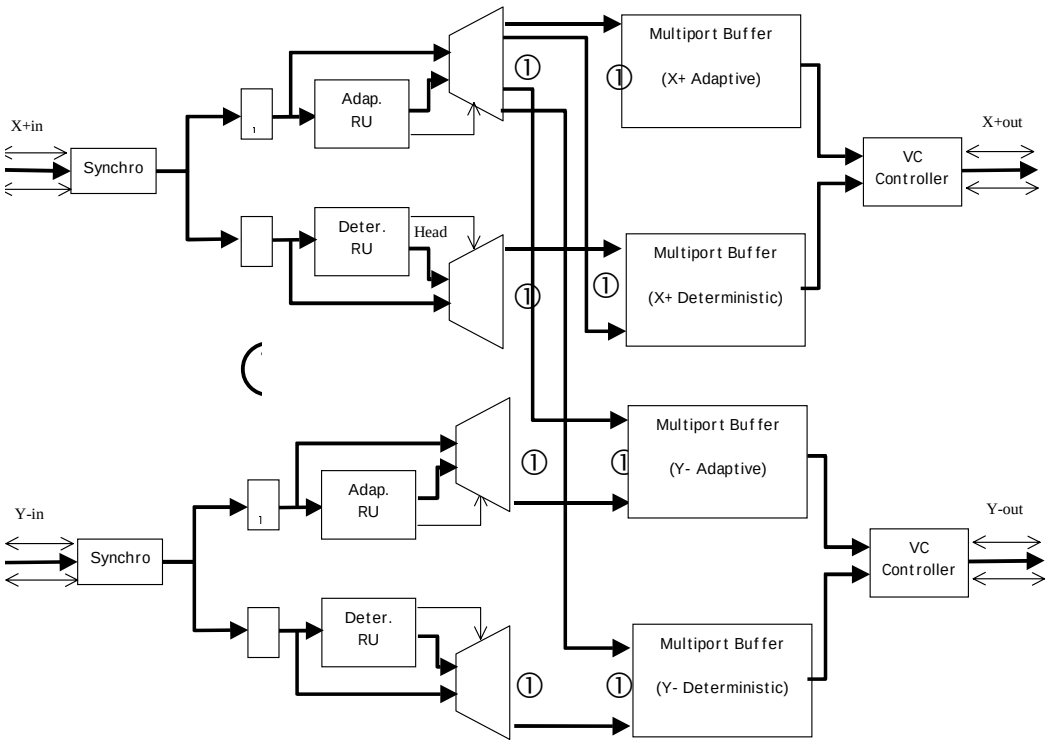

Figura 6-5. Encaminador adaptativo con método de evitación de *deadlock* basado en la *Burbuja* y con buffers en la salida.

Afortunadamente, un hecho fácilmente constatable en este tipo de encaminadores (canal virtual adaptativo + vía de escape determinista), es que la ocupación de los buffers deterministas, en general, es bastante más baja en promedio que la de los canales adaptativos. En la Figura 6-6 se puede apreciar, como ejemplo, el nivel de ocupación promedio de cada una de las clases de canales virtuales para una red 16x16 con un tráfico sintético de tipo matriz transpuesta y con un nivel de carga próximo a la saturación. A la vista de estos datos, no parece necesario emplear almacenamiento en la salida para los canales virtuales deterministas. Por lo tanto, es lógico plantearse una estructura híbrida para los canales del router, es decir, emplear almacenamiento en la entrada para los canales deterministas y almacenamiento en la salida para los canales adaptativos. Ello da lugar a una estructura para el encaminador como la que aparece en la Figura 6-7. Con este planteamiento, además de reducirse notablemente el área total, frente a la propuesta anterior, se reduce notablemente el número de entradas de las memorias multipuerto. Además, la etapa de multiplexación de los canales virtuales puede ser agrupada de forma conjunta con el

buffer de salida, lo que permite reducir en un ciclo la longitud del pipeline frente a la propuesta anterior.

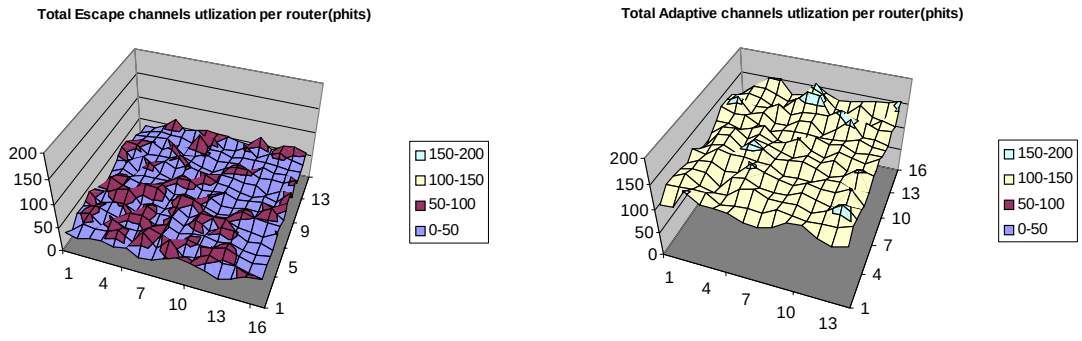

Figura 6-6. Mapas de ocupación promedio para las colas adaptativas y deterministas en un toro 16x16 con tráfico Matriz Transpuesta.

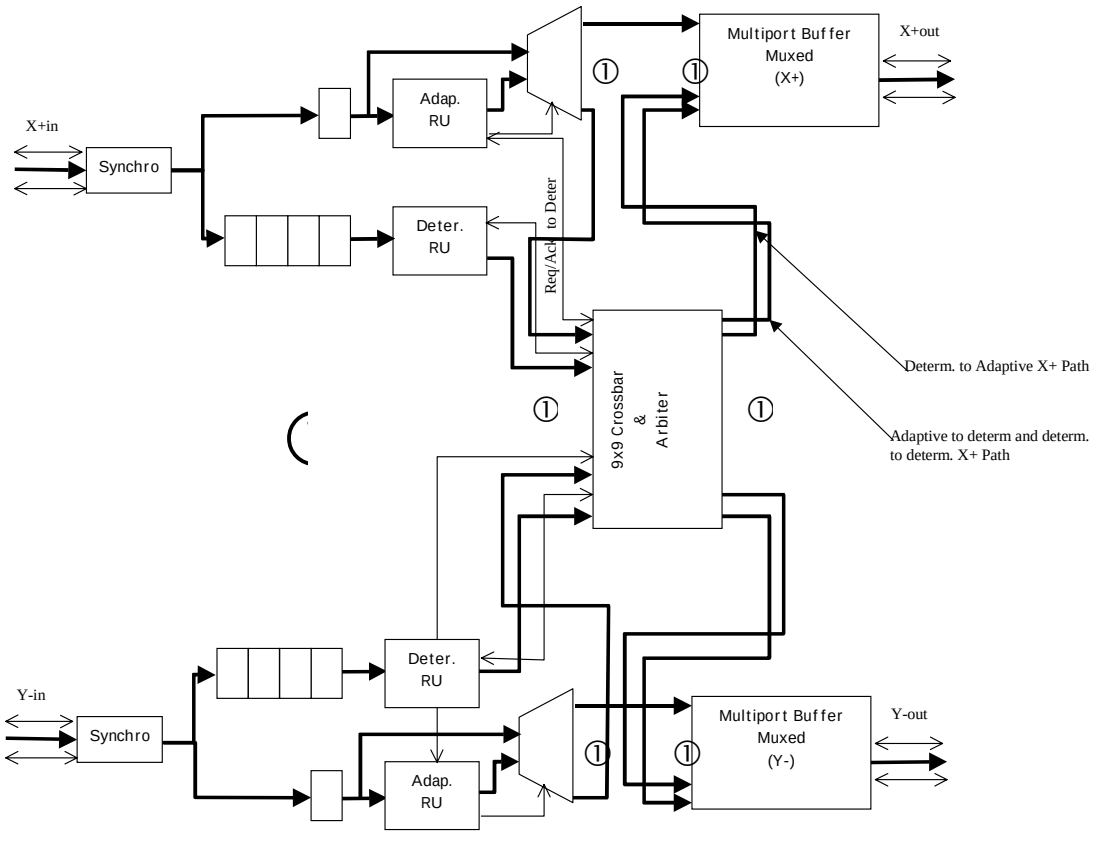

Figura 6-7. Encaminador adaptativo con método de evitación de *deadlock* basado en la *Burbuja* y con esquema de almacenamiento híbrido.

Para la implementación de este router son necesarias 5 memorias multipuerto (consumo y canales de transito adaptativos). Cada memoria multipuerto ha de poseer un total de 4 puertos de escritura. Los 3 primeros corresponden a los canales de entrada adaptativos y el último recibirá los paquetes provenientes de cualquiera de los canales deterministas de entrada que pretendan pasar a un canal adaptativo en el encaminador actual. Como se puede apreciar en la figura, es necesaria la incorporación de un crossbar que gestione los saltos entre las diferentes clases de canales. Su tarea es permitir el salto tanto desde los canales virtuales de entrada adaptativos hacia una salida determinista, el paso desde los canales deterministas de entrada hacia un canal adaptativo de salida y el paso desde los canales deterministas de entrada hacia los canales deterministas de salida. La solución adoptada no esta libre de contención, pero el número de paquetes que avanzan por canal adaptativo es muy superior a los que pretenden cambiar de canal o avanzan por el canal determinista, luego es previsible que no afecte excesivamente al throughput frente a la propuesta de utilizar almacenamiento en la salida tanto para el canal adaptativo como para el determinista. De hecho, experimentalmente se comprueba que la pérdida de rendimiento es prácticamente despreciable.

6.3 Implementación Hardware.

Una vez descritos los distintos encaminadores alternativos, un paso previo al análisis de rendimiento es determinar las implicaciones tecnológicas de cada propuesta. La vía elegida en este capítulo, como en el resto del trabajo, es proponer directamente un diseño *hardware* para cada tipo de encaminador y caracterizar de forma precisa sus parámetros fundamentales. Esta tarea se ha realizado siguiendo la metodología descrita en la Sección 2.2.4 en la página 36. En la síntesis de estos encaminadores el proceso tecnológico elegido corresponde a una tecnología de 0.35µm, 5 capas de metal y 2 niveles de polisilicio de la *foundry* MIETEC/ALCATEL (Ver Sección 2.2.3 en la página 35). A diferencia del resto del trabajo se ha optado por emplear una tecnología de implementación más evolucionada, pasando de 0.7µm a 0.35µm. Esta decisión ha estado motivada por el hecho de que el tipo de estructuras que aparecen en los encaminadores con espacio de almacenamiento en la salida presentan una complejidad elevada debido fundamentalmente al elevado interconexionado. Esta nueva tecnología soporta hasta 5 niveles de metalización lo que permite abordar los diseños y alcanzar implementaciones lógicas viables. De hecho, los resultados de este estudio están condicionados al empleo de tecnologías de implementación avanzadas. Aunque ésta no sea actualmente de las más empleadas por la industria microelectrónica, si que es suficiente para cubrir nuestros propósitos.

A continuación pasaremos a detallar los aspectos más relevantes de la implementación *hardware*. En este sentido, es necesario hacer un estudio detallado de las memorias multipuerto,

dado que es el componente crítico de los routers con almacenamiento en la salida. El resto de componentes de los encaminadores son derivaciones, prácticamente idénticas, de encaminadores expuestos en capítulos precedentes.

6.3.1 Implementación de las Memorias Multipuerto. Memorias FIFO Segmentadas.

Este componente es crítico y de él depende la viabilidad de optar por métodos de almacenamiento en la salida. Las memorias multipuerto completas son muy costosas, debido fundamentalmente a que cada posición de memoria debe poseer múltiples filas y líneas asociadas. Por ello, en la mayoría de los sistemas se opta por emplear bancos de memoria entrelazados con un solo puerto. Por ejemplo, los buffers en el "*Knockout Switch*" [135] utilizan esta técnica en una arquitectura con almacenamiento en la salida. Este tipo de métodos implican una baja utilización de la memoria y la replicación de múltiples elementos.

Cuando una memoria entrelazada es empleada como buffer de almacenamiento para paquetes de múltiples palabras (*phits*), todos los accesos en sus puertos son secuenciales. Esta característica permite emplear un solo registro de direcciones para los bancos de memoria. Esto es equivalente a agrupar todos los bancos de memoria en uno solo, con un bus de datos tan ancho como el número de bancos a considerar (puertos de la memoria) multiplicado por el tamaño de la palabra. Este único banco suele denominarse *"wide memory"*. El manejo de *buses* de datos tan anchos es perfectamente asumible para una implementación VLSI actual. Un sistema que utiliza este método de almacenamiento en su buffer central es el switch "*Vulcan*" de IBM [121].

Sin embargo, este tipo de sistemas tiene un problema serio a la hora de emplearse en sistemas como los analizados en este trabajo. Antes de poder escribir una palabra en la memoria hay que esperar a la recepción completa del paquete [72]. Por lo tanto, antes de que cualquier paquete pueda comenzar a ser transmitido por el buffer de salida asociado es imprescindible que el paquete completo haya alcanzado la memoria y por tanto el encaminador. Es claro que este problema hace inviable la utilización de esta arquitectura para implementar los buffers multipuerto necesarios, dado que su utilización requiere el uso de un control de flujo *Store and Fordward*. Para resolver este problema, en este tipo de implementaciones se suele recurrir a un camino de datos alternativo en forma de crossbar, como en el caso del *Vulcan.* Sin embargo, en [72] se propone una arquitectura alternativa que elimina el problema de las implementaciones basadas en la *wide memory* y que también reduce notablemente los costes de la memoria entrelazada. La arquitectura propuesta corresponde al buffer central del switch ATM "*Telegraphos*" [73]. La propuesta realizada en ese trabajo se basa en la utilización de una memoria segmentada (*pipelined memory*). Esta organización, propuesta para la memoria multipuerto, es aplicable en los

mismos casos que la basada en la *wide memory*, es decir, cuando el acceso en cada uno de los puertos es secuencial. De esta forma, se reduce de forma notable el coste que implican las arquitecturas basadas en bancos de memoria entrelazados. Esta arquitectura es similar a la memoria entrelazada pero la generación de la dirección de escritura y lectura en cada uno de los bancos es la correspondiente al banco anterior en el ciclo precedente. De esta forma se usa la misma dirección en los bancos siguientes en ciclos consecutivos. Esto simplifica el control global de los bancos, permitiendo una implementación VLSI mas rápida que en el caso de basar la implementación en una *wide memory* o en una memoria entrelazada.

Tomando como base la estructura de la memoria multipuerto segmentada, comentada previamente, se ha propuesto una arquitectura para las memorias multipuerto necesarias en los diferentes encaminadores. La idea original, donde se aplicaba este concepto, correspondía a un buffer central, por lo tanto, con varios puertos de escritura y lectura. En nuestro caso existirá una memoria multipuerto por cada uno de los puertos de salida y por tanto solo necesitan un único puerto de lectura para cada memoria. De la misma forma que en el caso de las memorias situadas en los puertos de entrada, se ha recurrido a una política de utilización del puerto de salida FIFO. Por lo tanto, cabe denominar el buffer multipuerto utilizado como una "*FIFO Multipuerto Segmentada"* ó *PMFIFO*. Un esquema básico de la estructura para una memoria de este tipo, simplificada a dos puertos de escritura en la que los paquetes tienen un longitud de 2 *phits*, es la que aparece en la Figura 6-8.

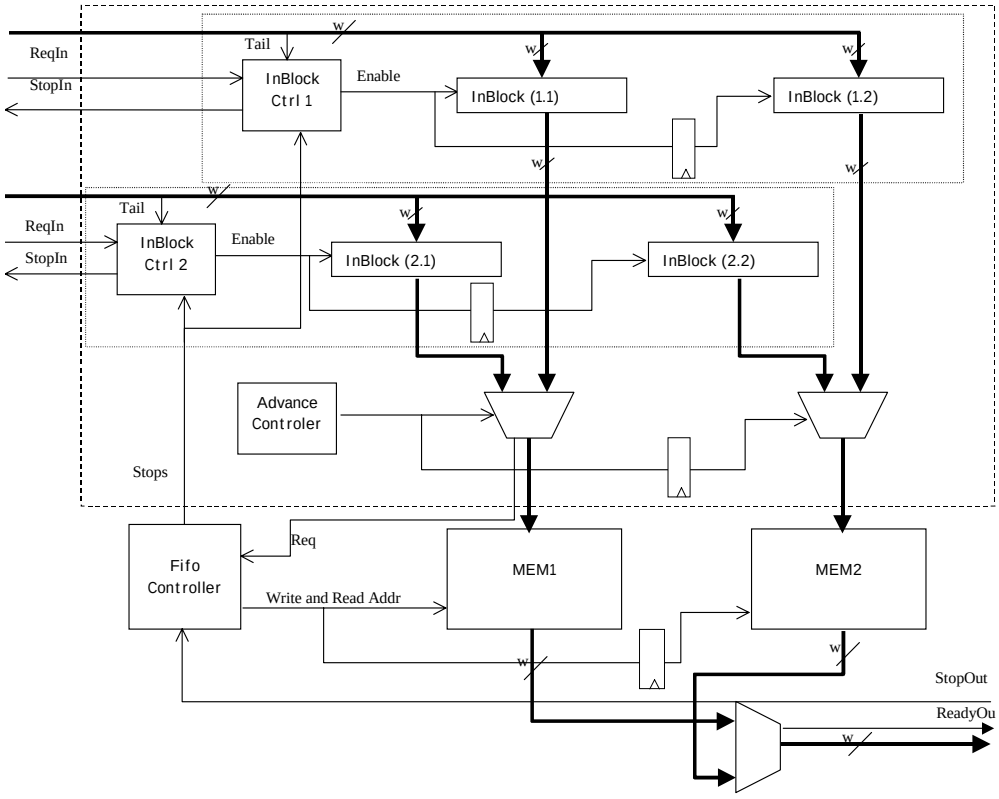

Figura 6-8. FIFO Multipuerto segmentada.

El número total de ciclos de paso para esta etapa, en zonas bajas de carga, es de dos ciclos, frente a los tres ciclos de la implementación presentada en [72]. Esto es debido a que el puerto de lectura es único. De no ser así, sería preciso incorporar a la salida un bloque muy similar al de entrada.

En cuanto a la estructura de la FIFO multipuerto necesaria para el router adaptativo con almacenamiento híbrido (Ver Figura 6-7), es necesario realizar la multiplexación entre la escritura del canal virtual adaptativo (correspondiente a la salida de la FIFO multipuerto) y el canal virtual determinista. Esta tarea puede realizarse sin afectar el pipeline de la memoria desde los puertos de entrada con destino adaptativo, es decir manteniendo su longitud en 2 ciclos, con una estructura similar a la mostrada en la Figura 6-9. El efecto introducido por la incorporación del canal determinista a multiplexar es que necesita, obviamente, un ciclo de reloj para atravesar la última etapa. El arbitrio en el puerto de escritura es realizado por el "*VC Control*" siguiendo un criterio de asignación *Round-Robin* de manera que se evite una posible inanición para cualquiera de los dos canales virtuales. Puesto que el número de canales a arbitrar es tan solo dos y que la multiplexación de los canales virtuales se realiza a nivel de paquete, este componente es sencillo y no introduce ningún *overhead* en el comportamiento de la PMFIFO original.

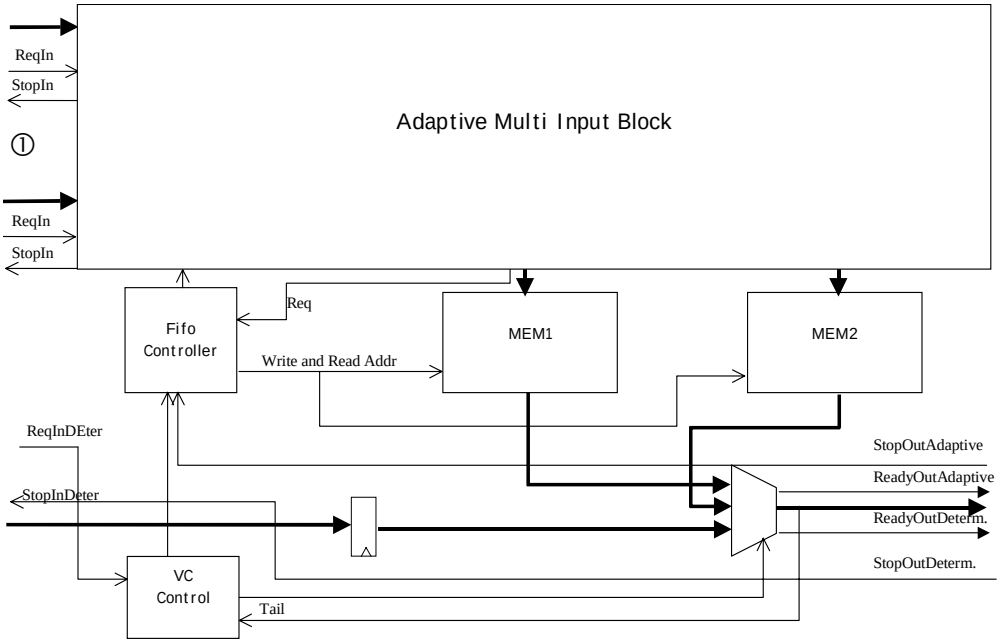

Figura 6-9. FIFO Multipuerto segmentada con multiplexación de dos canales virtuales.

El número de etapas necesarias para un correcto funcionamiento ha de ser igual al número de *phits* con compongan cada paquete y mayor o igual que el número de puertos de escritura. Cuando los paquetes estén compuestos por una cantidad elevada de *phits*, el número de etapas necesarias puede llegar a ser muy alto. Un número de etapas alto en el pipeline de la memoria implica un área considerable debido, fundamentalmente, a un incremento en la complejidad del bloque de recepción de las entradas (*Adaptive Multi Input Block*). La solución adoptada frente a este problema es paralelizar varios *phits* del paquete en una sola palabra más ancha a la entrada de los datos en el encaminador. Sin embargo, al agrupar un número arbitrario de *phits* por palabra la longitud del pipeline del router (en número de ciclos) puede verse afectada. Una solución a este problema es agrupar solo dos phits por palabra. Esto se puede realizar situando memorias FIFO asimétricas en la entrada (el bus de lectura es el doble de ancho que el de escritura) y la serialización repitiendo el proceso inverso en las memorias multipuerto de la salida. Un esquema de esta solución, para el caso del encaminador adaptativo, puede verse en la Figura 6-10.

En este caso, la longitud del pipeline del encaminador queda inalterado con respecto al caso en el que no se realiza la paralelización de los datos. Esto se logra avanzando una copia de la cabeza del mensaje a la unidad de routing correspondiente sin esperar a recibir el siguiente *phit* del mensaje. Una vez gestionadas las conexiones a realizar (tanto por la propia RU como por el crossbar), en el caso de ser necesario, se incorpora la cabecera convenientemente decrementada

en la primera palabra almacenada en la FIFO. Por lo tanto, el tiempo que tarda en atravesar la cabeza el encaminador queda inalterado.

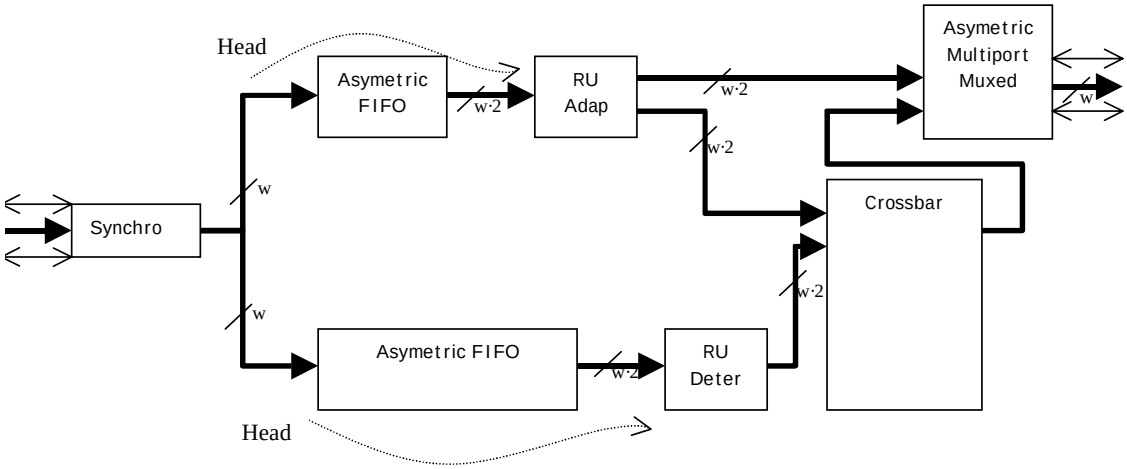

Figura 6-10. Estructura que permite reducir el número de etapas en el buffer multipuerto para el caso del encaminador adaptativo.

6.3.2 Resultados de la implementación *Hardware*.

Las estructuras de los pipelines de las implementaciones hardware correspondientes a los diferentes routers a evaluar se detallan en las figuras siguientes.

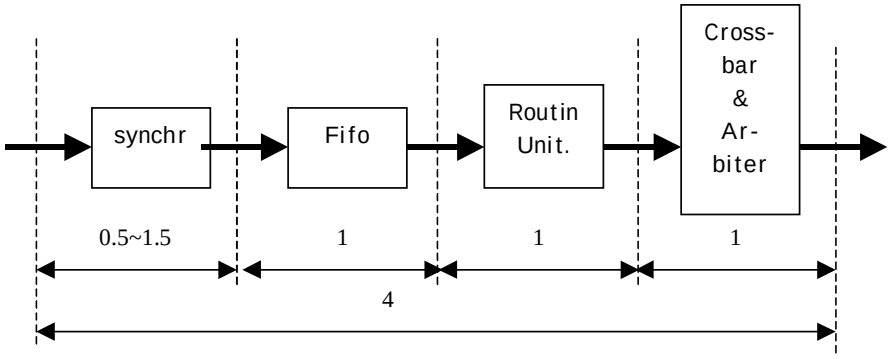

Figura 6-11. Estructura del pipeline para los encaminadores con almacenamiento en la entrada (determinista y adaptativo).

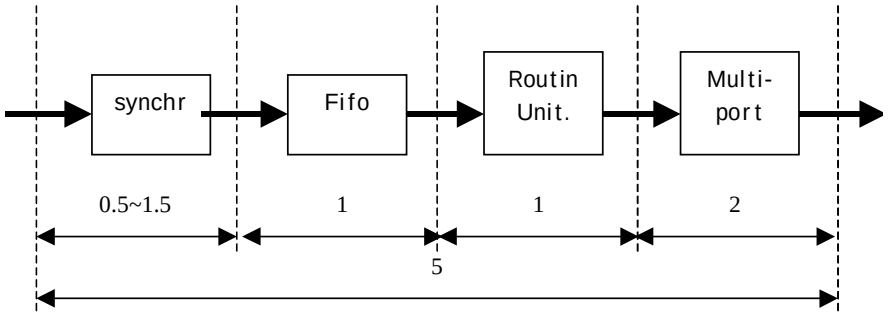

Figura 6-12. Estructura del pipeline para los encaminador determinista con almacenamiento en la salida y los caminos de salida adaptativos del encaminador híbrido.

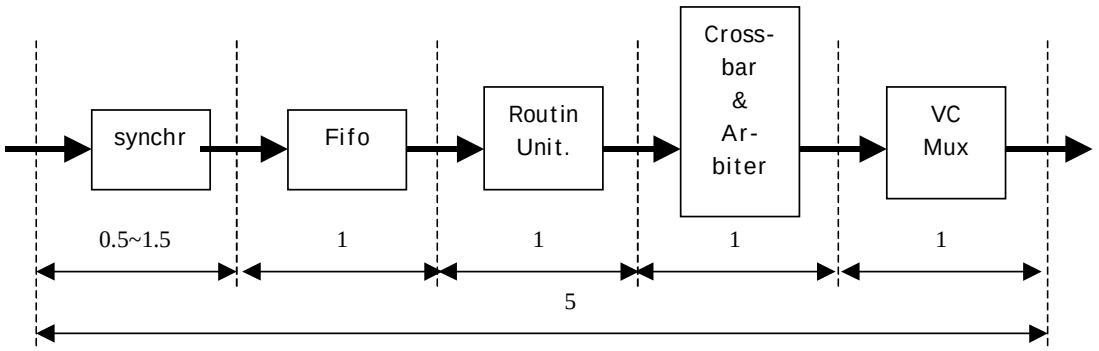

Figura 6-13. Estructura del pipeline para los canales de entrada deterministas en el encaminador adaptativo con almacenamiento híbrido.

A continuación se muestran las características de tiempo y área (en condiciones típicas de operación) para cada uno de los componentes de los encaminadores evaluados sobre la tecnología empleada. Estos datos corresponden a un ancho de datos para los canales de comunicación entre routers de 33 bits (4 bytes +1 cola), es decir el tamaño de *phits* que se manejan en todos los routers es de 33 bits. Por otro lado, se ha fijado la longitud de paquete a 10 *phits* lo que implica que cada paquete incorpora un total de 40 bytes de información. El empleo de 33 bits en la anchura de los canales de comunicación en lugar de 17 como en capítulos previos, queda justificado porque estamos empleando una tecnología bastante más evolucionada. Con esta elección, el *pin-count* del encaminador será de 350 pines, lo que es perfectamente realista desde este punto de vista. Por otro lado, los valores fijados para la longitud de los paquetes se encuentran condicionados por la configuración elegida para el sistema multiprocesador que permitirá evaluar el rendimiento de cada red de interconexión con cargas reales.

Por otro lado, la capacidad de almacenamiento de las FIFOs se ha fijado de tal manera que, en el caso del encaminador adaptativo con almacenamiento en la salida, la ganancia en *throughput*

que implicarían valores más elevados es despreciable frente al incremento de área del encaminador. Experimentalmente, como en evaluaciones previas, se observa que este hecho sucede a partir de una capacidad de almacenamiento de 4 paquetes por canal virtual. Con el fin de realizar una comparativa justa entre todos los encaminadores se opta por fijar la capacidad de las memorias multipuerto en el encaminador de referencia a 4 paquetes y en función de los requerimientos totales de almacenamiento que necesita este encaminador, ajustar la capacidad de las memorias del resto de encaminadores. La capacidad de cada uno de los tipos de almacenamiento aparece en la Tabla 6-1. Como se puede apreciar, la capacidad resultante de todos los encaminadores se sitúa en torno a 1.3KB, lo que es perfectamente asumible para una tecnología de 0.35µm.

Buffer Capacidad Packs (bytes)	**Encaminador**			
	InBuff Deter	**InBuff Adapt**	**OutBuff Deter**	**OutBuff Híbrido**
Fifo Inyeccion	2(80)	2(80)	6(240)	2(80)
Fifo Determinista	8 (320)	4(160)	1(40)	2 (80)
Fifo Canal Adaptativo	--	4(160)	--	1 (40)
Multipuertos	--	--	5(200)	4 (160)
Multipuertos Consumo	--	--	4(160)	4(160)
Total	34 (1360)	34 (1360)	34 (1360)	34 (1360)

Tabla 6-1. Configuración de los espacios de almacenamiento para cada encaminador.

módulo	**Encaminador**							
	InBuff Deter		**InBuff Adapt**		**OutBuff Deter**		**OutBuff Hibrido**	
	Critical Path (ns.)	Area (mm^2)	Critical Path (ns.)	Area (mm^2)	Critical Path (ns.)	Area (mm^2)	Critical Path (ns.)	Area (mm^2)
Syncronizador	2.72	0.020(x5)	2.72	0.020(x5)	2.72	0.020(x5)	2.72	0.020(x5)
Fifo Inyección	3.40	0.462(x1)	3.67	0.444(x1)	3.49	1.392(x1)	3.49	0.504(x1)
Fifo Entrada Determinista	**3.40**	**1.69(x4)**	3.67	0.848(x4)	3.49	0.282(x4)	3.49	0.504(x4)
Fifos Entrada Adaptativa	--	--	3.67	0.848(x4)	--	--	3.49	0.282(x4)
Unidad de Routing Determinista	2.88	0.016(x5)	3.66	0.117(x5)	3.07	0.043(x5)	3.49	0.118(x5)
Unidad de Routing Adaptativa		--	3.66	0.117 (x4)		--	3.49	0.145(x4)
Crossbar	3.17	0.192(x1)	**3.67**	**0.597(x1)**	--	--	**3.49**	**0.937(x1)**
Multipuerto	--	--	--	--	**3.35**	**1.380(x2)x 1.760(x2)y**	3.49	1.611(x4)
Multipuerto Consumo	--	--	--	--	**3.35**	**1.574(x1)**	3.49	1.611(x5)
Total	**3.40**	**7.594**	**3.67**	**9.305**	**3.35**	**10.066**	**3.49**	**13.910**

Tabla 6-2. Características de área y tiempo para cada uno de los routers evaluados.

Sobre los resultados mostrados en la Tabla 6-2, cabe indicar que en el caso del encaminador híbrido, que pese a tener un crossbar 9x9, éste no es el elemento que impone el tiempo de ciclo del encaminador. Realmente, puede observase que su tiempo de ciclo es ligeramente inferior al crossbar 9x5 (encaminador determinista con almacenamiento en la entrada). La razón es que el camino crítico de este módulo únicamente depende del número de entradas a arbitrar (como se vio en el capítulo anterior). Además, en el caso del crossbar 9x9 no es necesario efectuar la multiplexación de los canales virtuales de salida (gestión de dos líneas de protocolo por puerto de salida), lo que se traduce en una ligera reducción en su tiempo de ciclo. Esto hace que el componente que imponga el tiempo de ciclo en este router sean las memorias multipuerto y que éste sea sensiblemente menor que el encaminador adaptativo con buffer en la entrada.

Por otro lado, se observa cómo en los encaminadores con buffer en la entrada el componente que marca el tiempo de ciclo es el crossbar en el caso adaptativo y la cola de entrada en el caso determinista. Este último dato se mantiene aproximadamente igual pese a disminuir la capacidad de la cola FIFO. Es decir una disminución en la capacidad de las colas de entrada de este encaminador prácticamente no va a afectar a su latencia y si a su productividad.

En cuanto a las unidades de decisión de routing de los encaminadores con buffer en la salida se aprecia un aumento en área, originado fundamentalmente por el demultiplexor de las líneas de datos que permite en cada caso conectar los datos de entrada a cualquiera de las memorias multipuerto de salida o al crossbar, en el caso adaptativo.

Existen pequeñas diferencias entre módulos en área y tiempo pese a poseer las mismas características funcionales. Por ejemplo, la unidad de decisión de routing para los canales deterministas en el encaminador adaptativo con buffer en la entrada y la del encaminador adaptativo con buffer en la salida son totalmente idénticas y las características de tiempo y área son sensiblemente diferentes. La causa de este hecho se justifica por el proceso de optimización seguido en la síntesis lógica. Una vez determinado el componente crítico, y por tanto el tiempo de ciclo mínimo del encaminador, el resto de módulos se sintetizan de nuevo con las restricciones de tiempo de peor caso, permitiendo de este modo obtener una ligera reducción en el área. Este modo de operar permite equilibrar el *pipeline* del encaminador y reduce sensiblemente su área.

6.4 Otras alternativas a la evitación del HLB.

Hasta este momento, únicamente hemos considerado el almacenamiento en la salida como única alternativa a la hora de reducir el *HLB* y sus efectos en el rendimiento de la red de interconexión. No obstante, sabemos que es posible evitar el *HLB* utilizando espacios de almacena-

miento en la entrada sin emplear una política estrictamente FIFO. Existen diversas alternativas en este sentido, como las colas DAMQ propuestas originalmente por *Tamir* en [126]. De hecho, existen encaminadores como el SPIDER, que emplean una arquitectura a la entrada con buffers de este tipo. Otra aproximación al problema es emplear varios canales virtuales de entrada, cada uno de ellos con colas FIFO, pero de capacidad reducida. La propuesta es muy próxima, desde el punto de vista conceptual, a la hecha por *Tamir*. En nuestro caso hemos optado por la segunda alternativa ya que es más fácilmente abordable desde el punto de vista *hardware*. Como quedó claro en el Capítulo 3, un incremento del número de canales virtuales del encaminador implica un incremento de coste tanto en el tiempo de ciclo como en el área ocupada por el encaminador. Este problema, como ha sido expuesto ampliamente a lo largo de todo el trabajo, puede dar al traste con todas las posibles ganancias en productividad resultantes de evitar el *HLB*.

6.4.1 Encaminador completamente multiplexado.

Frente a la problemática de emplear un número elevado de canales virtuales, la mayoría de los encaminadores actuales emplean estrategias de segmentación dirigidas a simplificar, en la medida de los posible, la gestión del crossbar cuando el número de canales virtuales a manejar por línea física es elevado [113]. Básicamente el objetivo en estos casos es intentar mantener acotada la complejidad del encaminador realizando la multiplexación de los canales virtuales antes de que los datos alcancen el crossbar. De esta forma, el tamaño del crossbar requerido es únicamente dependiente del número de entradas y salidas físicas del encaminador. Para el caso de un router con almacenamiento en la entrada y algoritmo de encaminamiento adaptativo basado en la *Burbuja*, el aspecto que presenta es el mostrado en la Figura 6-14. Como se puede apreciar, se ha acoplado la función de multiplexación de los canales virtuales a la unidad de decisión de encaminamiento, lo que hace que el crossbar tenga tan solo 5 entradas en lugar de 9. Este primer encaminador le denominaremos *completamente multiplexado*.

Como efecto colateral de esta implementación, la longitud del pipeline del encaminador se va a incrementar en un ciclo, dado que se puede anticipar que el mecanismo de arbitrio y encaminamiento en la unidad de decisión de routing no puede efectuarse en un solo ciclo sin afectar seria-

mente el tiempo de ciclo del encaminador. En este caso es necesario recurrir a un sistema de arbitrio que gestione las posibles colisiones en los canales de entrada.

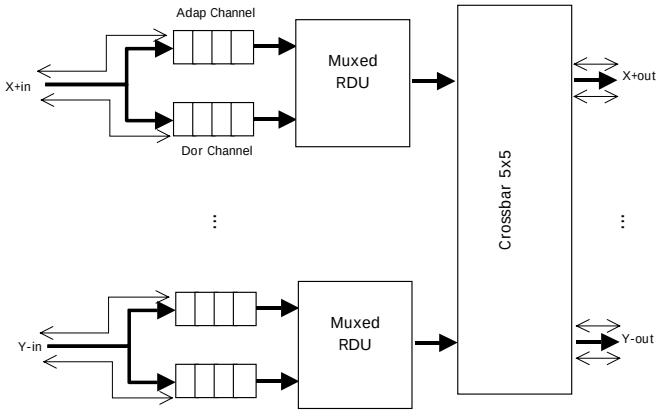

Figura 6-14. Router multiplexado con *Burbuja* adaptativa y almacenamiento en la entrada.

Con este tipo de estructuras es posible plantearse el empleo de un número superior de canales virtuales sin afectar de forma notable a los costes *hardware* del encaminador. En nuestro caso optamos por emplear espacios de almacenamiento en los canales adaptativos del encaminador con capacidad para cuatro paquetes. Según esto, para garantizar que los canales adaptativos sean completamente libres de *HLB*, es necesario emplear un canal virtual independiente para cada uno de esos paquetes. En la Figura 6-15 se puede apreciar la estructura del encaminador resultante.

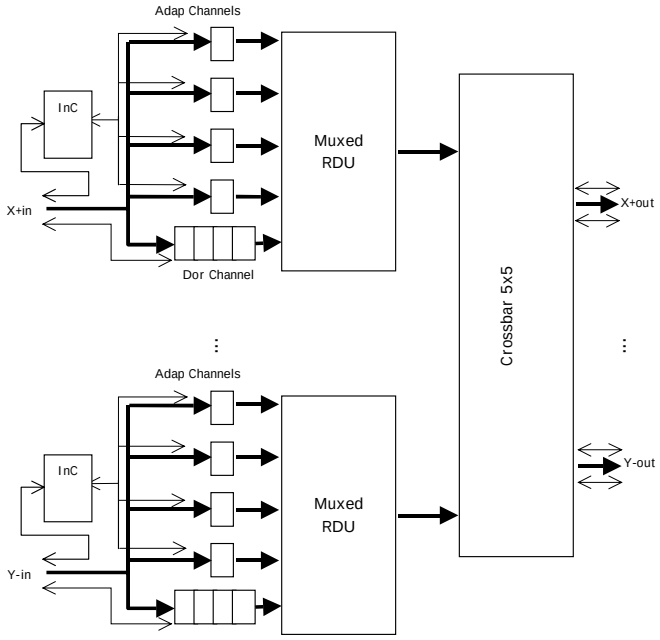

Figura 6-15. Incremento en el número de canales virtuales para la evitación del HLB en el encaminador multiplexado[1].

Como se ha comentado, a la hora de proponer la estructura del encaminador adaptativo con esquema de almacenamiento híbrido, los bajos índices de ocupación de los canales deterministas indican que no es rentable introducir sistemas de evitación de *HLB.* Por lo tanto seguiremos empleando espacios de almacenamiento con política de gestión de tipo FIFO y con la misma profundidad que en el encaminador con buffer en la entrada convencional. Es importante indicar que el tratamiento de los canales virtuales adaptativos no es tal desde el punto de vista exterior del encaminador, dado que no requieren ninguna línea adicional de protocolo. La gestión de las colas asociadas a esos canales de entrada se realiza mediante el bloque denominado *InC* y su modo de operación es muy similar al encargado de controlar la escritura en los bloques de entrada de la cola *PMFIFO* descrita en la Sección 6.3.1. En este caso el control se realiza mediante un *token* circulante, que a diferencia del anterior, ha de tener en cuenta el estado de ocupación de cada una de las colas antes de avanzar los datos y solo avanzará a la siguiente posición cuando la cola actual haya recibido el *phit* de cola. Este control habilitará el puerto de lectura de cada una de las FIFOs en el caso de que la escritura esté permitida. La señal de parada hacia el router vecino de todas estas colas puede ser obtenida a través de una sencilla operación AND de las señales de parada de las colas independientes. Además, este control se ve notable-

1. Notar que, por simplicidad, no se han representado los sincronizadores.

mente simplificado por la sencillez que implica el tipo de control de flujo que estamos empleando.

En este caso la unidad denominada *RDU Multiplexada* es la encargada de gestionar todas las peticiones que le llegan desde cada una de las colas de entrada de su canal físico. De la misma forma que la unidad descrita en la Figura 3-12 en la página 108, su tarea es decodificar la cabecera de los paquetes que llegan a ella y efectuar las peticiones hacia el crossbar. El modo de operación de la unidad de control sigue siendo muy similar, pero en este caso es necesario resolver las posibles peticiones que lleguen tanto de cualquiera de los canales adaptativos como desde el canal determinista. La estructura general de esta unidad esta representada en la Figura 6-16 y en ella destacan tres grandes bloques:

- Árbitro
  Es el módulo encargado del arbitraje entre las cinco colas a las que tiene que atender la unidad de routing. En este caso, el esquema de arbitraje emplea una política *Round-Robin.* Este módulo es prácticamente idéntico al utilizado para arbitrar cada uno de los subcrossbars de la propuesta OAC del capítulo previo (Ver Sección 4.2.1 en la página 142). Como en aquel caso, la tarea que ha de efectuar este módulo es elegir un solo canal de los que están solicitando atención a la *RDU*.
- Registro de cabezas.
  Este bloque es el encargado de registrar todas la cabezas de los mensajes almacenados en los canales que solicitan paso en cualquier instante. También ha de realizar las comparaciones con cero (como en el resto de encaminadores, estamos empleando routing relativo) y decrementar cada cabeza, almacenándolas en las posibles direcciones de avance del paquete. Es necesario que el módulo pueda recibir en paralelo cualquiera de las cabezas solicitantes ya que de esta forma permitirá operar a la unidad de control entre varios ciclos de petición sin pérdida de ciclos de reloj. Al ser necesario tal número de registros es previsible que el área ocupada sea alta, aunque asumible en aras de mejorar el rendimiento.
- Unidad de Control
  Es básicamente la misma que se empleaba en la implementación no multiplexada pero con la peculiaridad de que los ciclos de petición son abiertos. De esta forma, cuando varias peticiones llegan al árbitro, una sola de ellas progresa (en el ciclo siguiente) a la unidad de control. El registro de cabezas recibe también cuál es el canal que ha ganado el acceso y avanza el resultado de la comparación a la entrada de la unidad de control. Los bits de cola y de signo también son facilitados por este bloque. En el ciclo posterior al arbitrio, la unidad de control comienza a lanzar peticiones de forma secuencial al crossbar. En el caso de llegar al

último canal de salida posible solicitado sin obtener el acceso al puerto de salida por parte del crossbar, en lugar de volver a comenzar a solicitar el primer canal de salida para el paquete almacenado en el canal actual, como ocurría en la *RDU* convencional, en este caso se solicita al árbitro que cambie el canal de entrada (mediante la señal *Próximo Canal Disponible*). En el caso de que exista alguno esperando, el ciclo siguiente de petición al crossbar corresponderá al nuevo canal. Dado que el arbitrio es libre de *starvation*, es claro que, en general, el módulo completo también estará exento de este problema. En el caso de obtener el acceso a alguno de los puertos de salida solicitados, la señal *FSMC* es la encargada de indicar al registro de cabezas que vaya avanzando los datos correctos. Esta unidad es la encargada de controlar también las señales de parada hacia las colas de entrada conectadas a la unidad. Obviamente, cuando se detecta el paso de una cola, el árbitro deberá seleccionar un nuevo canal de entrada. En el caso de que no exista ninguna petición pendiente se esperará que llegue una nuevo requerimiento.

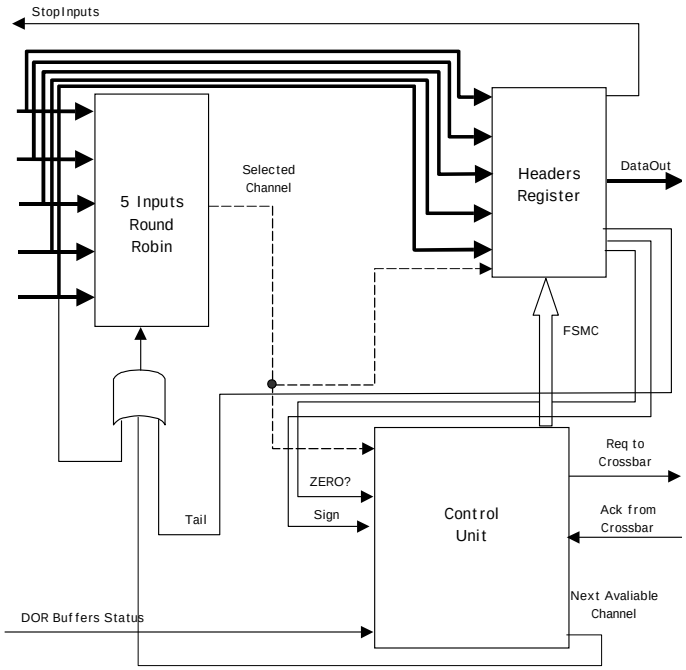

Figura 6-16. Estructura de la unidad de decisión de encaminamiento multiplexada.

6.4.2 Encaminador parcialmente multiplexado.

Con la estructura descrita previamente corremos el riesgo de limitar el rendimiento de los canales deterministas del encaminador. A diferencia del caso del encaminador adaptativo con esquema de almacenamiento híbrido, en este caso el tráfico de los canales adaptativos puede frenar el que circula por la vía de escape. En consecuencia, esto puede incrementar de forma

notable la cantidad de mensajes que circulan por los canales deterministas. En situaciones altas de carga, indirectamente, el incremento de las tasas de ocupación de los canales deterministas puede limitar las vías de escape de los canales adaptativos lo que, a la larga, produciría una caída en la de carga aceptada por la red.

Es claro que los canales deterministas se ven retardados ya que es aproximadamente 5 veces menos probable que un canal determinista gane el acceso al crossbar. Esto, junto a los tiempos de espera de tránsito (cuando cualquier paquete gana el acceso al puerto de salida, el resto de canales ha de esperar al paso de la cola) hace que, en promedio, el canal determinista se vea mucho más afectado que en el encaminador adaptativo con buffer en la salida. Recordar que en aquel caso los canales deterministas han de competir con un solo canal adaptativo para alcanzar la salida.

A la vista de estos datos propondremos una nueva estructura cuya única diferencia con la completamente multiplexada es que los canales deterministas tienen un canal de acceso directo al crossbar. De acuerdo con esto, la estructura del encaminador es la que se muestra en la Figura 6-17. Lo hemos denominado *parcialmente multiplexado* y como se puede apreciar posee básicamente los mismos componentes que el completamente multiplexado, salvo tres diferencias. Los canales deterministas tienen una unidad de decisión de routing propia, el crossbar tiene nueve entradas en lugar de cinco (es por tanto idéntico al encaminador adaptativo convencional con buffer en la entrada) y las *RDU*s multiplexadas tienen cuatro entradas.

Por último, y con el fin de verificar la problemática asociada al encaminador completamente multiplexado, hemos pasado a evaluar las tasas de ocupación por los canales virtuales deterministas en cada router para un tráfico uniforme y con una red 8x8 por encima de saturación. Los resultados se muestran en la Figura 6-18 y como se puede apreciar, para el caso del router parcialmente multiplexado, las tasas de ocupación son mucho menores que en el completamente multiplexado. A la vista de estos datos, es previsible que el encaminador completamente multiplexado presente problemas de caída de rendimiento al superar el límite de saturación de la red.

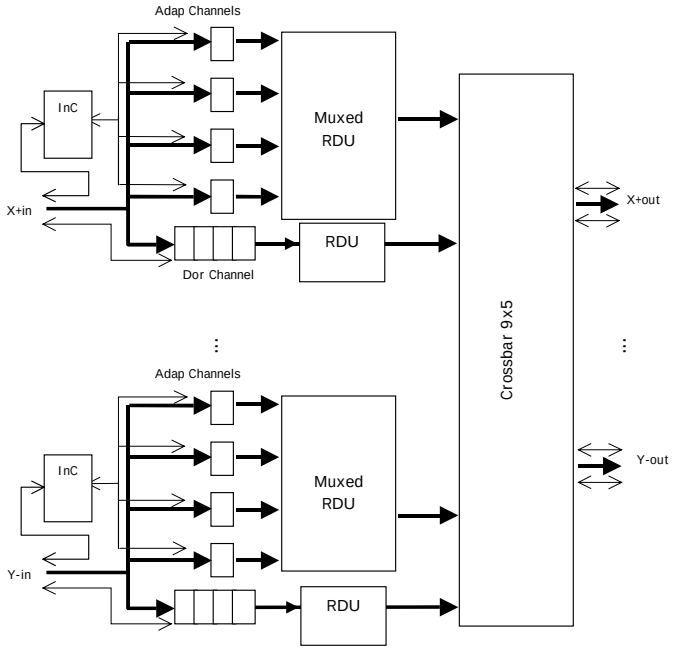

Figura 6-17. Encaminador adaptativo parcialmente multiplexado con buffer en la entrada.

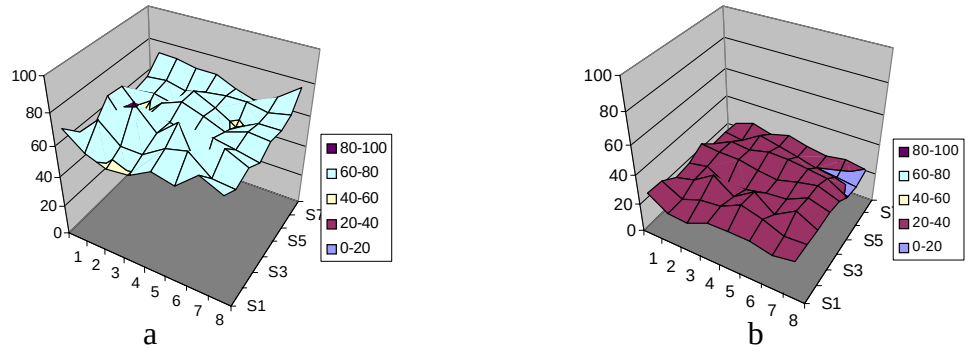

Figura 6-18. Ocupación en *phits* de los canales deterministas en el encaminador completamente multiplexado (a) y en el parcialmente multiplexado (b), para un patrón de destinos uniforme en un toro 8x8.

6.4.3 Implementaciones hardware.

Teniendo en cuenta el modo de operación de la RDU multiplexada, el *pipeline* de los dos encaminadores resultantes es el mostrado en la Figura 6-19 (notar que en el caso del parcialmente multiplexado los canales virtuales deterministas poseen un *pipeline* similar al caso del encaminador con buffer en la entrada y sin multiplexación)

.

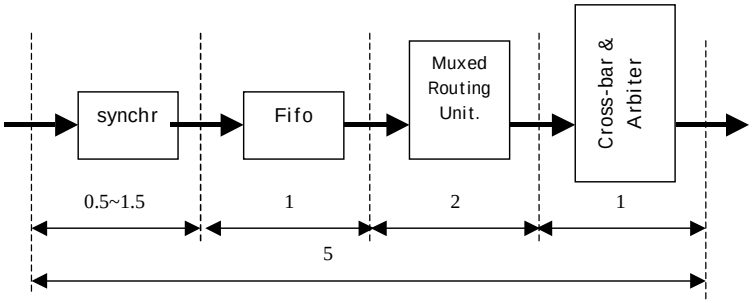

Figura 6-19. Estructura del pipeline para el encaminador adaptativo multiplexado y los canales adaptativos del parcialmente multiplexado.

El resto de componentes de los dos encaminadores es muy similar al encaminador adaptativo con buffer en la entrada, con la salvedad de que el crossbar tendrá cinco entradas en lugar de nueve para el caso del encaminador completamente multiplexado. Además, en el caso del canal de inyección la unidad de decisión de encaminamiento tendrá únicamente cuatro entradas. De la misma forma que en los encaminadores considerados previamente, la anchura de los canales de comunicación es de 33 bits (1 de cola) y la capacidad por canal virtual es de 4 paquetes. El tamaño de los paquetes se ha fijado a 10 *phits*. Por lo tanto, la capacidad total es idéntica a la del resto de encaminadores.

De acuerdo con esto, se ha procedido de la misma forma que en el resto de encaminadores, alcanzando una implementación de cada componente a nivel lógico cuyos resultados, para la tecnología empleada en este capítulo, se muestran en la Tabla 6-3.

	InBuff Adapt Part Muxed		**InBuff Adapt Muxed**	
	Critical Path (ns.)	Area (mm^2)	Critical Path (ns.)	Area (mm^2)
Syncronizador	2.72	0.020(x5)	2.72	0.020(x5)
Fifo Inyeccion	3.67	0.444(x1)	3.49	1.130(x1)
Fifo Entrada Deterministas	3.67	0.848(x4)	3.49	0.985(x4)
Fifos Entrada Adaptativas	3.67	1.124(x4)	3.49	1.130(x4)
Unidad de Routing Inyeccion Mux	3.67	0.415(x1)	3.52	0.535(x1)
Unidad de Routing Convencional	3.66	0.117(x4)	--	--
Unidad de Routing Mux.	3.67	0.415(x4)	3.52	0.574(x4)
Crossbar	3.67	0.597(x1)	3.49	0.241(x1)
Total	**3.67**	**11.157**	3.52	12.762

Tabla 6-3. Características de área y tiempo para los routers multiplexados.

Los resultados de síntesis muestran que el tiempo de ciclo del encaminador multiplexado es sensiblemente inferior al mostrado por el encaminador adaptativo no multiplexado con buffer en la entrada. La unidad que marca el critical path del circuito es la *RDU* multiplexada. De las dos etapas de este módulo, es la unidad de control la que marca el tiempo de ciclo. Por lo tanto, de acuerdo con la implementación realizada, el tiempo de ciclo del encaminador es prácticamente independiente del número de canales virtuales empleados. Aparentemente este hecho entra en contradicción con la teoría puesta en los inicios de este trabajo que permitió reducir a dos canales virtuales el mínimo número necesario para garantizar que la red es libre de interbloqueo. Sin embargo, es preciso puntualizar que este encaminador posee una etapa más en su pipeline lo que implica que la latencia de paso sea algo superior al caso no multiplexado. De esta forma el encaminador tiene un tiempo de paso en ausencia de contención de 17.6 ns frente a 14.68 ns del caso no multiplexado, lo que representa un incremento en el tiempo de paso de aproximadamente un 20%.

Bajo este punto de vista, el encaminador sin multiplexar (con un solo canal virtual adaptativo) sigue alcanzando mejores prestaciones en latencia de paso. Además, es necesario considerar que la penalización que sufren los canales virtuales deterministas incorporará previsiblemente una degradación en *throughput.* Por otro lado, de emplear un mecanismo de evitación de bloqueo basado en canales virtuales usando una estrategia de este tipo, no llegaríamos a las diferencias en latencia tan altas como las observadas en el Capítulo 3, pero en cierta medida tenderían a mantenerse. Además, es necesario señalar que el control de flujo VCT permite que la implementación multiplexada tenga las características previamente expuestas. El empleo del control de flujo *wormhole* tradicional sabemos que requiere multiplexación a nivel de flit, lo que indudablemente complicaría el multiplexador de los canales virtuales y fundamentalmente el arbitraje del crossbar.

En cuanto al encaminador parcialmente multiplexado, el tiempo de ciclo lo sigue marcando el crossbar. Teniendo en cuenta esto y que su *pipeline* tiene un ciclo más que el encaminador sin multiplexar, su latencia base sera aproximadamente un 25% mayor que la característica del encaminador no multiplexado.

Por otro lado, el área de ambos encaminadores sufre un incremento considerable al compararlo con el caso no multiplexado. Este incremento está aproximadamente entre un 25% y un 30%. Está causado por el incremento de complejidad de las colas de almacenamiento del encaminador y la inclusión de la unidades de decisión de routing, cuyo registro de cabezas implica un aumento considerable de área al compararla frente a la *RDU* convencional. Pese a estos resultados, estos encaminadores no llegan a ser tan costosos en área como los encaminadores con

buffer en la salida (ya sea el encaminador adaptativo o el determinista). Las diferencias se sitúan entre un 10% y un 15%. Por lo tanto, los resultados parecen indicar que, a priori, las opciones multiplexadas pueden ser más interesantes en términos de coste frente a los encaminadores con buffer en la salida.

6.5 Evaluación de rendimiento.

A la vista de los resultados de la síntesis de los diferentes encaminadores, el siguiente paso es efectuar una comparativa de rendimiento entre ellos. Para efectuar esta tarea se ha recurrido al simulador *SICOSYS* (Ver Apéndice A).

Como en capítulos previos, el análisis de rendimiento se ha realizado desde dos puntos de vista: en primer lugar empleando únicamente cargas de trabajo sintéticas y enfocando la segunda parte del análisis a estudiar el comportamiento de cada uno de los encaminadores utilizando las cargas reales de nuestro banco de pruebas sobre una arquitectura ccNUMA.

6.5.1 Cargas Sintéticas.

En la Tabla 6-4 aparecen los resultados de latencia base para los diferentes encaminadores y tráficos sintéticos ya conocidos. Implícitamente, estos datos incorporan los resultados que se muestran en las Tablas 6-2 y 6-3.

Encaminador	Random	Matr-Tra	Perfec-Shu	Bit- Rever	Bimodal
InBuffer Deter.	103.2	108.3	103.3	109.5	118.9
InBuffer Adapt.	111.0	116.8	111.5	118.1	129.2
OutBuff Deter.	115.7	124.0	116.2	123.0	130.7
OutBuff Adap.	119.9	128.7	120.1	129.3	137.1
InBuffer Adap Muxed	120.9	129.9	121.2	130.4	138.3
InBuffer Adap Part Muxed	138.7	145.8	138.9	146.3	160.3

Tabla 6-4. Latencia base en nanosegundos para una carga de 0.05% con respecto a la bisección.

Por otro lado, en la Tabla 6-5 se muestra la carga máxima aceptada para cada uno de los encaminadores, estando expresada esta tanto en *phits* consumidos por ciclo de reloj como por nanosegundo. De esta forma son tenidos en cuenta las implicaciones tecnológicas de cada uno de los encaminadores.

Encaminador	Random	Matr-Tra	Perfec-Shu	Bit- Rever	Bimodal
InBuffer Deter.	11.50 (39.1)	3.92 (13.3)	5.27 (17.9)	3.58 (12.2)	7.33 (24.9)
InBuffer Adapt.	10.87 (39.9)	7.60 (27.9)	10.14 (37.2)	8.82 (32.4)	7.82 (28.7)
OutBuff Deter.	14.77 (49.5)	4.08 (14.5)	6.13 (22.3)	4.11 (13.7)	9.11 (30.54)
OutBuff Adap.	16.30 (56.9)	9.7 (34.0)	13.13 (45.8)	12.35 (43.1)	11.21 (39.1)
InBuffer Adap Muxed	10.83 (38.13)	8.46 (29.80)	6.46 (22.75)	8.68 (30.57)	8.34 (29.36)
InBuffer Adap Part Muxed	11.97 (43.93)	8.14 (29.88)	7.18 (26.36)	9.80 (36.0)	9.51 (34.93)

Tabla 6-5. *Throughput* máximo aceptado expresado en phits consumidos por nanosegundo (phits consumidos por ciclo).

Por último, en las Figuras 6-20, 6-21 y 6-22 se muestra la evolución de la latencia y *throughput* para algunos tráficos. Por simplicidad, solo se muestran los más significativos ya que en el resto de tráficos el comportamiento de las diferentes alternativas es bastante similar.

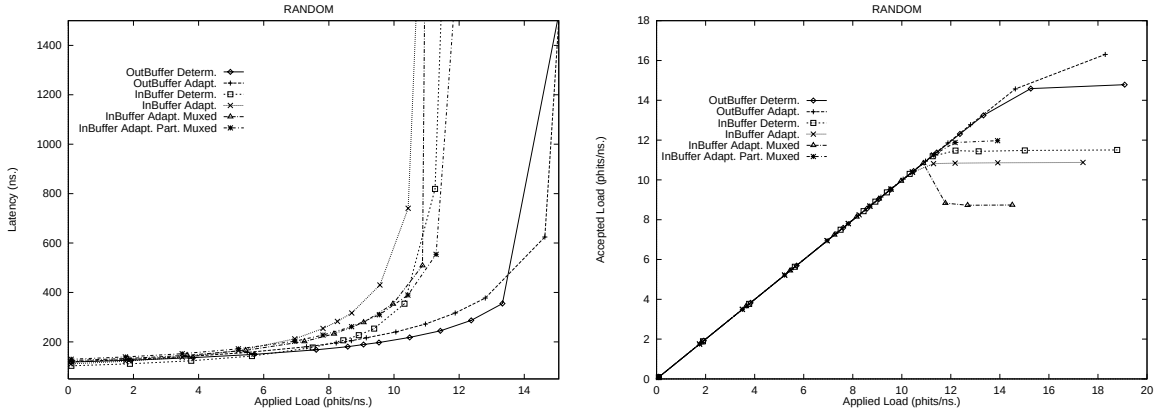

Figura 6-20. Evolución de la latencia y el throughput ofrecido en toro 8x8 con tráfico uniforme (L=10 phits)

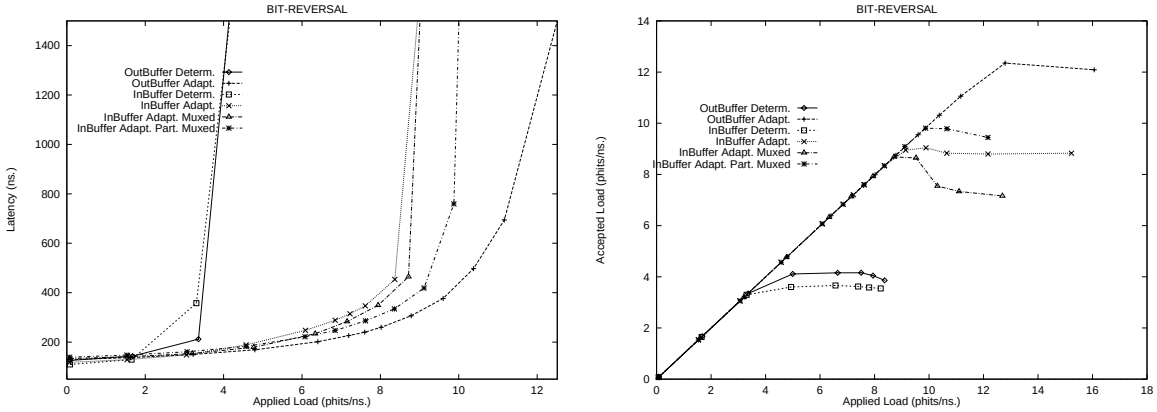

Figura 6-21. Evolución de la latencia y el throughput ofrecido en toro 8x8 con tráfico *Bit Reversal* (L=10 phits)

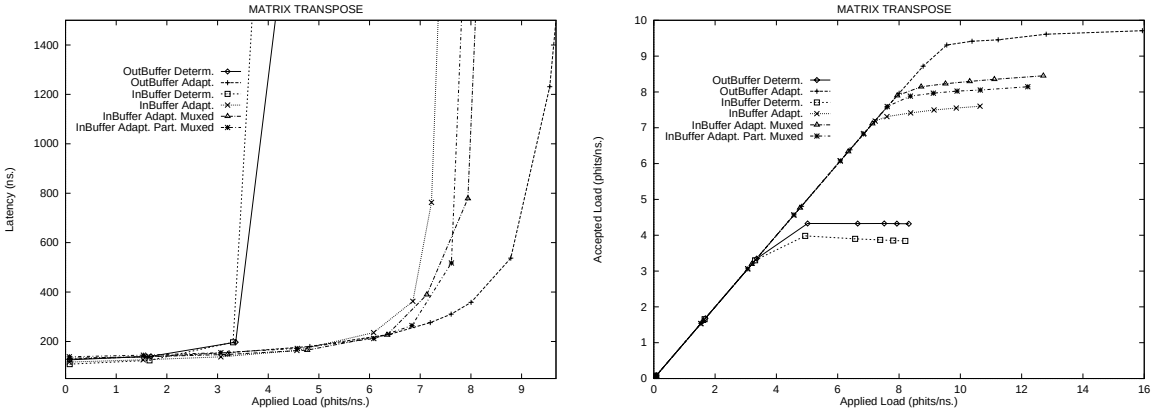

Figura 6-22. Evolución de la latencia y el throughput ofrecido en toro 8x8 con tráfico Matriz transpuesta (L=10 phits)

A la vista de estos resultados se puede concluir que el encaminador determinista con buffer en la salida permite alcanzar unos resultados en productividad muy discretos, salvo en patrones de tráfico uniforme. El rendimiento observado en los tráficos con patrón de destino no uniforme no justifica, en modo alguno, el coste adicional tanto en área que incorpora este encaminador (en torno a un 15% con respecto al encaminador adaptativo con buffers en la entrada y un 40% con respecto al encaminador determinista con buffers en la entrada) como en tiempo (pese a tener un tiempo de ciclo sensiblemente inferior al encaminador adaptativo con buffer en la entrada su pipeline es más largo). Los resultados muestran como el rendimiento alcanzado por este *router* para los tráficos uniformes supera notablemente el rendimiento ofrecido por el encaminador adaptativo con buffer en la entrada. Sin embargo, el rendimiento para los patrones de tráfico no uniformes se encuentra a medio camino entre los dos encaminadores con buffer en la entrada. Luego es claro, bajo estas condiciones de tráfico, que no es recomendable la utilización de este tipo de estructuras dado que, en general, el comportamiento de las aplicaciones reales es más próximo a patrones de destino no uniformes.

Por otro lado, el encaminador adaptativo con esquema de almacenamiento híbrido permite obtener unos resultados en *throughput,* frente al resto de alternativas que pueden calificarse de excelentes en todo el rango de cargas evaluadas. Estas ganancias llegan a ser de hasta un 250% con respecto al encaminador determinista con buffer en la entrada, un 230% con respecto al encaminador determinista con buffer en la salida y hasta un 40% con respecto al encaminador adaptativo con buffers en la entrada, sea o no multiplexado. Esta mejora se traslada en un incremento de coste, en área, que oscila entre un 90% con respecto al encaminador determinista con buffers en la entrada, un 50% con respecto al encaminador adaptativo con buffers en la entrada y un 20% con respecto al encaminador determinista con buffers en la salida. Además se introduce una degradación en la latencia base con respecto a los encaminadores con buffers en la entrada,

pero siempre está por debajo del 10%. A tenor de estos datos, puede decirse que el encaminador con una estructura de almacenamiento híbrido se trata de una implementación viable y que puede alcanzar unos rendimientos extraordinariamente altos en productividad con respecto al resto de alternativas.

Los datos obtenidos muestran que el binomio adaptatividad - buffer en la salida es una solución que permite obtener unos resultados altos en *throughput* con un incremento en coste (tanto espacial como temporal) razonable. La causa de este incremento en la zona de operación de la red se encuentra en que el problema de bloqueo en la cabeza y las paradas provocadas por el arbitraje interno limitan considerablemente la posible mejora de rendimiento introducida por la adaptatividad. Al eliminar este problema, el rendimiento de la red se incrementa notablemente. En el caso determinista, solo sucede esto para tráficos de tipo uniforme. En el resto de tráficos el rendimiento se aleja considerablemente del caso adaptativo con buffer en la salida. Como es lógico, esto sucede porque el mecanismo de encaminamiento determinista descompensa la utilización de los recursos de la red. Además, con este tipo de encaminamiento, la mayoría de los paquetes que podemos encontrar en una cola en un instante dado, en su mayoría se dirigen al mismo nodo destino, por lo que la evitación del *HLB* es irrelevante de cara al rendimiento de la red. Sin embargo, en el caso adaptativo al permitir que los paquetes se adecuen a las condiciones del tráfico esto no es cierto, resultando que la evitación del *HLB* permite mejorar considerablemente la productividad de la red.

La otras implementaciones orientadas a la evitación del *HLB,* como son los encaminadores adaptativos multiplexados, muestran mejora en términos de productividad, con respecto a su contrapartida no multiplexada, tan solo en algunos casos. En cuanto al rendimiento alcanzado por el encaminador completamente multiplexado, se aprecia, en algunos tráficos, una caída considerable en la carga aceptada por la red cuando la carga aplicada supera el límite de saturación. En casos como el del tráfico uniforme o el *bit-reversal*, la caída llega a ser de hasta un 20%. En el caso del encaminador parcialmente multiplexado, al no penalizar los canales deterministas, este tipo de efectos no aparecen de manera tan clara. De hecho, como era de esperar, en la mayoría de los casos este encaminador llega a superar el rendimiento alcanzado por el encaminador adaptativo con almacenamiento en la entrada y sin multiplexar. Notar que la pequeña diferencia observada entre los dos encaminadores multiplexados en el caso de el tráfico *matriz transpuesta,* está motivada por las diferencias en el periodo de reloj y no por las características estructurales, ya que, como se puede ver en la Tabla 6-5, el parcialmente multiplexado es mejor en términos de *phits* consumidos por ciclo de reloj. El único tráfico en el que el encaminador parcialmente multiplexado exhibe un comportamiento ligeramente peor que el no multiplexado es el tráfico *perfect-shuffle.* Aparentemente, las características de este tráfico (con muchos canales

alternativos entre origen y destino) hace que el realizar el doble arbitrio en los canales adaptativos limite sensiblemente la productividad de la red.

Por otro lado, en cuanto a la latencia se observa de forma clara un incremento de hasta un 20% para el completamente multiplexado y hasta un 25% para el parcialmente multiplexado con respecto al encaminador sin multiplexar. Los resultados de síntesis muestran que los dos encaminadores, aproximadamente, poseen el mismo coste en área de silicio. Cuando comparamos los resultados alcanzados por los encaminadores multiplexados frente al encaminador adaptativo con almacenamiento híbrido, queda claro que no llegan, en ningún, caso a alcanzar el *throughput* máximo característico de este último. De la misma forma, como se puede apreciar en los resultados expuestos previamente, la latencia promedio exhibida por los mensajes, a niveles bajos de carga, es siempre inferior para el encaminador adaptativo con bufferes en la salida. En este caso, las memorias multipuerto resultan sensiblemente menos costosas, en términos de tiempo, que las RDU multiplexadas (tan solo 0.03 ns) lo que, al final, se traduce en una ligerísima mejora en latencia (inferior al 1%) para el encaminador completamente multiplexado y en un 5% para el parcialmente multiplexado (recordar que en este último caso el módulo crítico sigue siendo el crossbar).

6.5.2 Cargas Reales.

De la misma forma que en capítulos anteriores, en este estudio emplearemos el simulador conducido por ejecución descrito en el Apéndice A, utilizando el banco de pruebas descrito en la Sección 2.3.3.

La configuración del sistema es esencialmente la misma que la empleada en capítulos precedentes. Bajo estas condiciones, cada procesador del sistema posee 16Kb de capacidad en la *cache* de primer nivel, 64Kb de capacidad en la *cache* de segundo nivel y el tamaño de la línea de *cache* en cualquiera de los dos niveles es de 32 bytes. El tamaño básico de los comandos que maneja la red es de 8 bytes. De acuerdo con esto, y puesto que un *phit* incluye 4bytes, el tamaño de los paquetes asociados a comandos básicos es de 2 *phits* y los asociados a datos es de 10 *phits*. El resto de parámetros del sistema se mantienen con respecto a la configuración empleada en capítulos previos. Para una descripción más detallada del mismo ver la Tabla 3-9 en la página 126.

Dado que estamos empleando una tecnología de 0.35 μm en el diseño *hardware* de los encaminadores, lo que representa una reducción notable frente a la tecnología empleada en capítulos anteriores, hemos considerado oportuno escalar la velocidad del procesador de la misma forma. Por ello, siguiendo la misma relación en la variación de la longitud del canal-velocidad del pro-

cesador, emplearemos procesadores que operan a 1.2 GHz. Estas cifras no son en absoluto irreales puesto que existen varios procesadores, de próxima aparición en el mercado, que van a operar a estas velocidades de reloj. Entre ellos, podemos citar el próximo Alpha 21364 [63]. Bajo estas circunstancias, y teniendo en cuenta los resultados de la síntesis de cada uno de los encaminadores, las relaciones de velocidad procesador/red, para cada uno de los casos analizados, son las indicadas en la Tabla 6-6.

Encaminador	Frecuencia de reloj del Encaminador	Velocidad relativa procesador/ red
BDOROB[a]	298 Mhz	4.01
BADAOB[b]	287 Mhz	4.19
BDORIB[c]	295 Mhz	4.08
BADAIB[d]	272 MHz	4.40
BADACM[e]	272 Mhz	4.40
BADACFM[f]	284 Mhz	4.22

a. Encaminador con buffer en la salida determinista

b. Encaminador con buffer en la salida adaptativo

c. Encaminador con buffer en la entrada determinista

d. Encaminador con buffer en la entrada adaptativo

e. Encaminador con buffer en la entrada adaptativo y completamente multiplexado.

f. Encaminador con buffer en la entrada adaptativo y parcialmente multiplexado.

Tabla 6-6. Velocidades relativas red/procesador en cada caso.

Bajo estas condiciones hemos evaluado el rendimiento ofrecido por cada encaminador y aplicación considerada, empleando en este análisis una configuración físicamente aislada para las redes de peticiones y respuestas. El número de nodos considerado, en todos los casos, ha sido 64.

6.5.2.1 FFT.

El tamaño de problema empleado es el establecido por defecto en la aplicación, es decir, consiste en 64K dobles complejos. Para el caso de esta aplicación y con la configuración expuesta previamente, los tiempos de ejecución con cada uno de los encaminadores alternativos son los indicados en la Figura 6-23. En primer lugar, las diferencias entre los diferentes encaminadores son bastante exiguas. De cualquier manera, el encaminador adaptativo con buffer en la salida muestra el mejor rendimiento de todos. Pese a tener una latencia base superior, en la mayoría de los casos, al resto de encaminadores, llega a mejorar el tiempo de ejecución hasta en un 7% con respecto a la peor de las alternativas. Por otro lado, los dos encaminadores con buffer en la entrada tienen un comportamiento muy similar, siendo en torno a un 5% más lenta la ejecución

de la aplicación en estos casos. En el caso del encaminador determinista con buffer en la salida, su bajo rendimiento estructural (recordar que con tráficos sintéticos uniformes se encontraba a medio camino entre los dos encaminadores con buffer en la entrada) y superior tiempo de ciclo hace que el rendimiento del sistema sea el peor de todos. Por otro lado, el empleo de múltiples canales virtuales por línea física con crossbars multiplexados, permite reducir sensiblemente el tiempo de ejecución de la aplicación con respecto a la estructura adaptativa con buffer en la entrada y con un solo canal virtual adaptativo. De cualquier manera, la mejora introducida no llega a ser tan acusada como en el caso adaptativo con buffer en la salida. Como se puede apreciar, el efecto de un tiempo de ciclo más pequeño para el encaminador completamente multiplexado (canales virtuales adaptativos y determinista multiplexados) permite una ligerísima mejora en el tiempo de ejecución. Pese al mejor tiempo de ciclo, la penalización que sufría en este caso el canal determinista tiende a limitar la ganancia final.

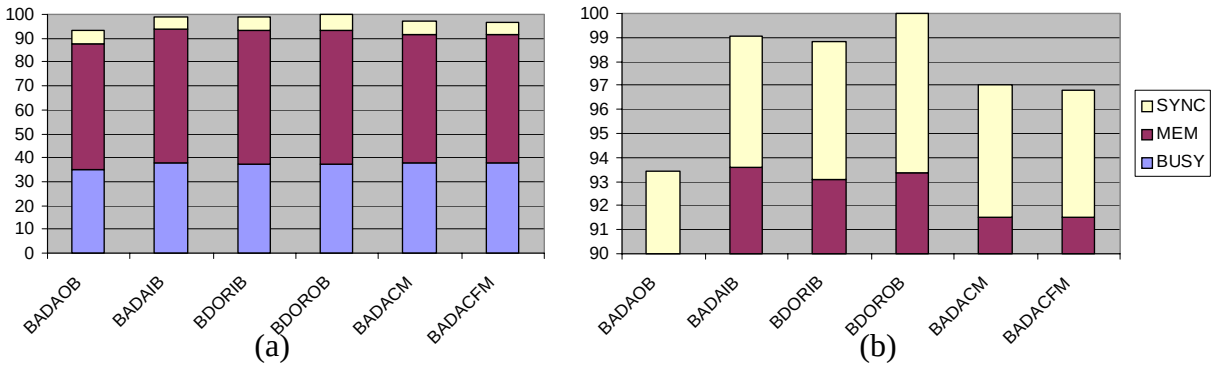

Figura 6-23. (a)Tiempo de ejecución normalizado para FFT con 64 procesadores, (b) Detalle.

A continuación pasaremos a analizar cual es el comportamiento de la aplicación a medida que avanza su ejecución. En la Figura 6-24 se representa este comportamiento (en términos estructurales) para los encaminadores más significativos y en el caso de la red de respuestas. Con el fin de que las figuras sean claras, en esta aplicación y sucesivas, solo representaremos el comportamiento para alguno de los encaminadores considerados. Como se puede apreciar, en este caso solamente representamos el comportamiento para los dos encaminadores deterministas y el adaptativo con buffer en la salida. Hemos optado por elegir el determinista con buffer en la entrada como muestra significativa de todos los encaminadores que se sitúan en la zona media del tiempo de ejecución para la aplicación. Por otro lado, en la Figura 6-25 se muestran el comportamiento de estos mismos encaminadores en términos reales (normalizando el eje de ordenadas en ambas figuras en ciclos de procesador).

Como puede apreciarse la demanda ejercida por la aplicación en la mayoría de los casos no es demasiado elevada, no superando en ningún caso los 30 *phits* inyectados por ciclo de red siendo

claramente menos elevada que la mostrada por esta misma aplicación en pruebas previas. Por ello, la aplicación nunca llega a hacer que se supere el punto de saturación de la red. Esto es acorde con las pequeñas diferencias observadas en el tiempo de ejecución observado en cada caso. Pese a estos datos, el mejor rendimiento estructural del encaminador adaptativo con buffer en la salida permite reducir la latencia promedio de los mensajes en las zonas altas de carga lo que permite compensar la superior latencia y tiempo de ciclo que tiene con respecto al encaminador determinista con buffer en la entrada. Sin embargo, el encaminador determinista con buffer en la salida no logra compensar la penalización que tiene en su tiempo de ciclo con respecto al determinista con buffer en la entrada, lo que definitivamente hace que sea el que peor tiempo de ejecución alcance.

Por otro lado, como se puede apreciar al comparar la Figura 6-24 y la Figura 6-25, el comportamiento es muy similar. A diferencia de capítulos precedentes, el tiempo de ciclo de cada uno de los encaminadores considerado es muy próximo, por lo que tiene muy poca repercusión en la aplicación. Es por ello que no se observen diferencias tan acusadas como las que aparecían en el capítulo anterior.

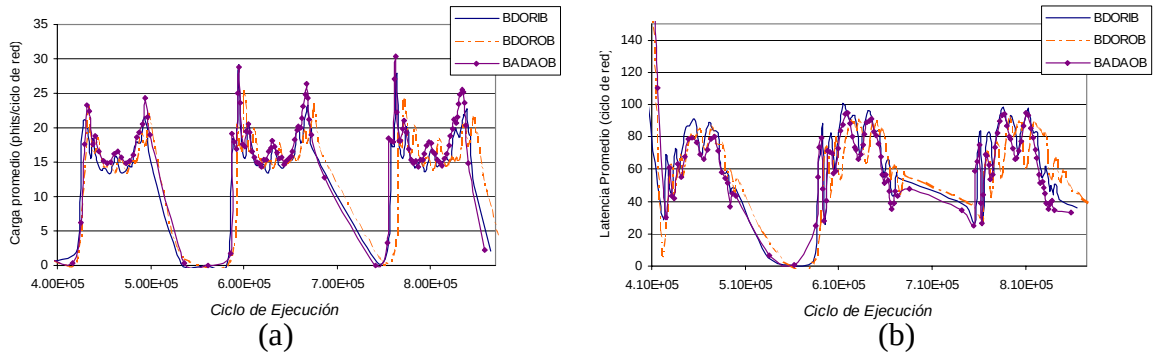

Figura 6-24. Traza de ejecución de la red de respuestas (*en términos estructurales*) para FFT con 64 procesadores, (a) Carga Promedio, (b) Latencia Promedio.

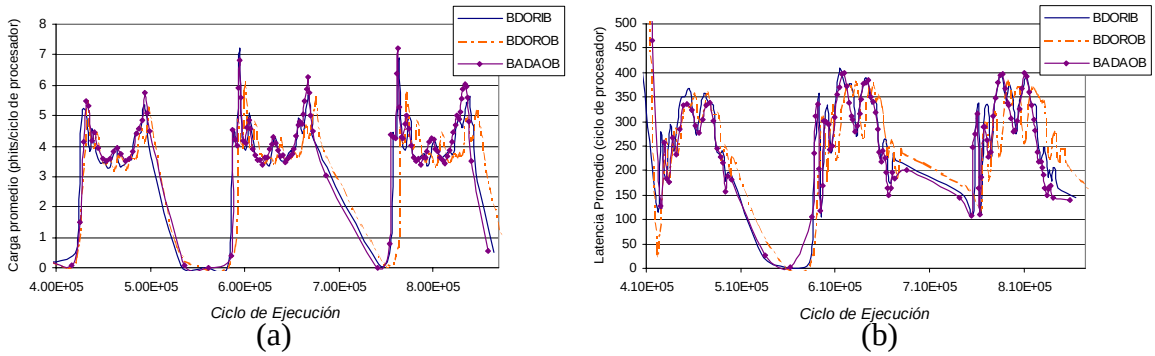

Figura 6-25. Traza de ejecución de la red de respuestas (*en términos reales*) para FFT con 64 procesadores, (a) Carga Promedio, (b) Latencia Promedio.

6.5.2.2 Radix.

Dado que únicamente estamos evaluando redes bidimensionales, el tamaño de problema que emplearemos en este análisis para la aplicación será el establecido por defecto. Bajo estas circunstancias, será preciso ordenar 1M de claves enteras con radio máximo de 1M. Los resultados en tiempo de ejecución de esta aplicación, para cada una de las estructuras consideradas, son los mostrados en la Figura 6-26. En este caso el encaminador que peor rendimiento ofrece es el adaptativo con buffer en la entrada y los mejores son el adaptativo con buffer en la salida y adaptativo con crossbar multiplexado completamente. Como se ve, las diferencias entre los encaminadores que mejor rendimiento ofrecen son bastante reducidas llegando a ser la máxima discrepancia entre ellos de algo más de un 1%.

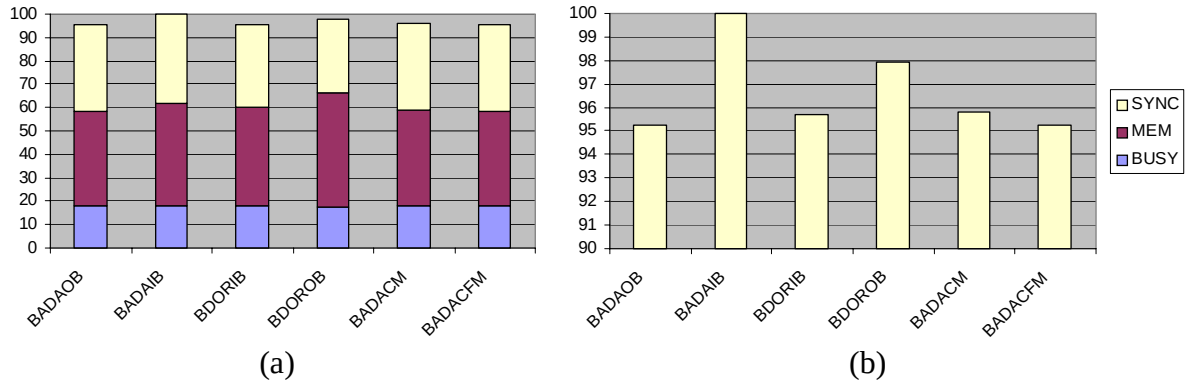

Figura 6-26. (a)Tiempo de ejecución normalizado para Radix con 64 procesadores, (b) Detalle.

Como se puede apreciar, en este caso el encaminador determinista con buffer en la salida logra superar al encaminador adaptativo con buffer en la entrada. Es preciso señalar que la mayor parte del tráfico generado en esta aplicación corresponde al intercambio de claves y puesto que están son generadas y asignadas de forma aleatoria, el tráfico asociado será prácticamente uniforme. Si observamos los datos que aparecen en la Tabla 6-5, podemos ver como el encaminador adaptativo posee menor *throughput* sostenido que el encaminador determinista con buffer en la salida bajo condiciones de tráfico uniforme. Sin duda, las características del tráfico de esta aplicación justifican la diferencia observada en el tiempo de ejecución.

Por otro lado, para entender que está sucediendo en la red de interconexión para que la mayoría de los encaminadores muestren un tiempo de ejecución muy similar, en las Figuras 6-27 y 6-28 se ha representado la evolución de la carga aplicada sobre la red y el retraso sufrido por los mensajes para tres encaminadores significativos tanto en términos estructurales como reales. En este caso, se ha considerado oportuno analizar el comportamiento de los dos encaminadores con buffer en la entrada (adaptativo y determinista) y el encaminador con buffer en la salida adaptativo. Como se puede ver con claridad, la carga aceptada por la red, en promedio, es substan-

cialmente más elevada que la observada en la aplicación anterior, llegando a superar en algunos casos los 50 *phits* por ciclo de red. Esta cifra representa una presión, más que considerable, por parte del sistema hacia la red. De la misma forma, se observa como el encaminador adaptativo con buffer en la salida muestra su indudable mejor rendimiento a nivel estructural. Las diferencias con respecto a los encaminadores con buffer en la entrada llegan a ser de casi 10 *phits* por ciclo de red, lo que representa una diferencia considerable. De la misma forma, la gráfica de latencia promedio en términos estructurales muestra claramente como los dos encaminadores con buffer en la entrada llegan a sobrepasar el punto de saturación de la red y sin embargo el encaminador con buffer en la salida esta aún lejos de dicho límite. Por otro lado, el encaminador adaptativo con buffer en la entrada llega a superar ligeramente el rendimiento que logra alcanzar el encaminador determinista.

Sin embargo, las diferencias observadas en las zonas altas de carga de la aplicación no parecen acordes con las diferencias en el tiempo de ejecución. De nuevo, aparecen zonas de tráfico local en las que los encaminadores con menor latencia tienden a superar a los demás. Como sabemos, en Radix está presente esta clase tráfico como consecuencia de las pausas de sincronización una vez superada la fase de cálculo del prefijo de paralelización. En concreto, la zona marcada como *A* en la Figura 6-27 corresponde, prácticamente en su totalidad, a este tráfico. En este caso, se observa como, en términos estructurales, el encaminador adaptativo con buffer en la salida tiende a perder lo ganado en las zonas altas al ser la latencia promedio la mayor de todas. Pese a que las diferencias son pequeñas, el porcentaje del tiempo de ejecución correspondiente a esta fase es elevado. Además, el porcentaje de tiempo de las zonas de baja carga (siempre por debajo de 15 *phits* por ciclo de red), también influye en mermar considerablemente las posibilidades, en términos globales, del encaminador en esta aplicación concreta.

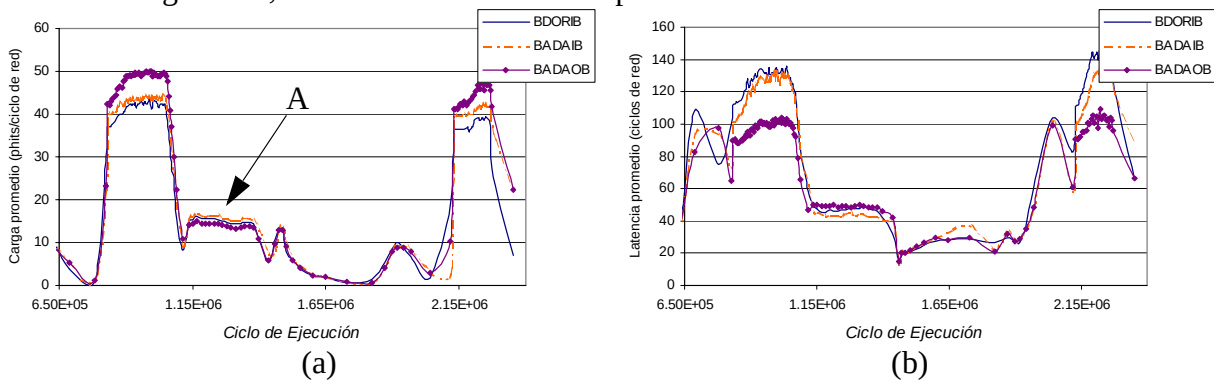

Figura 6-27. Traza de ejecución de la red de respuestas (*en términos estructurales*) para Radix con 64 procesadores, (a) Carga Promedio, (b) Latencia Promedio.

Por otro lado, parece extraño comprobar como el encaminador adaptativo con buffer en la entrada muestra el peor tiempo de ejecución y sin embargo logra superar claramente en latencia

y *throughput* al encaminador determinista con buffer en la entrada. Para aclarar este aspecto es preciso atender a la representación en términos reales de carga y latencia observada en la red. Como podemos ver, al tener en cuenta el superior tiempo de ciclo del encaminador adaptativo, el comportamiento que aparece en la figura anterior tiende invertirse, siendo realmente el encaminador adaptativo el que menos *throughput* es capaz de manejar y además mayor latencia tiene a lo largo de toda la ejecución de la aplicación. Por ello, es completamente lógico que la aplicación tenga un tiempo de ejecución más alto que en el caso del encaminador determinista. Aparentemente esto entra en contradicción con el comportamiento observado para esta aplicación en el Capítulo 3. Sin embargo, como es conocido, el rendimiento ofrecido por el encaminamiento adaptativo con tráfico uniforme no dista demasiado del alcanzado empleando encaminamiento determinista. Además, en este caso estamos empleando mensajes de longitud 10 *phits* en lugar de 20, lo que también tiende a favorecer al encaminador determinista frente al adaptativo. De esta forma se puede concluir que los datos no son en absoluto incoherentes sino consecuencia de la configuración empleada para el router y en especial a la anchura de los canales de comunicación.

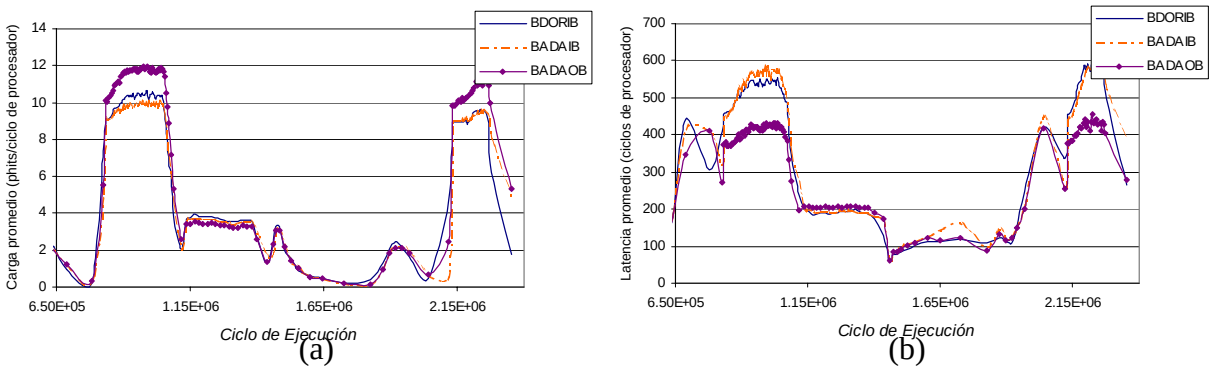

Figura 6-28. Traza de ejecución de la red de respuestas (*en términos reales*) para Radix con 64 procesadores, (a) Carga Promedio, (b) Latencia Promedio.

6.5.2.3 LU.

Por último, para finalizar el análisis de rendimiento sobre cargas reales, pasaremos a evaluar como se comporta cada encaminador considerado en el caso de LU. De la misma forma que en capítulos precedentes, a la hora de ejecutar la aplicación se ha optado por emplear un tamaño de problema inferior al marcado por defecto. Dada la cantidad de arquitecturas alternativas a evaluar y lo demandante de esta aplicación se han empleado matrices de 256x256 elementos y con un tamaño de bloque de 16x16.

El tiempo de ejecución observado para esta aplicación, con cada encaminador, se muestra en la Figura 6-29. En este caso, las diferencias entre los distintos encaminadores tienden a ser mayores que en el caso de las aplicaciones previas. Como se puede apreciar, de nuevo, el encamina-

dor adaptativo con buffer en la salida ofrece el mejor rendimiento, llegando a reducir hasta en un 8% el tiempo de ejecución obtenido con el encaminador determinista con buffer en la salida y en un 6-7% el tiempo de ejecución de los encaminadores con buffer en la entrada. Por otro lado, los encaminadores multiplexados se encuentran a medio camino de estas cifras, llegando a mostrar la aplicación un tiempo de ejecución entre un 1.5% y un 2% mayor que en el caso del encaminador adaptativo con buffer en la salida.

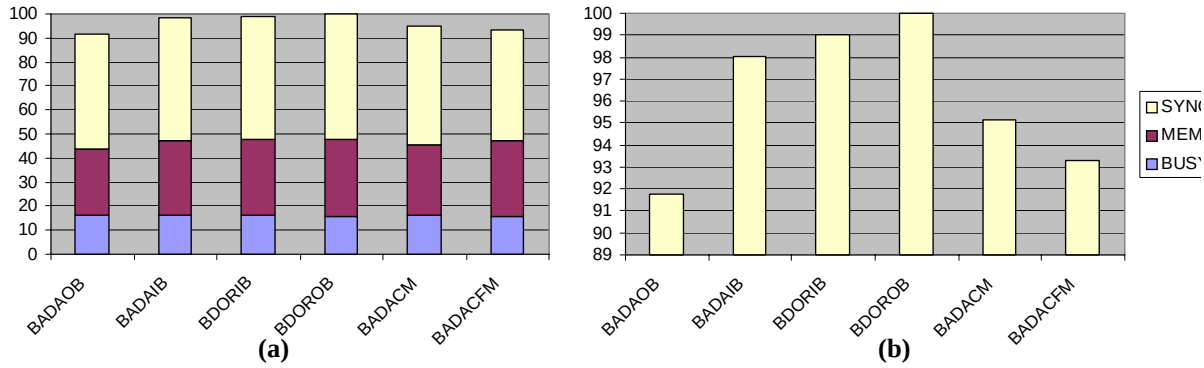

Figura 6-29. (a)Tiempo de ejecución normalizado para LU con 64 procesadores, (b) Detalle.

Como ya hemos comentado en capítulos anteriores, esta aplicación no se caracteriza por someter a la red a una elevada carga, sino más bien su característica fundamental es que da lugar a puntos calientes de carga en los nodos que mantienen los bloques de la diagonal de la matriz a factorizar. Por ello, el encaminamiento adaptativo logra sacar una ventaja clara sobre el determinista ya que permite evitar, en gran parte, la presencia de los puntos de elevada carga y minimizar su impacto en la latencia de los mensajes. Como se aprecia, en este sentido el encaminador adaptativo con buffer en la entrada logra mejorar el tiempo de ejecución con respecto a cualquiera de los deterministas. Las diferencias no son tan altas como en capítulos previos por la misma razón que en Radix, es decir, la menor longitud de los mensajes reduce sensiblemente las ganancias posibles de la adaptatividad. Por otro lado, es claro como el encaminamiento adaptativo con buffer en la salida consigue mejorar el tiempo de ejecución. Sin duda, en esta aplicación la contención causada por el *HLB* limita el rendimiento de la red. Como podemos observar, la diferencia entre los encaminadores que incorporan encaminamiento adaptativo y mecanismos de evitación del *HLB* logran, con diferencia, los mejores tiempos de ejecución. La evitación de los puntos calientes de la red, cosa que no puede hacer el encaminador determinista con buffer en la salida, consigue mejorar considerablemente el tiempo de ejecución de la aplicación.

Por otro lado, en la Figura 6-30 se muestra la evolución en el estado de la red de respuestas a medida que la aplicación es ejecutada. En este caso, con el fin de mantener la claridad, se han

representado únicamente la evolución de la red para el caso los encaminadores con buffer en la entrada y en la salida adaptativo. Por otro lado, únicamente reflejaremos el rendimiento en términos estructurales, ya que es suficiente para apreciar los efectos más interesantes que ocurren en la red. Como se puede comprobar, el nivel de la carga ejercida por el sistema sobre la red es bastante bajo. En el peor de los casos, apenas si alcanza los 10 *phits* por ciclo de red. Sin embargo, la latencia de los mensajes es elevada, especialmente si la comparamos con el comportamiento mostrado por las aplicaciones anteriores. Para apreciar mejor las diferencias, y dado lo extenso de la ejecución de la aplicación, en la Figura 6-31 se ha hecho una ampliación de la fase inicial de la aplicación. Como puede verse en el caso de la latencia, las diferencias son acusadas, llegando a mostrar el encaminador determinista con buffer en la entrada latencias superiores en casi 50 ciclos de red sobre el encaminador adaptativo con buffer en la salida. Aunque las diferencias con el encaminador adaptativo con buffer en la entrada no sean tan acusadas, si son claras. En definitiva, estas diferencias son muestra inequívoca de que la adaptatividad permite minimizar la influencia de la existencia de zonas de elevada carga en el sistema. Esto junto con la evitación de *HLB* permite disminuir considerablemente el retraso de los mensajes lo que, en definitiva, permite reducir el tiempo de ejecución de la aplicación.

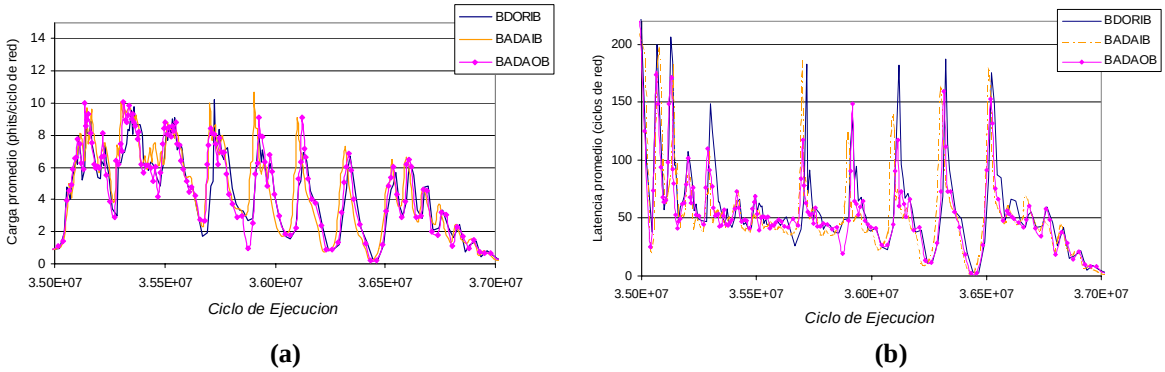

Figura 6-30. Traza de ejecución en la red de respuestas (en *términos estructurales)* para LU con 64 procesadores, (a) Carga promedio, (b) latencia promedio.

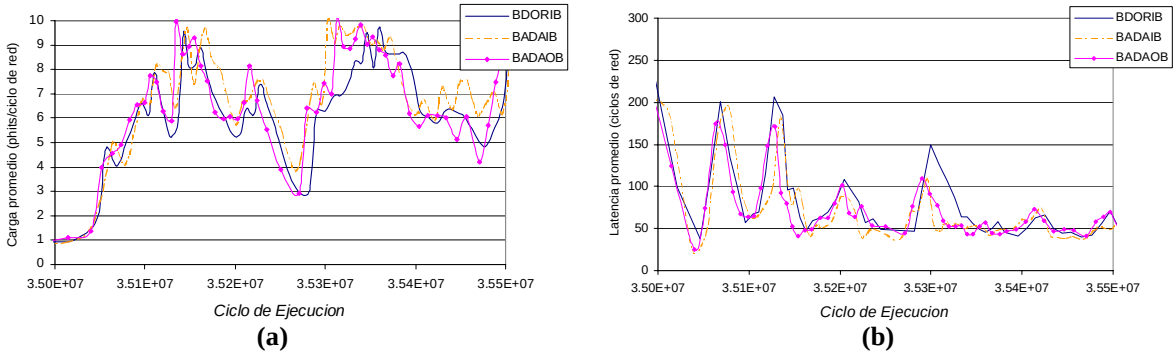

Figura 6-31. Detalle de la traza de ejecución para LU con 64 procesadores, (a) Carga promedio, (b) latencia promedio.

6.6 Conclusiones.

Hemos analizado el problema del *HLB* y de qué modo afecta al rendimiento de las redes de interconexión en términos de máximo *throughput* alcanzable. En esta línea, se han estudiado combinaciones entre estructuras, libres o no de este efecto, con mecanismos de encaminamiento adaptativos y deterministas. Las principales conclusiones extraídas son las siguientes:

- Emplear arquitecturas con espacio de almacenamiento en la salida y con encaminamiento determinista, no es rentable. En este caso, solamente los tráficos con patrón de destinos uniforme sacan una ligera ventaja a otras alternativas con colas de almacenamiento simples en la entrada, incluso con adaptatividad. Este hecho, junto con el mayor coste en área de los primeros, hace que se conviertan en una alternativa poco recomendable.
- El empleo conjunto de arquitecturas con espacio de almacenamiento en la salida y mecanismos de encaminamiento adaptativo es muy rentable. Se logra mejorar de forma drástica la productividad de la red sin introducir un coste excesivamente alto. Las tecnologías de implementación actuales, como la empleada en este capítulo, hacen que sea necesario tener en cuenta este tipo de estructuras a la hora de diseñar el encaminador. Una de las razones que fundamentan la viabilidad de estos encaminadores es la posibilidad de implementar las memorias multipuerto en *hardware* sin un coste desmesurado. Otro hecho que facilita la viabilidad final de la implementación es reducir, en la medida de lo posible, los requerimientos en número de puertos de entrada para estas memorias. Nosotros hemos logrado este objetivo simplificando en lo posible el encaminador al emplear un esquema de almacenamiento híbrido.
- Hemos probado estructuras que evitan los efectos del *HLB* empleando un número elevado de canales virtuales. Es una estrategia empleada habitualmente pero, como muestran los resultados obtenidos, el rendimiento alcanzado por los encaminadores adaptativos que incorporan este mecanismo no llega a alcanzar, en ningún caso, el obtenido por el encaminador adaptativo con almacenamiento en la salida. La razón básica de este hecho es que los efectos que introduce la adaptatividad en la red de interconexión hace que las colisiones de los paquetes, a la hora de acceder a los recursos proporcionados por el encaminador, limiten la productividad máxima de la red. En el caso de almacenamiento en la entrada multiplexado para los canales adaptativos, las colisiones se siguen produciendo tanto en el arbitraje de las *RDU* como en el crossbar. Sin embargo, en la estructura con buffer en la salida, los datos pueden avanzar hasta el buffer sin ningún tipo de colisión con otros paquetes. En el primer caso, las colisiones son determinantes y a la postre son la consecuencia de la caída

de rendimiento. Por lo tanto, el buen rendimiento alcanzado por el encaminador con buffer en la salida se debe a un efecto combinado de la eliminación del *HLB* y de la no existencia de ningún bloqueo en los canales adaptativos que introduzca contención.