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 4

Efectos de la Implementación: un Mecanismo Simple de Arbitraje Distribuido.

Una de las principales causas del mayor coste temporal de los encaminadores adaptativos frente a los deterministas es el incremento de complejidad en la gestión hardware del encaminamiento adaptativo de los paquetes. En este capítulo abordaremos el análisis de este tipo de aspectos para el caso del encaminador propuesto en el capítulo precedente y nos centraremos en consideraciones relativas a su implementación. Concretamente, probaremos las ventajas de una nueva propuesta de arbitraje frente a otras soluciones estándar.

4.1 Introducción.

El empleo de mecanismos de encaminamiento adaptativo, como ha quedado reflejado en el capítulo anterior, implica un incremento en el coste hardware del encaminador con respecto a encaminadores deterministas. El aumento se produce tanto en el área como en la velocidad final del router. La causas de la disminución de velocidad y aumento de área son el incremento en el número de canales virtuales a considerar por línea física para asegurar que la red es libre de interbloqueos y el aumento de complejidad que implica el algoritmo de encaminamiento en sí. Hemos planteado una nueva metodología que permite reducir, a tan sólo dos, el número de canales virtuales requeridos para asegurar que la red es libre de interbloqueos. Sin embargo, es necesario profundizar en el análisis de cómo puede afectar el modo de realizar el arbitraje interno del encaminador, tanto desde el punto de vista de rendimiento como de coste. Generalmente, este aspecto ha sido menospreciado en diversos estudios sobre redes de interconexión, pero nosotros lo consideramos de gran importancia y por lo tanto, es necesaria su evaluación. De cualquier forma, algunos trabajos como [62], indican de qué manera el árbitro o planificador puede ser uno de los elementos más críticos del encaminador.

En el caso de los encaminadores deterministas, generalmente, los únicos conflictos que pueden existir son los relacionados con la asignación de los puertos de salida. En este caso, el árbitro ha de evitar que ningún puerto de salida sea concedido a más de un canal de entrada simultáneamente. Estos conflictos se resuelven con un arbitraje entre todos los canales de entrada que pueden acceder a un puerto de salida dado, es decir, cada puerto de salida del router tiene asociado un árbitro independiente que impide que el mismo recurso sea concedido de forma simultánea a dos canales de entrada diferentes.

Por otra parte, en un router con encaminamiento adaptativo, la complejidad del arbitraje interno del encaminador es superior, ya que cada canal de entrada puede solicitar simultáneamente varios canales de salida. Es por ello necesario incorporar un arbitraje que elimine la posibilidad de que se conceda más de un recurso y que logre maximizar el número posible de asignaciones lo más rápidamente posible y con el menor coste. Como es evidente, cuanto mayor sea el número de canales de entrada a los que se puede atender simultáneamente concediendo un canal de salida válido, mayor será el rendimiento del encaminador y la productividad de la red. Para abordar este problema existen numerosas soluciones de diferente complejidad. Entre ellas, podemos señalar las siguientes: a) Añadir además del arbitraje entre los canales de entrada un arbitraje entre todos los puertos de salida del encaminador. A grandes rasgos, esto equivale a realizar dos arbitrajes consecutivos. b) Obligar a que las peticiones de cada canal de entrada hacia los canales de salida correspondientes se realicen de manera secuencial. y c) Atender úni-

camente a uno de los canales de entrada por ciclo de reloj y evaluar simultáneamente cual es la viabilidad de todas las salidas solicitadas.

En este capítulo se analizará el impacto sobre el rendimiento de cada una de las alternativas introducidas previamente. Concretamente, las alternativas de arbitraje analizadas son las siguientes:

- *Output Arbiter Crossbar* (OAC). Permite que todos los canales de entrada puedan solicitar de forma concurrente un único canal de salida de entre todos los posibles. Una unidad de decisión de routing externa al arbitro es la encargada de aplicar la función de selección así como de recorrer secuencialmente todos los canales de salida, hasta lograr la concesión de uno válido.
- *Sequential Input Crossbar* (SIC). En este caso, solamente un canal de entrada es atendido por ciclo de reloj, permitiendo que este canal de entrada solicite simultáneamente todas las salidas deseables. Esta forma de arbitrar entre las posibles entradas es similar conceptualmente a la propuesta empleada en [112].
- *Symmetric Arbiter Crossbar* (SAC). Contempla que todos los canales de entrada pueden solicitar de forma concurrente todas las posibles salidas que acerquen el paquete a destino siendo todas ellas atendidas simultáneamente. Esto propuesta esta basada en [125].

Básicamente, la opción SAC va a poseer un rendimiento en *throughput* más alto, en términos estructurales, que el resto de propuestas. Sin embargo, se puede anticipar que el coste asociado a este tipo de arbitraje va a penalizar el tiempo de ciclo del router completo. La estrategia OAC, básicamente, implica los mismos costes que los de un encaminador determinista, ya que el arbitraje se realiza también a nivel de puerto de salida. La propuesta SIC, intenta simplificar el coste de SAC, partiendo de la misma idea pero eliminando la concurrencia entre peticiones desde distintos canales de entrada, basándose en que es poco probable que varios canales de entrada soliciten en el mismo ciclo de reloj un puerto de salida.

Al hora de analizar cada propuesta, se ha llevado a cabo una implementación *hardware* para cada una de las alternativas, así como una evaluación de rendimiento de las diferentes propuestas con el algoritmo adaptativo propuesto en el capítulo anterior para el caso de redes *k-ary 2-cube* y *k-ary 3-cube*.

4.2 Estructuras Consideradas.

A continuación pasaremos a describir, de forma detallada, el modo de operación de cada una de las estrategias de arbitrio citadas en el punto anterior.

4.2.1 Router Basado en la Estructura *OAC.*

Esta estructura permite que todos los canales de entrada puedan solicitar de forma concurrente un único canal de salida de todos los posibles. Su desarrollo surge como una evolución de los encaminadores deterministas. En este tipo de routers, cada uno de los posibles canales de entrada puede solicitar solamente un canal de salida para el paquete almacenado en la cabeza de cada una de las colas de entrada. Bajo estas circunstancias, las peticiones que se realizan a los puertos de salida son independientes y no se puede dar la concesión múltiple de una misma salida. Esto permite que los mecanismos de petición y control funcionen de modo concurrente para cada uno de los canales a arbitrar. La estrategia coincide con la empleada en el *Bubble Router* (Ver Sección 3.4.1).

La complejidad añadida en el router OAC a la hora de incorporar la adaptatividad se encuentra en la unidad de decisión de routing (RDU) (ver Figura 4-1). En este caso, las peticiones no son simples, ya que cada canal de entrada puede solicitar varios canales de salida. Para resolver este problema se ha recurrido a secuencializar las peticiones. Es decir, la RDU solicita secuencialmente al crossbar cada uno de los posibles canales de salida. De esta manera, el resto de la estructura del encaminador sigue siendo como la de un router determinista, salvo la unidad de decisión de routing, que es bastante más compleja. Aún así, el incremento de complejidad de la RDU no va a afectar a la frecuencia de reloj del encaminador.

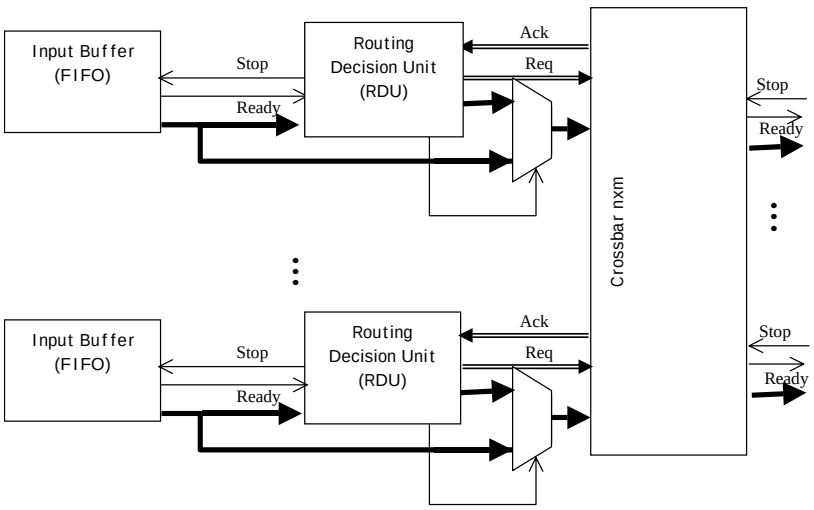


Figura 4-1. Estructura parcial del router para la propuesta OAC.

En este caso, el papel que juega el crossbar es el de actuar de interface entre el estado de los nodos vecinos (que es imprescindible conocer para aplicar la adaptatividad) y el proceso de petición de recursos. Además, es el encargado facilitar a la RDU el estado de ocupación de los puertos de salida locales (existe o no una entrada utilizando actualmente el recurso solicitado), evitando así posibles colisiones. Bajo estas condiciones, es este modulo el que tomará las decisiones sobre cómo y a quién se asigna cada uno de los canales de salida. Bajo este punto de vista, el árbitro del encaminador está integrado dentro de las unidades de control del crossbar.

En cuanto a la estructura interna del crossbar, dado que cada canal de entrada puede solicitar únicamente una salida, es posible subdividir el crossbar *nxm* en *m* subscrossbars *nx1* completamente independientes. De esta manera, el tiempo de ciclo del crossbar es únicamente dependiente del número de canales de entrada. En la Figura 4-2 se puede apreciar la estructura básica de este módulo.

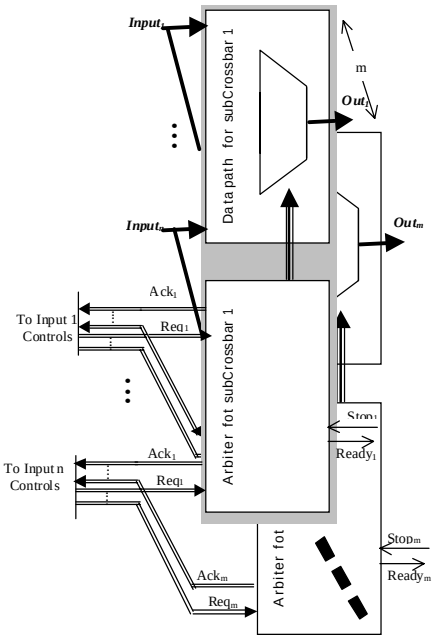


Figura 4-2. Estructura interna del crossbar OAC.

Los componentes de cada subcrossbar son el camino de datos y la unidad de control. La función del camino de datos es multiplexar todas las entradas del crossbar y propagar hasta la salida aquella entrada que determine el árbitro. La unidad de control es la encargada de arbitrar entre los canales de entrada que soliciten el canal de salida asociado a cada subcrossbar. La política de asignación de recursos empleada es de tipo *Round-Robin*, evitando de esta manera la aparición de problemas de inanición. De acuerdo con este criterio, el árbitro decide, en función del

estado de ocupación del puerto de salida y de la posición del *token*, cual es la entrada a la que se le concede el puerto de salida asociado.

La sencillez de esta estructura facilita que el arbitraje y transmisión del primer *phit* del mensaje se pueda realizar en un solo ciclo. Para poder implementar esta mejora es preciso que el dato se registre en el camino de datos del crossbar en el mismo ciclo que se lanza la petición desde el canal de entrada. En nuestro caso, la *RDU*, a la vez que envía la petición al árbitro del crossbar, transmite la cabecera del mensaje actualizada que corresponde a la dimensión del canal de salida solicitado por la línea de datos. De esta forma, es posible hacer operar el *Round-Robin* y el camino de datos en el mismo ciclo, sin penalizar en exceso el tiempo de ciclo del encaminador. Cuando el arbitraje determina que el puerto solicitado es accesible, en el mismo ciclo establece la conexión entre el registro que mantiene la cabeza correcta en el camino de datos y la salida. Por lo tanto, en el siguiente ciclo el dato estará listo para ser retransmitido hacia el encaminador vecino. La principal causa que permite realizar el arbitraje y paso por el camino de datos en el mismo ciclo, es que este último es extremadamente sencillo (esta constituido por tan solo un multiplexor).

4.2.2 Router Basado en la Estructura *SIC.*

A diferencia de la estructura anterior, el planteamiento de esta propuesta es radicalmente distinto al que sería oportuno para una arquitectura de tipo determinista. En este caso, cada uno de los canales de entrada solicitan todos los posibles canales de salida para el paquete almacenado en la cola asociada. Para evitar los posibles conflictos en los canales de salida se recurre a un arbitrio global que atiende de forma secuencial a los canales de entrada. En la Figura 4-3 se muestra la estructura parcial de un router con este tipo de arbitrio. Como se puede apreciar, este sistema de arbitrio es el empleado en el encaminador *wormhole* adaptativo del capítulo anterior.

En primer lugar, el *AD* (*Address Decoder*) es el encargado de procesar la cabecera del paquete que se encuentra en la salida de la *FIFO*, comparando a cero, seleccionando los puertos de salida adecuados en función del signo de los *offsets* del *Routing Record* y generando cada una de las cabezas correspondientes a cada opción de salida. Los puertos de salida que acercan el paquete a destino son enviados al árbitro global. Éste, empleando una política *Round-Robin*, se encargará de seleccionar un canal de entrada con peticiones activas (Ver Figura 4-4). El canal seleccionado es procesado por la *RDU* y en función del estado de los puertos de salida y la función de selección, se asigna uno de los canales de salida solicitados por el *AD* (Notar que el árbitro es también el encargado de aplicar la función de selección). Una vez hecho esto, se actualiza el estado del crossbar indicando al multiplexor correspondiente cual es el canal de entrada que

debe ser propagado hacia la salida. Simultáneamente se permite el avance hacia el *AD* del resto del paquete en ciclos siguientes. Notar que el *AD* envía las dos cabezas alternativas a la entrada del crossbar de forma simultánea al envio de la petición al árbitro. De no hacerlo así, sería necesario un ciclo de reloj adicional para actuar después de la llegada de la señal de reconocimiento desde el árbitro.

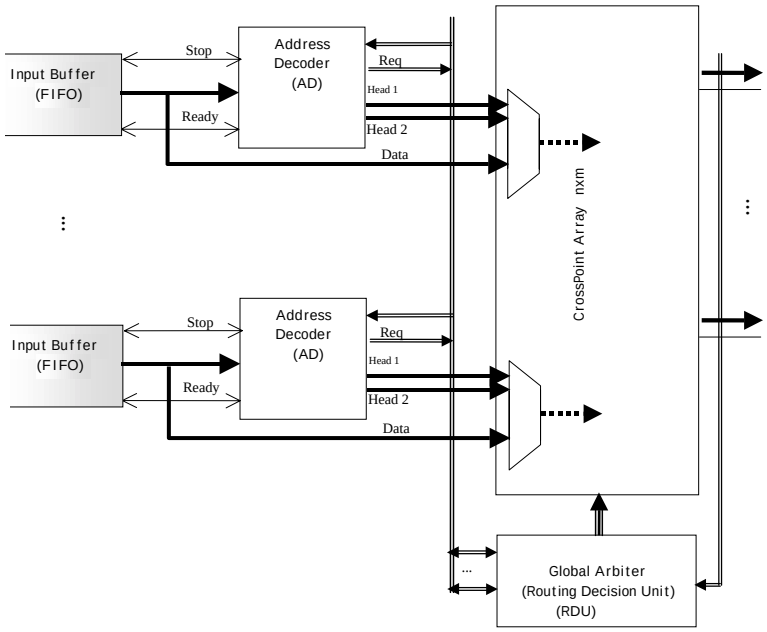


Figura 4-3. Estructura parcial del router para la propuesta SIC.

En cuanto al rendimiento de esta estructura, conviene señalar que es necesario un ciclo de reloj más que en la propuesta OAC. En primer lugar, el crossbar ha de manejar más señales de entrada y es bastante más costoso que el camino de datos del caso OAC. Requerir, aproximadamente, el doble de entradas, a causa de la transmisión de las dos cabezas, implica un multiplexor más grande, lo que hace que no sea posible hacer que el arbitraje y el camino de datos operen en el mismo ciclo. Además, el arbitraje también tiene que manejar más líneas que en el caso anterior, lo que implica que su coste temporal sea superior, siendo además el encargado de aplicar la función de selección. En definitiva, para mantener el tiempo de ciclo del encaminador acotado, es preciso segmentar los dos procesos en módulos. Esto implica que, en general, serán necesarios tres ciclos de reloj para procesar la cabeza del paquete y asignar la salida. Sin embargo, en la propuesta OAC, el proceso de arbitraje y paso por el crossbar se realiza en el mismo ciclo de reloj. En este último caso, dado que la RDU requiere un solo ciclo en ausencia de contención, el tiempo empleado en todo el proceso puede llegar a ser de tan solo 2 ciclos.

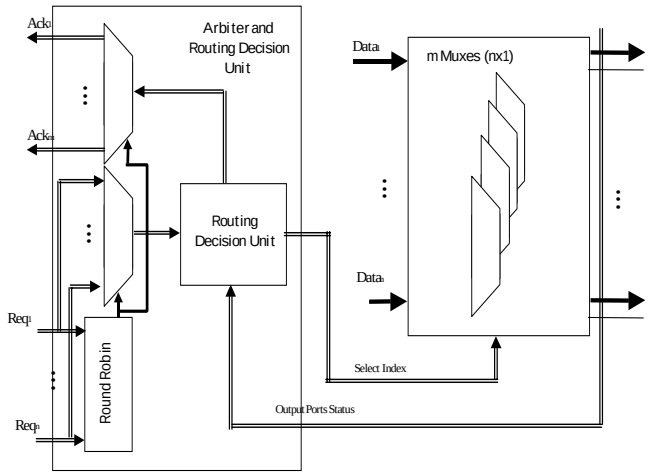


Figura 4-4. Estructura interna del arbitrio en SIC.

4.2.3 Router Basado en la Estructura *SAC.*

Por último, vamos a describir la implementación para un árbitro basado en la propuesta SAC (*Symmetric Arbiter Crossbar*). Con este tipo de arbitraje la configuración del router es idéntica a la que se muestra en la Figura 4-3. Sin embargo, ahora todos los canales de entrada pueden solicitar simultáneamente múltiples canales de salida y son atendidos a la vez por el árbitro. El problema fundamental que implica este planteamiento es la posible asignación múltiple de puertos de salida. Al no estar secuencializado ni el proceso de atención de entradas (SIC) ni el proceso de petición de puertos de salida (OAC), es necesario redefinir completamente el esquema de arbitraje con respecto a las dos implementaciones anteriores y arbitrar conjuntamente los canales de entrada con los de salida.

El problema de realizar un doble arbitrio entre puertos de entrada y salida (arbitraje simétrico) ha sido previamente estudiado en [125]. En dicho trabajo se propone un árbitro próximo al ideal denominado *Wave Front Arbiter (WFA)*. Se proponía esta estructura como mecanismo de arbitraje para encaminadores deterministas con colas DAMQ [126]. Aunque en este trabajo el contexto es diferente, la problemática asociada a ese tipo de colas es muy similar a la del encaminamiento adaptativo en lo que respecta al arbitraje interno del encaminador. Cabe indicar que este sistema ha sido empleado en el SGI Spider [55].

Este árbitro se basa en una matriz de celdas similar a la mostrada en la Figura 4-5. Esta figura corresponde al árbitro de un crossbar de 4 entradas y 4 salidas. Las filas y columnas de la matriz

corresponden respectivamente a las peticiones y estado de los canales de entrada y salida del crossbar.

Cada fila *i* de la matriz indica cuales son los puertos de salida solicitados por el paquete almacenado en la entrada *i*, apareciendo doblemente marcadas las celdas correspondientes a los canales de salida solicitados. En el caso de la figura, los puertos solicitados por la entrada 1 son los puertos de salida 2 y 3. Cada columna de la matriz corresponde a un puerto de salida. En la figura, el recurso se concede cuando la celda correspondiente se encuentra sombreada. Por ejemplo, el puerto de salida concedido a la entrada 1 es el número 2. El criterio a la hora de asignar cada uno de los puertos aparece en la figura: el proceso consiste en recorrer las celdas de la matriz siguiendo una *forma de onda* que se desplaza paralela a la diagonal, como se indica con las líneas (1),(2)...(7) de la figura. En el caso de que la celda *(i,j)* presente una petición, se tiene en cuenta si anteriormente el puerto de salida *j* ha sido asignado a otra entrada *k<i*, evaluando el valor de la señal que procede del *Norte*. Además, se tiene en cuenta si una de las salidas solicitadas por *i* ya ha sido concedida previamente. Esto se hace evaluando la señal entrante en la celda que proviene del *Oeste*. En caso de que el puerto de salida no este ocupado y no se haya asignado previamente otro canal de salida *k<j*, se asigna el puerto (actualizando la señal al *Sur* de la celda a 1 y anulando el resto de peticiones de la fila, colocando a 1 la señal que sale al *Este* de la celda).

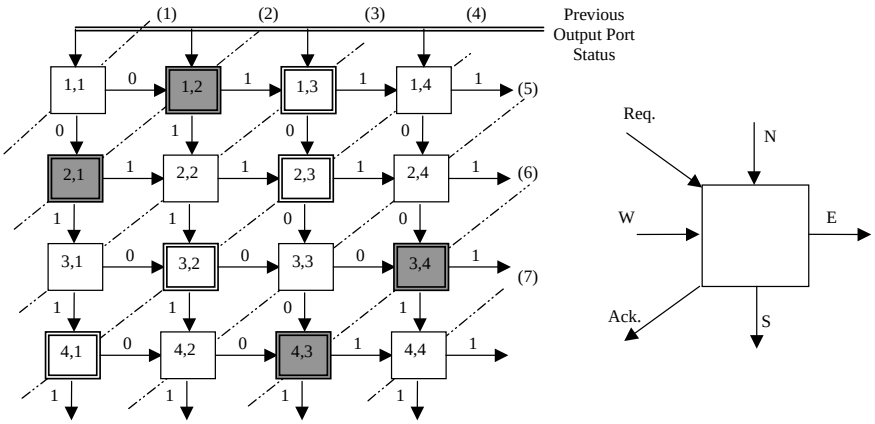


Figura 4-5. Wave Front Arbiter 4x4 y celda de arbitrio.[1]

Este tipo de asignación de recursos proporciona un *throughput*, en teoría, muy próximo al máximo número de puertos del encaminador. Sin embargo, la estructura, tal y como se ha descrito, no está exenta de problemas. Uno de ellos es la inanición. Al concederse los canales con criterios estáticos (la entrada asociada a la primera fila de la matriz de celdas tiene más priori-

1. En la asignación final que aparece en la Figura se supone que ningún puerto de salida ha sido asignado previamente.

dad), es posible alcanzar una situación en la que una cola de entrada de menos prioridad quede bloqueada indefinidamente. Con el fin de evitar este problema se puede recurrir a la utilización de un *token* que permita variar el orden de prioridad en los canales de entrada dinámicamente. Sin embargo la utilización de criterios semejantes a los que se pueden usar en los árbitros *Round-Robin* no es clara, ya que los canales a conceder son múltiples. Para solventar esta situación, en este trabajo se ha optado por utilizar un *token* a modo de contador (varía independientemente de las condiciones del tráfico). Otro tipo de mecanismo igualmente válido para resolver este problema es la utilización de un *timeOut,* aunque a priori parece más complicado que el mecanismo propuesto. La estructura del árbitro con prioridades no estáticas es la que aparece en la Figura 4-6.

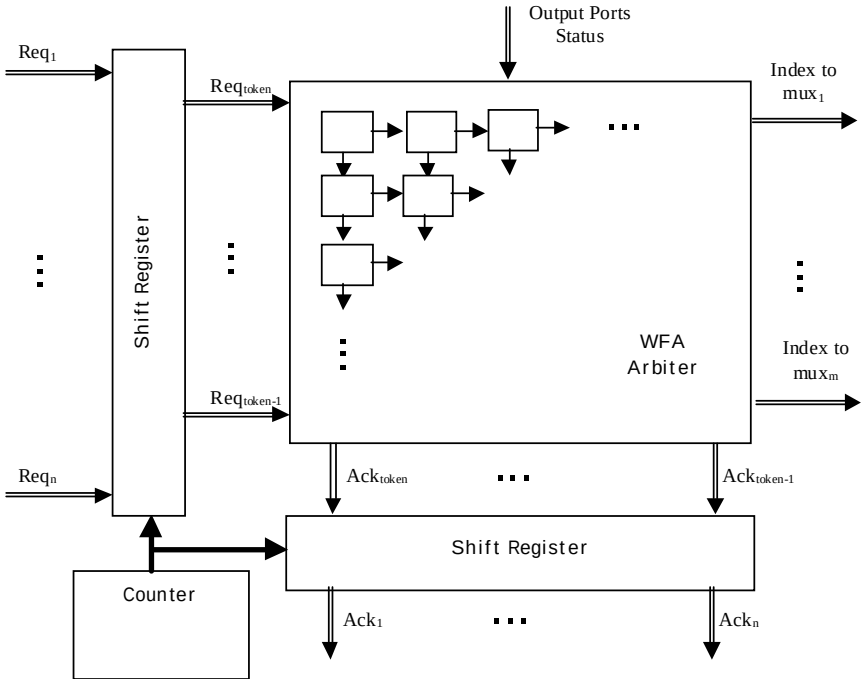


Figura 4-6. Mecanismo de evitación de un posible starvation en WFA.

Otro aspecto a tomar en cuenta es la función de selección de los canales. En el *WFA* descrito previamente se supone siempre la misma prioridad en los puertos de salida, independientemente de cual sea la entrada que los solicita. Con esta estructura, el canal de salida correspondiente a la primera columna de la matriz de celdas del arbitro, tendrá una prioridad más alta de ser asignado (en el caso de estar libre) independientemente del canal de entrada que lo solicite. Bajo este punto de vista, independientemente del criterio de selección y de los canales de entrada, el orden de prioridad en la asignación es fijo.

La solución adoptada en esta tesis para eliminar esta limitación es la mostrada en la Figura 4-7. La función de selección tiene sentido solo en el caso de que el canal de salida sea adaptativo, dado que, en el caso de solicitar también canales de escape, siempre serán la última opción en la selección del puerto de salida definitivo. En general, su orden de preferencia es siempre menor que el de los canales adaptativos. Bajo estas circunstancias, su prioridad de asignación siempre es mínima, lo que hace que estén asociados a las últimas columnas de la matriz de celdas del WFA. Para efectuar la selección definitiva de un puerto entre los canales de salida adaptativos, cada canal de entrada tiene asociado una serie de líneas adicionales mediante las cuales se indica el orden de prioridad de cada petición de canal de salida. Estas lineas son utilizadas para realizar un reordenamiento de las señales internas de la matriz (N y S). De esta manera, la primera celda de la fila corresponderá a la salida de máxima prioridad y así sucesivamente hasta la última celda de la fila. Una vez asignado el puerto de salida correspondiente, es necesario establecer los índices del camino de datos del crossbar, para lo que es necesario reordenar en su estado original los resultados (S) de la primera fila de celdas. Como se puede apreciar en la figura, esto se hace mediante un conjunto de multiplexores situados entre cada una de las filas de la matriz de celdas original. Se repite el proceso para el segundo canal de entrada, reordenando de nuevo las celdas de la fila correspondiente en función del bus de prioridades. De manera consecutiva se repite el proceso hasta recorrer todas las celdas de la matriz.

Cabe indicar que, tanto la generación de la lista de prioridades como la reordenación de la señales de petición y reconocimiento, son llevadas a cabo por el *Address-Decoder* de cada canal de entrada con el fin de equilibrar las dos etapas.

Como se muestra en [125], los resultados en *throughput*, en términos estructurales, que alcanza este tipo de arbitrio pueden llegar a ser muy superiores a los de los otros tipos de estructuras. Sin embargo, las restricciones que impone el algoritmo de encaminamiento en sí mismo y los patrones de tráfico, afectan notablemente al rendimiento máximo del sistema. Por ello, se puede anticipar que el coste añadido de la implementación hardware de este sistema van a mermar su rendimiento final.

.

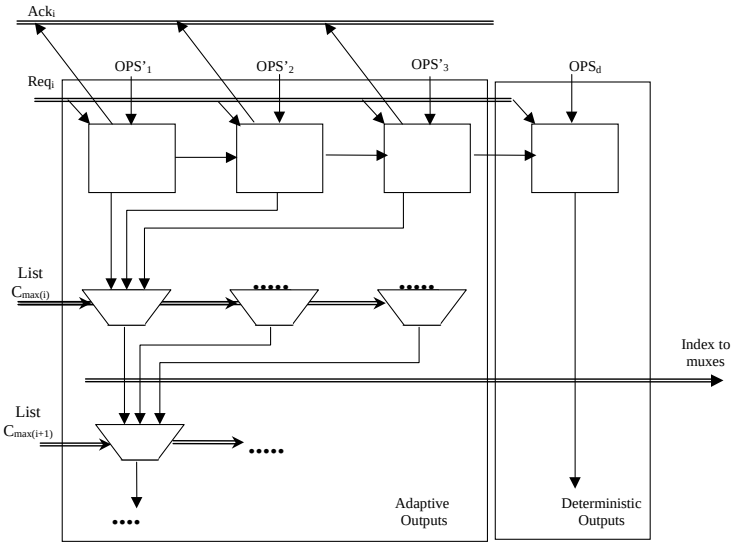


Figura 4-7. Incorporación de la función de selección en la matriz del árbitro WFA.[1]

4.3 Diseños Hardware: Aplicación a Encaminadores de Redes *k-ary n-cubes*.

Con el fin de probar la eficacia de las tres propuestas de arbitraje en una situación concreta, se ha optado por utilizar encaminamiento mínimo y completamente adaptativo sobre topologías *k-ary 2-cube* y *k-ary 3-cube* (Toros 2-D y 3-D). El mecanismo de evitación de bloqueos está basado en la *Burbuja* (Ver Capítulo 3). Con este tipo de algoritmo se necesitan únicamente dos canales virtuales por línea física para evitar interbloqueos. Por otro lado, la función control de flujo empleada en todos los encaminadores es *Virtual Cut Through* (VCT). Por lo tanto, la multiplexación de canales virtuales se puede realizar a nivel de paquete, luego esta tarea puede ser fácilmente realizada por el crossbar sin incrementar excesivamente su coste, evitando así la utilización un controlador de canales virtuales de salida que incrementaría en un ciclo el pipeline del encaminador. Bajo estas condiciones, el crossbar tiene un total de 9 canales de entrada y 5 canales de salida para el caso 2-D y para el caso 3-D, 13 canales de entrada y 7 de salida.

En el caso del árbitro *WFA* de la implementación SAC, la multiplexación basada en paquetes afecta sensiblemente a su estructura. La solución adoptada para evitar que en un mismo ciclo de petición, tanto el canal virtual adaptativo como el determinista asociados a una misma línea física, sean asignados simultáneamente, se incorpora una serie de puertas NAND tanto en el

1. Las entradas OSP' corresponden al estado de los puertos de salida reordenados de acuerdo con la lista de prioridades $C_{max(i)}$

envio del índice a la matriz de conexionado como los reconocimientos enviados a los *Address Decoders* de tal forma que siempre la asignación que prevalece es la adaptativa sobre la determinista (Ver Figura 4-8). Pese a ser conservadora, la principal ventaja de esta solución es que el incremento de complejidad que introduce es prácticamente despreciable. Podríamos suponer que esta solución introduce una bajada en el rendimiento ya que con este sistema estamos cancelando la concesión de todos los puertos de salida antes de que todos los puertos de salida solicitados sean evaluados para su concesión. Sin embargo, sabemos que solamente se puede solicitar un canal determinista[1], luego en realidad esta solución no afecta al rendimiento en términos generales dado que nunca se puede solicitar un canal determinista aparte del cancelado.

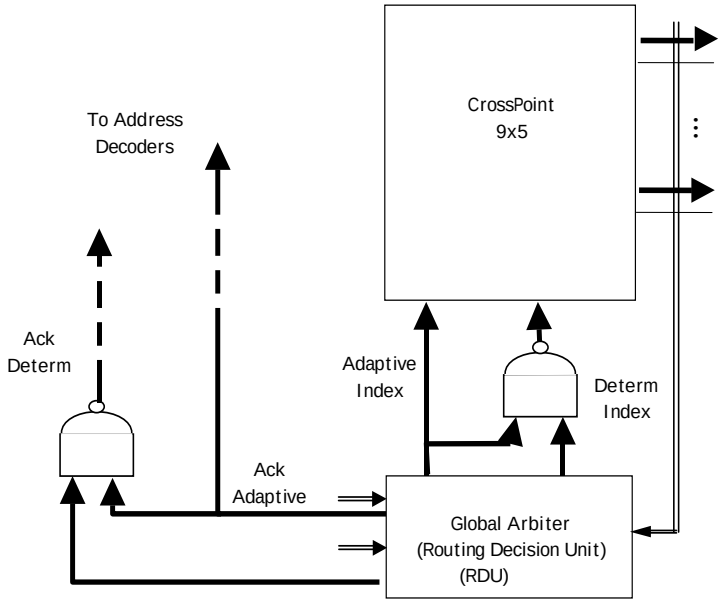


Figura 4-8. Multiplexación a nivel de paquete con el árbitro SAC.

En cuanto a la función de selección elegida a la hora de indicar la prioridad de cada canal de salida adaptativo, se ha utilizado una basada en X/Y dinámica. La elección de esta función de selección se basa en que ofrece mejores resultados que otras basadas en criterios de tipo estático como zig-zag, etc., sin implicar ningún incremento de complejidad en el encaminador. Se podrían plantear funciones de selección más evolucionadas como las basadas en históricos (MAX-CREDIT [55], LRU o LFU [131], etc.,) aunque este tipo de análisis se puede considerar ortogonal al ámbito del estudio que se presenta.

1. El acceso a los canales de salida deterministas ha de respetar las restricciones de la subfunción de encaminamiento determinista para asegurar que la red es libre de interbloqueo. De esta forma, cada paquete únicamente podrá avanzar por uno solo de los canales deterministas.

Para llevar a cabo la implementación lógica de las tres propuestas de arbitraje, descritas en el apartado anterior se ha procedido siguiendo la metodología mostrada en la Sección 2.2.4. El proceso de diseño *hardware* parte de descripciones VHDL de todos los componentes del router y su posterior verificación. Para la implementación hardware se han empleado celdas estándar de canal 0.7 μm y 2 niveles de metal (librería tecnológica ECPD07 de ES2/ATMEL).

En las Figuras 4-9 y 4-10 se muestra la segmentación de cada uno de los routers, con el tiempo empleado en ciclos para cada etapa. La estructura del pipeline es idéntica en el caso 2-D y 3-D. De la misma forma que en el capítulo precedente, cada uno de los encaminadores de la red funciona de forma síncrona, siendo el protocolo de comunicación entre encaminadores asíncrono, lo que permite evitar problemas relacionados con el *skew* del reloj. Esto obliga a la incorporación de un módulo de sincronización en la entrada de cada canal físico.

De igual forma que en el capítulo precedente, la anchura de los enlaces entre encaminadores vecinos es de 17 bits en total, estando uno de ellos dedicado a indicar la cola del paquete. Por otro lado, cabe reseñar que se ha mantenido constante la anchura de los enlaces al pasar de redes 2-D a 3-D. Dado que el interés de nuestro análisis no se centra en aspectos topológicos hemos considerado oportuno no alterar estos parámetros. Con respecto a la capacidad de almacenamiento, seguiremos empleando colas de 80 *phits* de profundidad por canal virtual y la longitud de los paquetes quedará establecida en 20 *phits*.

.

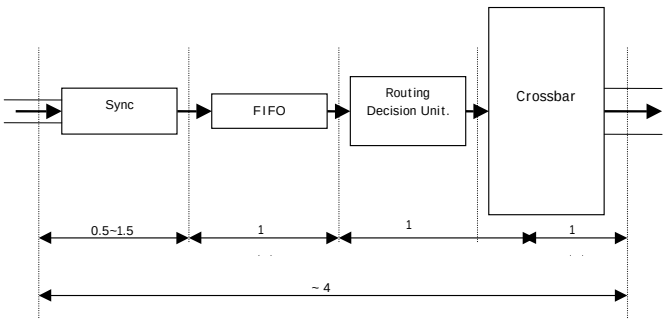


Figura 4-9. Segmentación del router con arbitrio OAC.

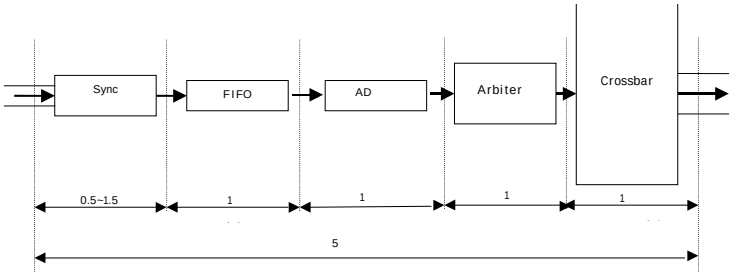


Figura 4-10. Segmentación del router con arbitrio SIC y SAC.

La herramienta directamente estima, de forma aproximada, los valores de tiempo y área de cada uno de los componentes del encaminador, comprobándose que, en todos los casos, es el árbitro el que marca el tiempo de ciclo del router. Dichos valores conducen a los resultados mostrados en la Tabla 4-1, para el caso de encaminadores 2-D y en la Tabla 4-2, para el caso de encaminadores 3-D.

Router	**Critical Path(ns.)**	**Tiempo de Paso (ns)**	**Área del Router (mm^2)**	**Area del Árbitro & Crossbar (mm^2)**
Implementación OAC	5.65	22.60	25.95	8.98
Implementación SIC	6.19	30.90	23.34	4.57
Implementación SAC	8.45	42.25	28.82	10.05

Tabla 4-1. Resultados finales de tiempo y área para cada una de las implementaciones en 2-D.

Router	**Critical Path(ns.)**	**Tiempo de Paso (ns)**	**Área del Router (mm^2)**	**Area del Árbitro & Crossbar (mm^2)**
Implementación OAC	6.28	25.12	45.57	16.24
Implementación SIC	7.5	37.50	34.68	7.70
Implementación SAC	11.69	58.45	59.04	32.06

Tabla 4-2. Resultados finales de tiempo y área para cada una de las implementaciones en 3-D.

Estos datos muestran claramente la penalización que introduce SAC tanto en términos de área como de tiempo de paso. Sin duda, la complejidad estructural de esta propuesta afecta, de modo determinante, a los resultados alcanzados por la herramienta de síntesis. Como se puede apre-

ciar el tiempo de ciclo del encaminador resultante es extremadamente alto comparado con el resto de alternativas, especialmente en el caso de los encaminadores 3-D.

Los resultados de la síntesis indican que los requerimientos de área de la estrategia SIC son menores que OAC, fundamentalmente porque en SIC se unifican todas las unidades de decisión de encaminamiento en el arbitro global, dejando que solo el módulo encargado de la decodificación de la dirección sea distribuido. Además, el mayor número de etapas del encaminador en el caso SIC permite optimizar mejor el área que ocupan los modulos no críticos. En este caso es posible reducir considerablemente el área ocupado por la matriz de conexionado del caso SIC, ya que su coste temporal es bastante menor que el del árbitro. Por ello, la herramienta de síntesis puede alcanzar una implementación lógica de área reducida. Otro factor que es crítico en este aspecto es que OAC incorpora un árbitro por puerto de salida. Todo ello, en conjunto, hace que el área del árbitro y crossbar de OAC sea más elevada que en caso de SIC. Sin embargo, por la misma razón se produce una degradación en la velocidad del reloj de la propuesta SIC con respecto a la OAC: el árbitro global debe gestionar más señales que los árbitros independientes de OAC. El incremento en el periodo de reloj es de un 10% para el caso 2-D y de un 15% para el caso 3-D, lo que repercute sensiblemente en la latencia promedio a cargas bajas y disminuye la productividad real de la red.

4.4 Análisis Comparativo Bajo Cargas de Trabajo Sintéticas.

En este apartado se presenta un análisis de rendimiento de cada uno de los encaminadores en dos configuraciones diferentes: en la primera, los encaminadores se encuentran aislados operando bajo condiciones de tráfico uniforme en todos sus canales de entrada, y en la segunda se analiza el rendimiento bajo patrones de tráfico sintético en una red concreta. En esta evaluación, de la misma forma que en el capítulo anterior, se ha empleado el simulador a nivel de transferencia de registros SICOSYS (Ver Apéndice A).

4.4.1 Encaminador Aislado

En esta configuración será posible evaluar el rendimiento en términos de *throughput* y latencia para cada una de las propuestas, sin tener en cuenta la descompensación en el uso de los recursos que introduce el tipo de tráfico empleado, ni la congestión que provoca la propia red. En este caso, únicamente hay contención a nivel del árbitro. Por lo tanto, esta configuración nos va a dar una idea del rendimiento máximo que puede llegar a proporcionar cada uno de los esquemas de arbitraje.

Se han utilizado dos tipos encaminadores distintos para cada una de las propuestas de arbitrio:

- Encaminador Bidimensional.
  Corresponde al empleado en una red toroidal bidimensional (*k-ary 2-cube*). Se ha configurado con 9 inyectores independientes, que pueden estar generando nuevos paquetes de forma simultánea. Cuatro de estos inyectores actúan sobre los canales virtuales adaptativos y los otros cuatro inyectan sobre los canales deterministas. El otro inyector corresponde al canal de inyección del encaminador. (Ver Figura 4-11).
- Encaminador Tridimensional.
  Es el empleado en una red de tipo toro 3-D (*k-ary 3-cube*). De forma similar al caso anterior, la evaluación de rendimiento de cada uno de los encaminadores se ha realizado incorporando 13 inyectores independientes en cada canal virtual y modelando adecuadamente el tráfico.

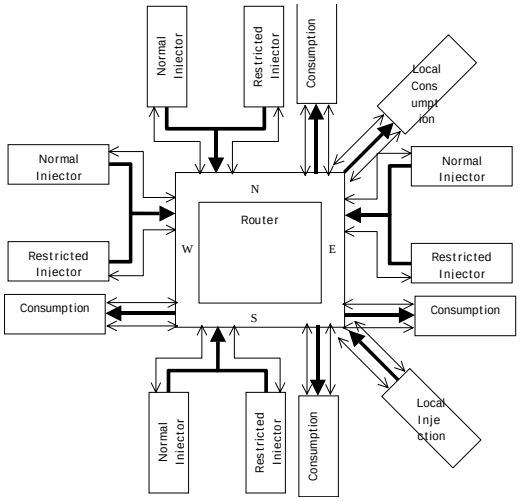


Figura 4-11. Configuración del encaminador aislado para una red 2D

Cada uno de los tráficos generados en los distintos tipos de canales modelan el tráfico de una red real, en el sentido de que respetan las condiciones del algoritmo de encaminamiento. Con este fin se genera un destino de forma aleatoria emulando las características de un algoritmo de encaminamiento mínimo. Consecuentemente, el paquete inyectado no puede salir por la dirección contraria en la que ha sido inyectado. Por otro lado, los inyectores adaptativos no presentan ningún tipo de restricción en el patrón de destinos (siempre tasa de generación aleatoria con distribución uniforme en los destinos) y los paquetes generados pueden salir por cualquier salida del router (de acuerdo con el algoritmo de encaminamiento). En cambio, los inyectores correspondientes a los canales deterministas solo pueden generar tráfico que sea coherente con los canales de escape. Bajo estas condiciones de tráfico (modelando los destinos de los paquetes como los de una red real bajo tráfico uniforme) y para una red con un número de nodos medio

o alto, podemos decir que la tasa de ocupación del canal de consumo será prácticamente despreciable frente al resto de los canales de salida del encaminador, suponiendo una red no excesivamente pequeña.

Otro aspecto a reseñar es la multiplexación de los canales de salida. Dado que los diferentes encaminadores utilizan un control de flujo *VCT*, la multiplexación se realiza a nivel de paquete. Por consiguiente, siempre y cuando un canal de salida esté transmitiendo un paquete por un canal virtual determinado, el otro canal virtual quedará inutilizado hasta que finalice la transmisión. En principio, esto no representa una gran diferencia con la multiplexación a nivel de *flit*, dado que, aunque el crossbar presenta un número de salidas superior, el índice de ocupación de las salidas ha de ser similar al que se puede alcanzar con la multiplexación a nivel de *flit*, puesto que la línea física de datos es la que va a limitar el flujo de información. Por otro lado, una consecuencia de realizar la multiplexación a nivel de *flit*, es que para las alternativas en que cada canal de entrada solicita simultáneamente todos los posibles canales de salida, al imponer la función de selección una prioridad más alta a los canales de salida adaptativos, la asignación final, en el caso de que sea satisfactoria, siempre será a un canal de salida adaptativo.

Por lo tanto, el throughput máximo que se puede llegar a alcanzar (en condiciones ideales) será aproximadamente igual a la suma de cada uno de los canales físicos de salida, sin incluir el canal de consumo. Este valor será del orden de $2 \cdot d \cdot b$ *(bytes/ciclo)*, donde d representa el número de dimensiones de la red y b la anchura de los enlaces en *bytes*.

Throughput Máximo en función de la longitud del paquete.

Las pruebas realizadas para este tipo de configuración, han consistido en evaluar el throughput máximo de cada alternativa para distintas longitudes de paquete. En las Figuras 4-12 y 4-13 se muestran los resultados para el caso de los encaminadores 2-D y 3-D respectivamente. Para la obtención de estos resultados, se ha utilizado siempre una profundidad de buffer por canal virtual, de 4 veces la longitud total del paquete considerado. Por encima de este valor no se observa prácticamente ninguna mejora de rendimiento.

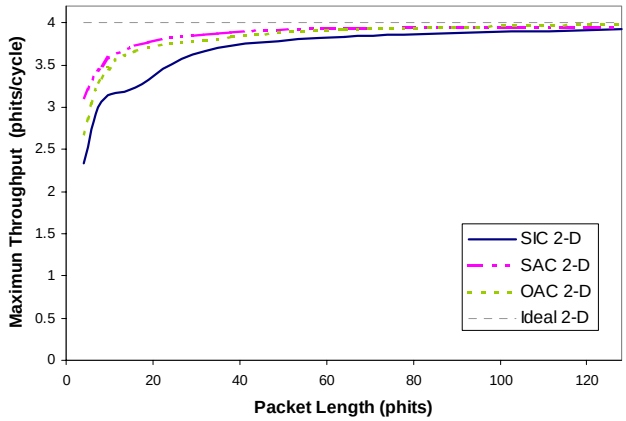


Figura 4-12. Throughput máximo con la longitud del paquete en 2-D.

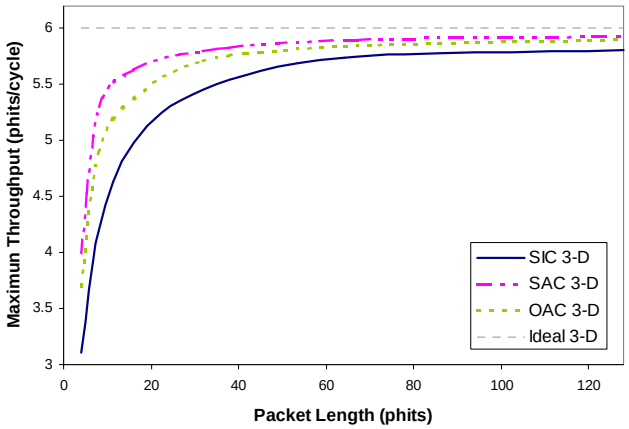


Figura 4-13. Throughput máximo con la longitud del paquete en 3-D.

Como era de esperar, para longitudes pequeñas, la capacidad del árbitro para atender de forma simultánea a varios canales es determinante a la hora de obtener un *throughput* más alto. Sin embargo a medida que la longitud de paquete aumenta, las diferencias en *throughput* tienden a desaparecer. Únicamente, el encaminador basado en la propuesta SIC se aleja de las demás alternativas. Notar que los resultados mostrados en este punto únicamente contemplan aspectos estructurales al no incorporar el tiempo de ciclo de cada uno de los encaminadores (*Throughput* expresado en *phits* por ciclo). Como se puede observar SAC ofrece, en el mejor de los casos (para paquetes muy cortos), un 20% (12%) de tráfico para el caso 2-D (3-D) que OAC y hasta un 30%(25%) de tráfico más que SIC. Cuando la longitud de los paquetes aumenta SAC prácticamente alcanza el mismo rendimiento que OAC y un 2% más que SIC.

Con el fin de mantener la coherencia con las condiciones empleadas en el capítulo anterior, estableceremos una longitud de paquete fija de 20 *phits* en todas las simulaciones. De la misma forma que en aquel caso, la elección viene motivada por la anchura de los canales de comunicación empleados y la configuración empleada para evaluar la red de interconexión en condiciones de carga real. De cualquier modo, es habitual que este tipo de arquitecturas empleen paquetes de longitud media próxima a la utilizada en nuestro análisis. Algunos ejemplos que podemos encontrar varían entre los 8 *flits* del SGI Origin[55] o 12 *flits* en el Cray T3E[112]. Si bien estos valores pueden oscilar en función del tipo de tráfico (tráfico de peticiones y tráfico de datos) y el carácter de los mensajes (usuario, mantenimiento, control, etc.).

En la Tabla 4-3 se detallan los datos, en términos tecnológicos, de la latencia base y *throughput* máximo alcanzado para la longitud de paquete establecida.

		OAC		**SIC**		**SAC**	
		2-D	**3-D**	**2-D**	**3-D**	**2-D**	**3-D**
Base Latency (ns.)		135.6	150.7	154.7	187.5	211.8	292.2
Through-put Max	(phits/ciclo)	3.65	5.55	3.24	5.54	3.77	5.73
	(flits/ns)	0.56	0.88	0.52	0.74	0.45	0.49

Tabla 4-3. Rendimiento con patrón de destinos uniforme en configuración aislada.

Las propuesta SAC presenta una productividad, en términos estructurales, más alta que la propuesta OAC. A su vez, esta última mejora considerablemente el rendimiento de la propuesta SIC. Al incorporar la complejidad asociada a cada implementación (*phits* consumidos por nanosegundo), la ganancia estructural de la propuesta SAC desaparece completamente, pasando a exhibir el rendimiento más alto en *throughput* la propuesta OAC. Además, es importante señalar que la latencia media a cargas medias-bajas de la propuesta OAC es notablemente mejor que la de SAC y llega a superar la latencia media de SIC.

A la vista de estos resultados, se puede decir que el coste de las mejoras introducidas en SAC no compensan su rendimiento, en comparación con los resultados obtenidos en OAC. Por otro lado, se puede observar el pobre rendimiento de SIC en estas pruebas. Esta bajada de rendimiento está causada por la contención que sufren los paquetes en espera de ser atendidos cuando la carga es elevada. Esto hace, que en determinados casos, al aumentar la carga se produzca una sensible disminución del *throughput* máximo ofrecido por el router.

4.4.2 Red *8-ary 2-cube* y *4-ary 3-cube*.

Por último pasaremos a evaluar el rendimiento de cada uno de los encaminadores analizados en el contexto de una red determinada. Para ello emplearemos una red de tipo *8-ary 2-cube* (toro 8x8) y a una red *4-ary 3-cube* (toro 4x4x4). En estas configuraciones, analizaremos el rendimiento para cada encaminador con los siguientes patrones de tráfico sintético: tráfico uniforme modal, uniforme bimodal, uniforme bimodal corto (ver página 119), matriz transpuesta, *bit reversal* y *perfect-shuffle*. Con este tipo de experimentos será posible analizar el impacto real de los diferentes planteamientos, teniendo en cuenta la contención debida a la naturaleza de cada tráfico y la propia red. En las tablas siguientes se muestran los valores de la latencia base y *throughput* máximo alcanzado por cada configuración y para cada uno de los tráficos considerados.

	Random	Bimodal Short	Bimodal	Matrix Tran.	Perfect-Shu.	Bit-rever
OAC	229.5	208.3	350.6	238.3	230.9	239.0
SIC	282.7	259.0	418.9	295.3	284.6	296.2
SAC	395.0	362.3	590.1	411.2	397.7	413.2

Tabla 4-4. Latencia base para los encaminadores 2-D (nanosegundos).

	Random	Bimodal-Short	Bimodal	Matrix Tran.	Perfect-Shu.	Bit-rever
OAC	254.7	235.7	402.2	271.8	257.6	250.9
SIC	301.8	276.5	474.7	306.7	307.7	295.0
SAC	482.4	442.8	764.5	490.0	491.5	471.7

Tabla 4-5. Latencia base para los encaminadores 3-D (nanosegundos)

	Random		Bimodal-short		Bimodal		Matrix Tran.		Perfect-Shu.		Bit-rever	
	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.
OAC	43.6	7.71	41.8	7.40	36.8	6.51	30.6	5.40	34.1	6.03	28.7	5.08
SIC	41.2	6.66	39.5	6.38	33.0	5.34	27.7	4.47	26.5	4.28	33.3	5.38
SAC	42.8	5.06	41.4	4.89	34.0	4.02	30.8	3.57	22.1	2.62	33.4	3.95

Tabla 4-6. Throughput máximo para encaminadores 2-D

	Random		Bimodal-short		Bimodal		Matrix Tran.		Perfect-Shu.		Bit-rever	
	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.	phits/ ciclo.	phits/ ns.
OAC	51.3	8.17	50.3	8.00	42.4	6.75	40.9	7.15	46.5	7.40	42.2	7.10
SIC	49.6	6.62	47.8	6.37	40.8	5.44	45.0	6.00	43.9	5.85	44.9	5.99
SAC	51.6	4.41	50.4	4.31	43.7	3.74	49.5	4.23	49.1	4.20	45.8	3.92

Tabla 4-7. Throughput máximo para encaminadores 3-D.

Finalmente en las figuras siguientes se muestra la evolución de la latencia y el *throughput* para todo el rango de cargas de trabajo con los patrones de tráfico más significativos. Para apreciar cual es la influencia de los costes *hardware* de cada implementación, representaremos las dos figuras de mérito de la red en términos estructurales (representación CNF) y tecnológicos (representación T-CNF).

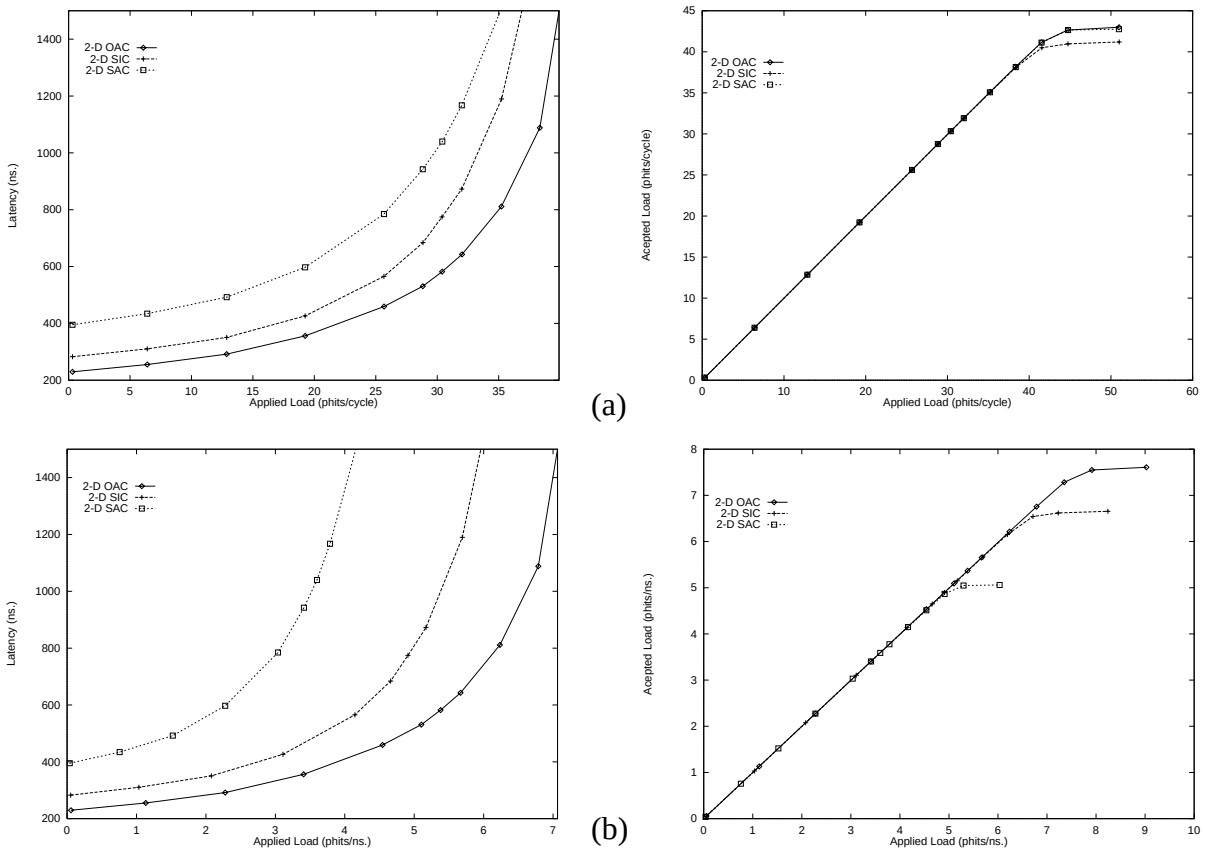


Figura 4-14. Latencia y Throughput para un toro 2-D de 64 nodos bajo tráfico uniforme (L=20 *phits)*. (a) Rendimiento estructural, (b) Rendimiento Tecnológico.

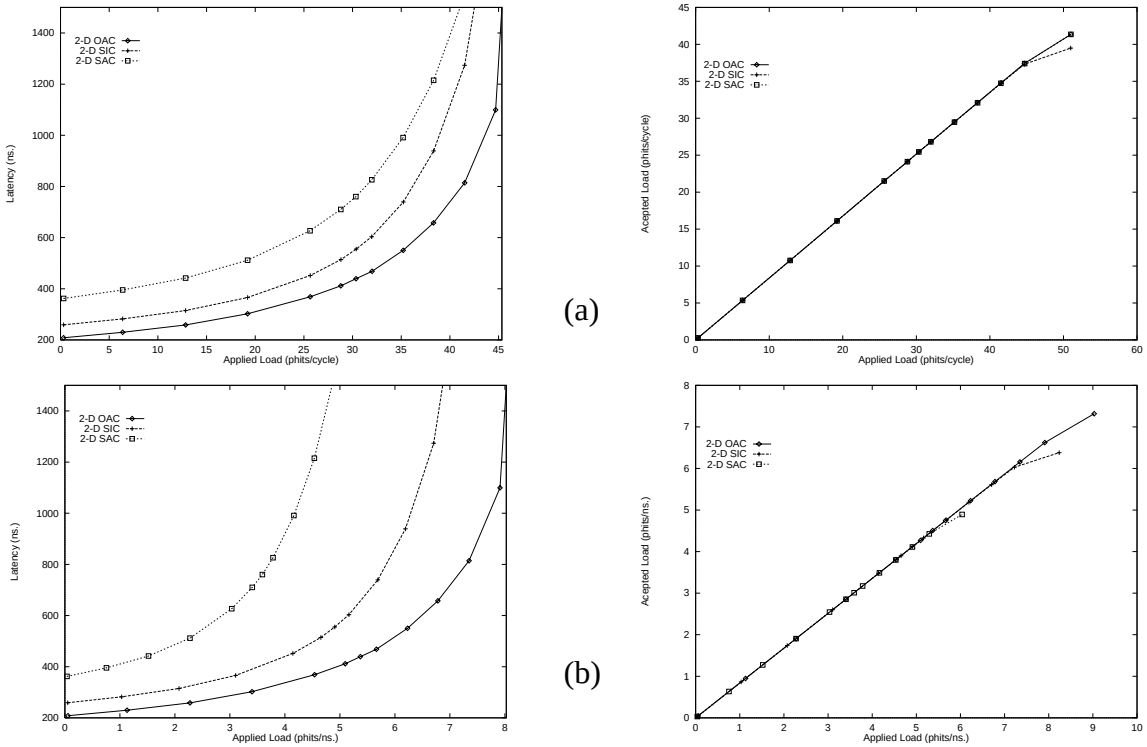


Figura 4-15. Latencia y Throughput para un toro 2-D de 64 nodos bajo tráfico bimodal corto (L_{largo}=20 *phits*, L_{corto}=4 *phits* con P_{largo}/P_{corto}=0.8) (a) Rendimiento estructural, (b) Rendimiento Tecnológico

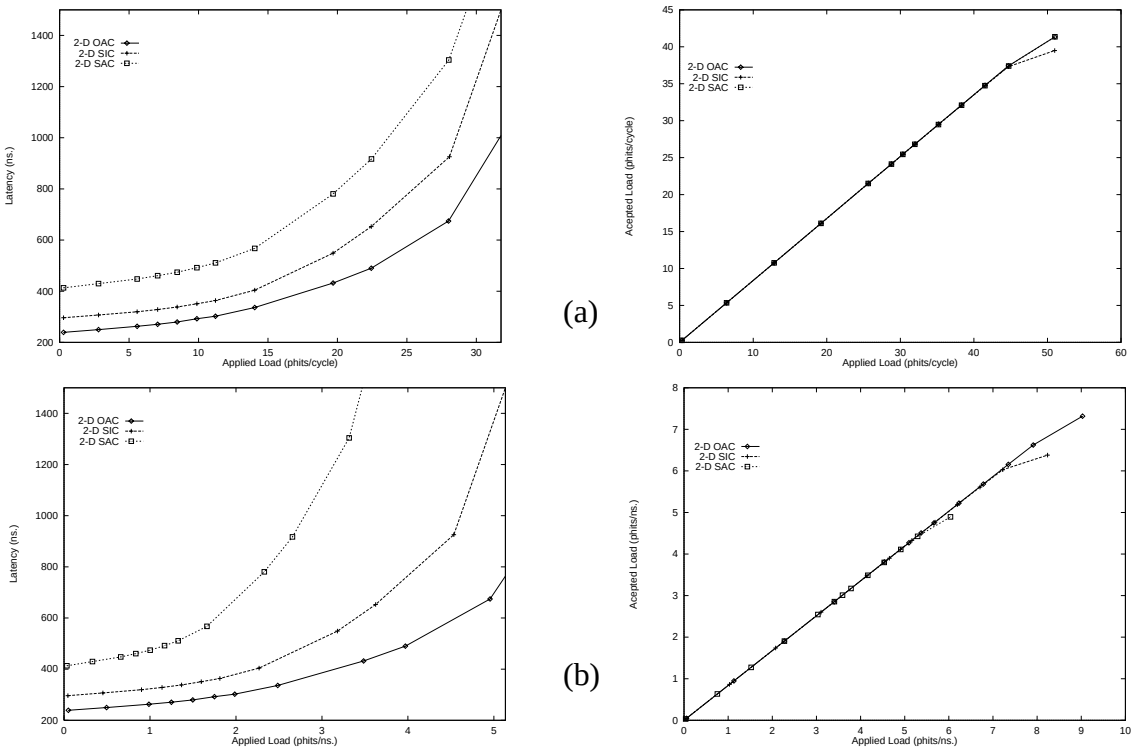


Figura 4-16. Latencia y Throughput para un toro 2-D de 64 nodos bajo tráfico Bit-Reversal (L=20 *phits)*.

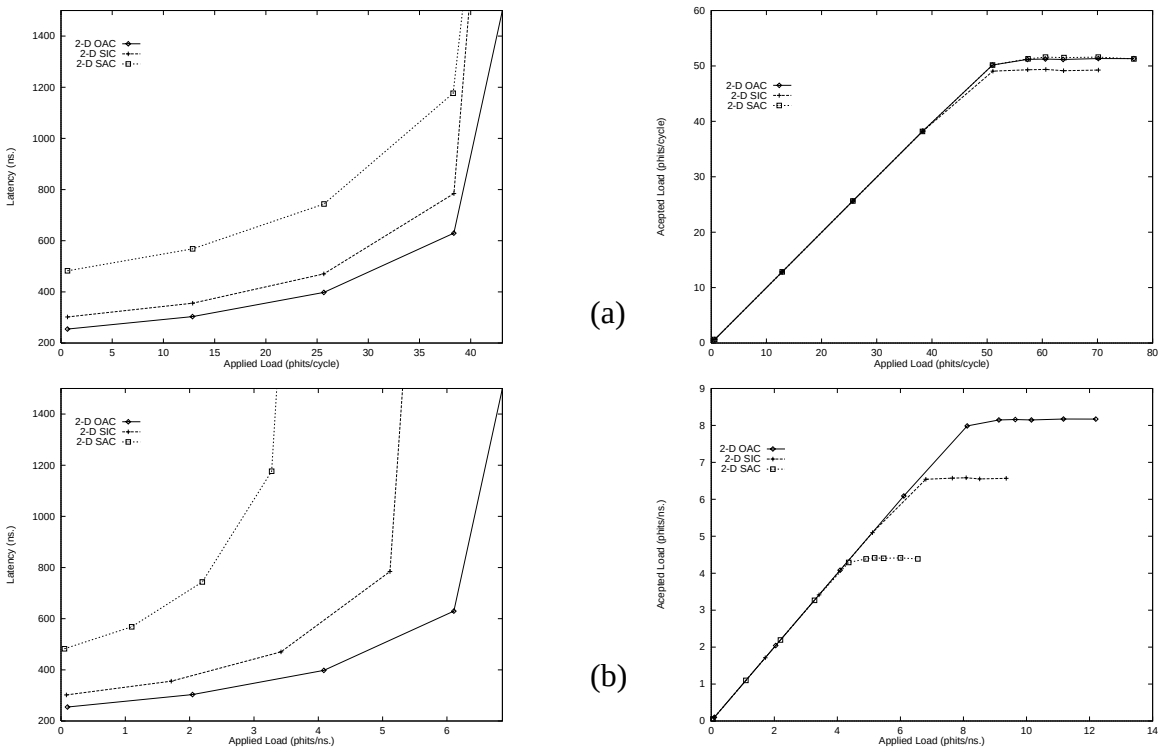


Figura 4-17. Latencia y Throughput para un toro 3-D de 64 nodos bajo tráfico uniforme (L=20 *phits*).

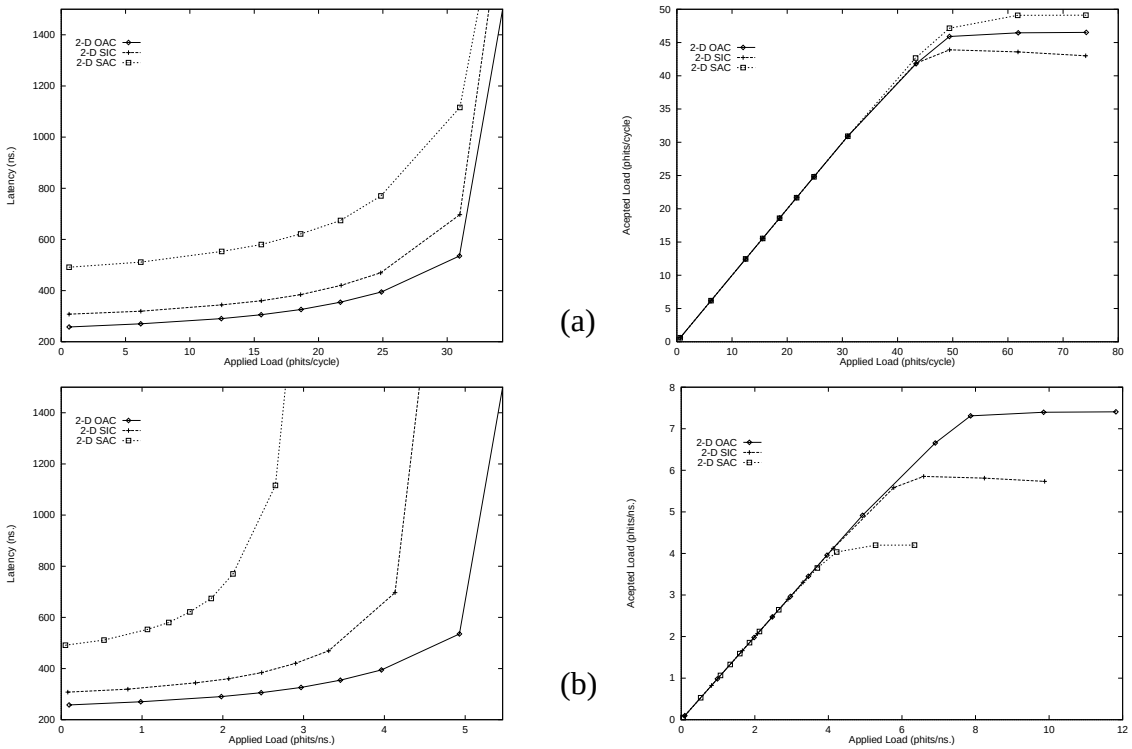


Figura 4-18. Latencia y Throughput para un toro 3-D de 64 nodos bajo tráfico Perfect-Shuffle (L=20 *phits*).

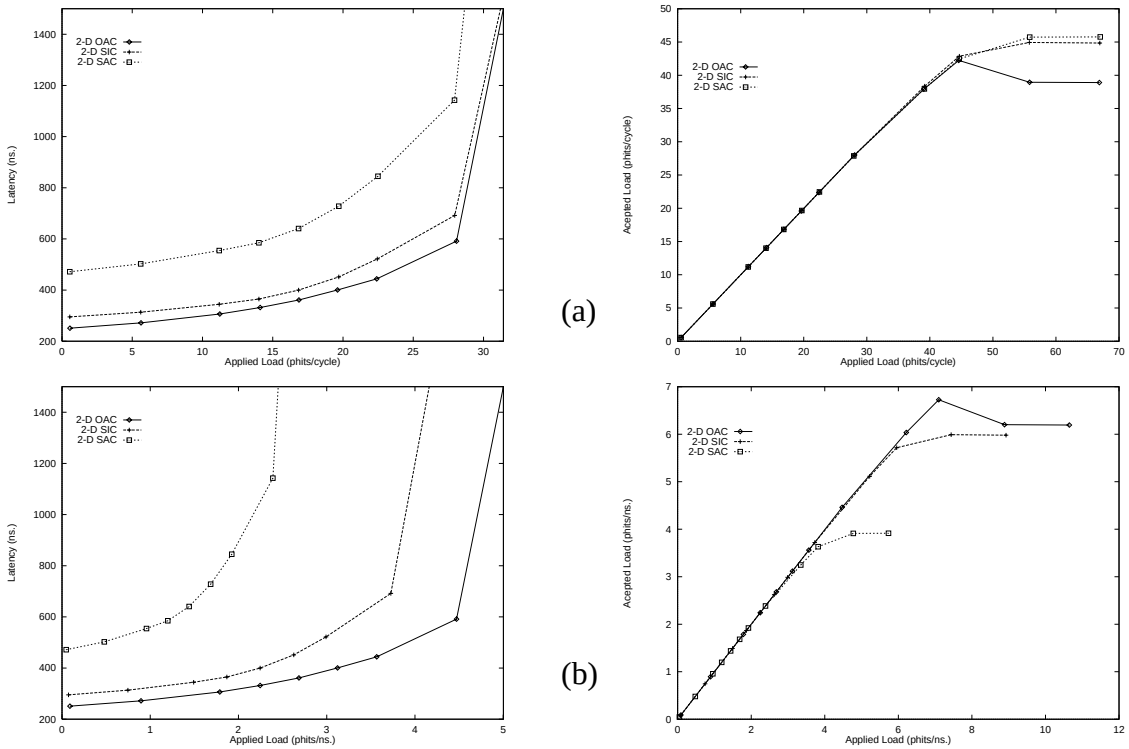


Figura 4-19. Latencia y Throughput para un toro 3-D de 64 nodos bajo tráfico Bit-Reversal (L=20 *phits*)

Estos resultados confirman los datos observados en la configuración aislada, mostrando de qué forma proponer una estructura conceptualmente complicada y optimizada como SAC, los condicionantes tecnológicos son cruciales en el rendimiento final de la red. La ganancia estructural que se observaba para este tipo de implementación, era importante al considerar una estructura aislada para el encaminador y tendía a perderse, con respecto a la propuesta OAC, a medida que se incrementaba la longitud de los paquetes. Al seleccionar una longitud de paquete similar a la empleada en los sistemas reales, las diferencias de productividad tendían a desaparecer. Sin embargo, lo que es determinante en el rendimiento final del encaminador son los diferentes costes tecnológicos. En este caso se ha observado como la propuesta SAC presenta un tiempo de ciclo superior en un 33% (2-D) y en un 46% (3-D) al del encaminador basado en la propuesta OAC, siendo ligeramente menores las diferencias con respecto al encaminador basado en la estrategia SIC. Estas diferencias influyen notablemente en el rendimiento final del sistema, dado que son mucho más importantes que las ganancias estructurales.

Por otro lado, es importante señalar que, entre todas las propuestas presentadas, la más sencilla a nivel conceptual (OAC) logra alcanzar los mejores resultados, tanto en términos de latencia como de *throughput*. Aparentemente, esta propuesta parte en desventaja con respecto al resto y de forma intuitiva se podría pensar que su rendimiento iba ser el más bajo. Sin embargo, de forma similar a la configuración aislada en términos estructurales, se observa una ligera ganan-

cia con respecto a SIC. Estas diferencias se ven incrementadas al incorporar las restricciones tecnológicas del diseño *hardware*. A nivel estructural la propuesta SIC proporciona los peores resultados debido a la contención que impone el arbitro: en una situación de carga alta los paquetes que esperan ser atendidos por el árbitro necesitan esperar mucho tiempo, que puede llegar a ser tantos ciclos como el número de canales virtuales de entrada. Esto hace que el tiempo promedio de paso por encaminador sea relativamente alto, y por tanto se produzca una caída en la productividad.

A la vista de los resultados de SIC vs. OAC, se puede decir que es mejor proponer estrategias de arbitrio en las que se favorezca la resolución concurrente de peticiones al nivel de la entrada que no secuencializar las peticiones favoreciendo la adaptación de los paquetes.

4.5 Análisis Comparativo Bajo Cargas de Trabajo Reales.

4.5.1 Características de la Arquitectura.

Con el objetivo de determinar si los importantes efectos del arbitraje del encaminador sobre el rendimiento de la red de interconexión, observados bajo cargas de trabajo sintéticas, se trasladan también a las aplicaciones reales, se ha empleado el entorno de simulación conducido por ejecución que se describe el Apéndice A. Las características más reseñables del procesador y jerarquía de memoria del sistema siguen siendo las que se emplearon en el capítulo anterior (Ver Tabla 3-9 en la página 126).

Las redes empleadas en este análisis se corresponden con las tres alternativas de arbitraje, variando el número de dimensiones. El tamaño de las redes analizadas es el mismo que el empleado en el análisis con cargas de trabajo sintéticas. En las dos configuraciones se han utilizado redes físicamente independientes para el tráfico de datos y el tráfico de peticiones con el fin de evitar un posible interbloqueo debido a la capacidad limitada de las colas de consumo. El tamaño de los comandos básicos del protocolo de coherencia es de 8 *bytes*. Por lo tanto, teniendo en cuenta que la línea de cache utilizada es de 32 *bytes*, tenemos que la longitud de los paquetes es de 20(16+4) *phits* para los mensajes de datos y de 4 *phits* para los mensajes de invalidación y requerimiento (Recordar que el ancho de los enlaces es de 2 bytes). Dado que el control de flujo empleado es VCT se ha utilizado una unidad mínima de almacenamiento en los bufferes de 20 *phits*. Aunque produce una pequeña degradación en el nivel de ocupación de los bufferes para situaciones de carga alta, permite que no sea necesaria la fragmentación y reensamblado de los paquetes de datos en la interface de red.

Por otro lado, es necesario tomar en cuenta la velocidad relativa de la red con respecto al procesador. Con este fin se ha mantenido la velocidad de reloj del procesador del capítulo anterior en todas las pruebas, luego su frecuencia de operación es de 650 MHz. Bajo estas condiciones y teniendo en cuenta los resultados de síntesis mostrados en las Tablas 4-1 y 4-2, la frecuencia de reloj de las redes correspondientes a cada una de las configuraciones de arbitrio son los que aparecen en la Tabla 4-8. Aunque estos valores son fuertemente dependientes del tipo de tecnología empleada (0.7μm en nuestro caso), la diferente escala en el rango de aplicación del procesador y del encaminador los hace bastante verosímiles. Por ejemplo, sistemas como el Cray T3E-1200/900 [114] tienen una relación en torno a 6 (75 MHz para los encaminadores y 450MHz para los procesadores).

	Frecuencia de reloj		**Velocidad relativa procesador/red**	
Tipo de Arbitrio	2-D	3-D	2-D	3-D
OAC	177 MHz	159 MHz	3.67	4.09
SIC	162 MHz	133 MHz	4.01	4.89
SAC	118 MHz	85 MHz	5.51	7.65

Tabla 4-8. Velocidades relativas entre el procesador y la red en cada caso.

En cuanto a las aplicaciones seleccionadas para llevar a cabo la comparativa, hemos seguido empleando las consideradas en el capítulo anterior, es decir, evaluaremos el rendimiento de cada propuesta con *FFT, Radix* y *LU*.

De forma similar a las evaluaciones de prestaciones con cargas reales realizadas en capítulos precedentes, restringiremos el análisis del rendimiento de cada aplicación, para cada tipo de arbitraje y configuración de red, a la fase central de cálculo de las aplicaciones. La fase de inicialización y finalización de las aplicaciones no son tenidas en cuenta ya que la red no juega un papel relevante en ellas.

4.5.2 Análisis Comparativo Para FFT.

En el caso *FFT*, el problema seleccionado es el establecido por defecto, es decir, la aplicación deberá calcular la transformada rápida de Fourier de una serie de 64K dobles complejos. En este caso, la aplicación posee muchas menos instrucciones que algunas de las ejecutadas posteriormente, lo que permite seleccionar este tamaño de problema sin requerir excesivo tiempo de simulación en el caso de las redes 3-D. En la Figura 4-20 aparece el tiempo de ejecución resultante, tanto para la configuración 2-D como la 3-D (ambas configuraciones están normalizadas al encaminador más lento de su categoría).

Las diferencias en tiempo de ejecución entre las tres alternativas, ya sea en 2-D o 3-D, son acusadas. En ambos casos, SAC exhibe un rendimiento muy pobre, llegando a resultar hasta un 35% más lenta la ejecución de la aplicación para estos encaminadores que para los basados en OAC. Por otro lado, los encaminadores OAC muestran una mejora notable de rendimiento frente a SIC, llegando a ser de casi un 15% en el caso 3-D. Otro efecto que se puede observar claramente, es que no todas las propuestas de arbitraje se comportan de la misma forma al pasar de 2-D a 3-D. De hecho, la penalización que introduce este cambio en el coste de cada encaminador tiende a contrarrestar la reducción en la distancia promedio topológica y la mayor productividad de la red. En esta línea, el tiempo de ejecución de la aplicación en SAC, al pasar a 3D, se reduce en un 15%, SIC en un 20% y OAC un 26%. De nuevo, la sencillez de OAC permite que el incremento en la complejidad del encaminador, al pasar de 2-D a 3-D, sea menor y por ello el tiempo de ejecución tienda a mejorar más que en el resto de alternativas. Por la misma razón, las diferencias entre las propuestas de arbitraje tienden a incrementarse al incrementar el número de dimensiones de la red.

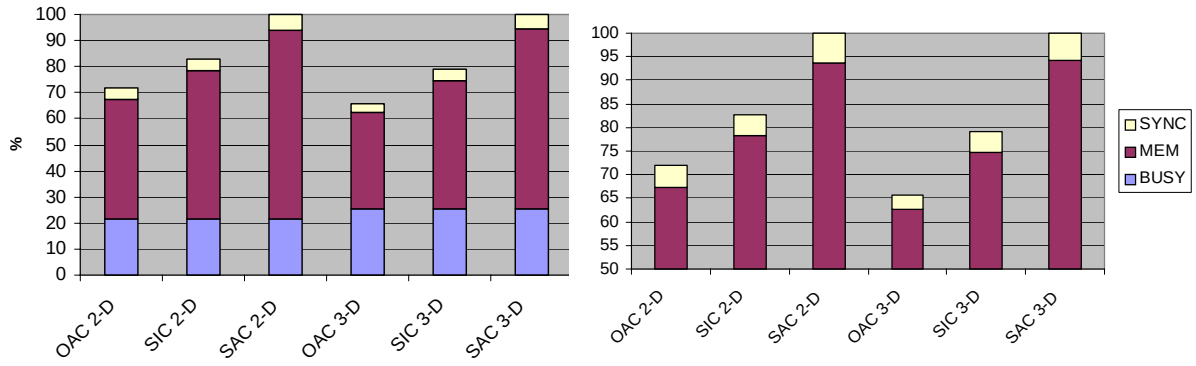


Figura 4-20. (a)Tiempos de ejecución de FFT para 64 procesadores en un Toro 8x8 y 4x4x4, (b) Detalle.

Los resultados de tiempo de ejecución observados para esta aplicación son acordes con los que se obtenían en el tráfico sintético. Sin embargo, es preciso profundizar más porqué puede influir tanto el mecanismo de arbitrio en el rendimiento del sistema completo. Para ello pasaremos a analizar en qué estado se encuentra la red a medida que se ejecuta la aplicación. En este sentido, centraremos nuestro foco de atención en la red de respuestas, ya que es la que más tráfico va a soportar y por ello la que más repercusión va a tener en la ejecución de la aplicación. En las Figuras 4-21, 4-22 y 4-23 se muestran estos datos. En primer lugar, en la Figura 4-21 se puede apreciar la demanda ejercida por la aplicación en términos de carga aplicada y la respuesta de la propia red de interconexión. La representación solo incluye efectos estructurales de los encaminadores, dado que las unidades empleadas en la escala del eje de ordenadas son *phits* inyectados por ciclo de red. De la misma forma que en el caso de las cargas sintéticas, el arbitraje SIC ofrece un rendimiento claramente inferior al resto de propuestas. Por otro lado, OAC

y SAC se encuentran bastante próximos, aunque SAC tiende a ofrecer un rendimiento ligeramente superior en las zonas de carga elevada.

Sin embargo la información que muestra esta figura es engañosa ya que no refleja de ningún modo el tiempo de ciclo de cada encaminador y por lo tanto en cada caso el eje de ordenadas esta distorsionado. Para aclarar las medidas, en la Figura 4-22, se muestra el rendimiento real de la red. Al representar la carga promedio que debe manejar la red en términos de *phits* por ciclo de procesador, estamos normalizando los resultados a la frecuencia de reloj del procesador, que obviamente es única. Aquí las diferencias se aprecian con mayor claridad. Las diferencias en los tiempos de ciclo de los encaminadores afectan considerablemente la productividad máxima que puede llegar a alcanzar la red. Es claro como en SAC este efecto hace que el rendimiento real de la red baje de forma notable. De forma similar, al incrementar el número de dimensiones de la red, la influencia es más acusada que en el resto de encaminadores, llegando a ser el nivel de *throughput* máximo inferior incluso al caso 2-D. Por otro lado, OAC mantiene un rendimiento mucho mas elevado que cualquiera de las demás propuestas. Frente a SIC, los efectos estructurales observados en la Figura 4-21 se incrementan al incorporar el efecto del tiempo de ciclo de la red. Como se puede apreciar en todos los casos, cuanto mayor es el máximo de productividad ofrecido por la red de interconexión, la anchura de dichas zonas es menor (Las unidades del eje de abcisas están normalizadas ya que son ciclos de procesador). Notar que el área encerrada por dichas zonas tiene que ser aproximadamente constante para todos los encaminadores ya que representa el numero de mensajes total generado por la aplicación. Este ensanchamiento de las zonas de carga acusada, obviamente repercute directamente en el tiempo de ejecución de la aplicación. De hecho, los tiempos de ejecución mostrados en la Figura 4-20 tienen una clara correlación con estas figuras.

Por último, la latencia, en cada caso, se comporta de forma muy similar. Como se puede apreciar en la Figura 4-23, la latencia promedio de los mensajes en SAC es considerablemente más elevada que en el resto de casos. Las diferencias con el resto de propuestas llegan hasta los 400 ciclos de procesador, lo que representa un cifra considerablemente elevada. De la misma forma que en el caso de la carga aceptada, el elevado tiempo de ciclo de los encaminadores basados en este sistema es la causa de estas diferencias. En los casos de OAC y SIC las diferencias son también claramente apreciables, si bien no tan acusadas como en SAC. Aquí las diferencias están motivadas por la ligera diferencia de tiempo de ciclo de cada encaminador y el mayor rendimiento estructural de OAC. Como se puede apreciar, la diferencia en el tiempo de ejecución de esta aplicación para ambas propuestas es próximo al 15% mientras que la diferencia en el tiempo de ciclo del encaminador es inferior al 10%. El rendimiento final, como no podría ser

de otra forma, viene influenciado tanto por las características tecnológicas del encaminador como por su rendimiento estructural.

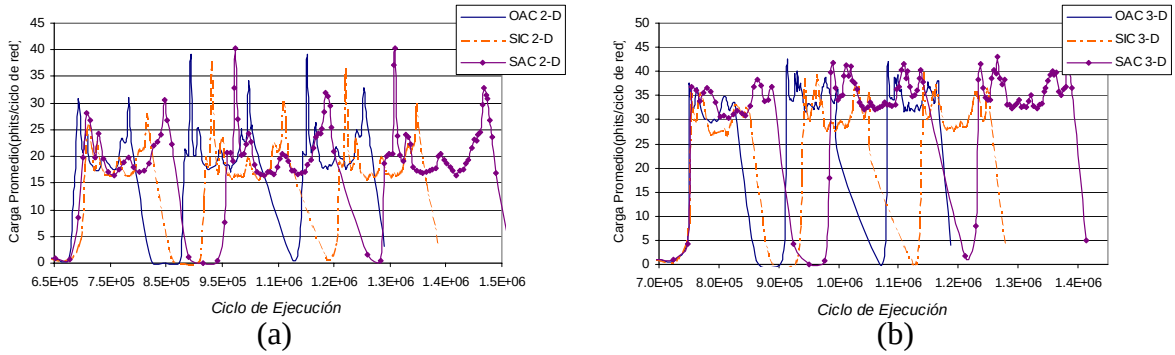


Figura 4-21. Evolución de la carga promedio (en *terminos estructurales*) sobre la red de peticiones en la ejecución de FFT: (a) redes 2-D, (b) redes 3-D.

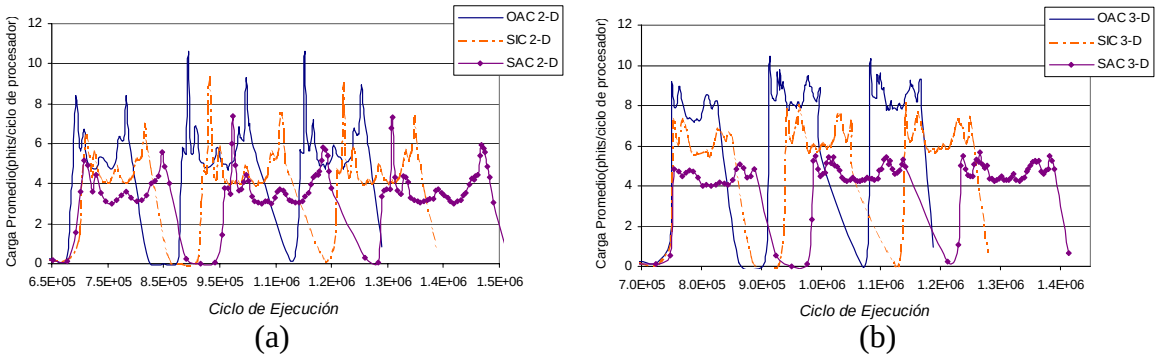


Figura 4-22. Evolución de la carga promedio (en *terminos reales*) sobre la red de respuestas en la ejecución de FFT: (a) redes 2-D, (b) redes 3-D.

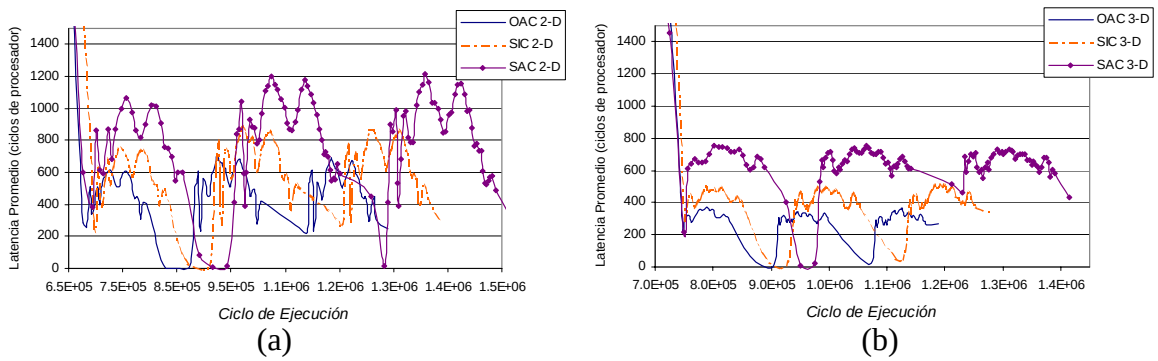


Figura 4-23. Evolución de la latencia promedio (en *terminos reales*) sobre la red de respuestas en la ejecución de FFT: (a) redes 2-D, (b) redes 3-D.

4.5.3 Análisis Comparativo para Radix.

El tamaño de problema seleccionado para Radix consiste en un conjunto de 512K claves enteras a ordenar y donde el valor del radio máximo de las claves también es 512K. La reducción en el tamaño del problema con respecto al establecido por defecto viene motivada por el excesivo requerimiento, en tiempo de simulación, en el caso de las redes 3-D. Pese a esta reducción, las características de escalabilidad de la aplicación no se ven afectadas. Como es conocido, la relación cálculo comunicación en esta aplicación únicamente depende del número de procesadores (Ver Capítulo 2), lo que refuerza la validez de los resultados que podamos obtener bajo estas circunstancias.

En la Figura 4-24 se muestran los resultados, en tiempo de ejecución, de la fase central de cálculo de Radix para cada propuesta de arbitraje, tanto para el caso 2-D como 3-D con 64 procesadores. Los datos están normalizados al tiempo de ejecución de la propuesta más lenta en cada número de dimensiones por separado.

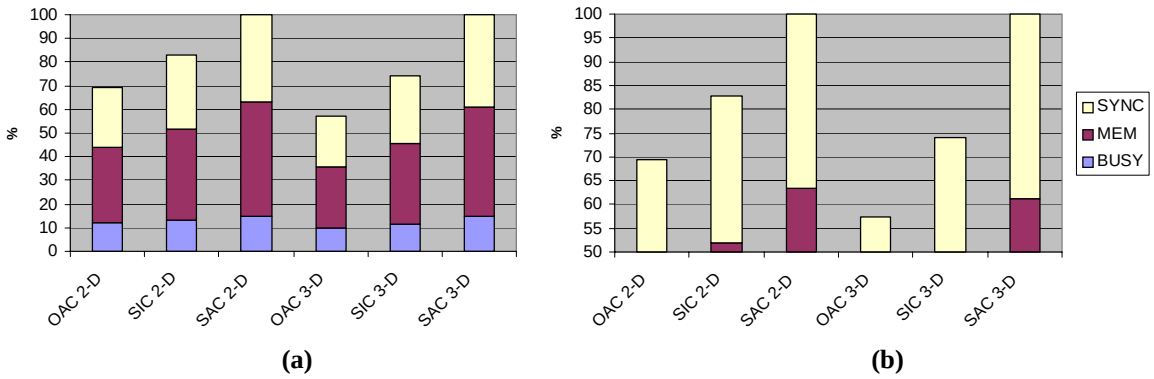


Figura 4-24. (a) Tiempos de ejecución de Radix para 64 procesadores en un Toro 8x8 y 4x4x4, (b) Detalle.

Los tiempos de ejecución que se muestran en la Figura 4-24 indican como, de nuevo, la complejidad de la implementación SAC, y el consiguiente incremento en su tiempo de ciclo, produce una perdida de rendimiento próxima al 30% con respecto a la implementación OAC en el caso 2-D, llegando a ser la diferencia en el caso 3-D de casi un 45%. El comportamiento del sistema para esta aplicación es bastante similar al de la aplicación anterior, llegando a incrementarse sensiblemente las diferencias. En este caso, al pasar de 2-D a 3-D, SAC y SIC incrementan el tiempo de ejecución en aproximadamente un 20% y un 10% respectivamente. Sin embargo, OAC con este cambio, permite mejorar el tiempo de ejecución en aproximadamente un 2%.

De igual forma que en la *FFT*, para clarificar estos datos pasaremos a estudiar cuál es el estado de la red de respuestas a medida que la aplicación *Radix* es ejecutada. En las Figuras 4-25, 4-26 y 4-27 se muestran las trazas de ejecución para cada caso, indicando respectivamente el nivel de carga de la red en términos estructurales y tecnológicos por una parte, y la latencia real de los mensajes por otra. De nuevo se puede constatar cómo la influencia estructural de cada encaminador es claramente apreciable. Como se puede ver en la Figura 4-25, SAC logra alcanzar mayor *throughput* que el resto de encaminadores en términos de *phits* consumidos por ciclo de red. De la misma forma que en el caso del tráfico sintético, las diferencias con OAC son bastante exiguas en las zonas de carga elevada y claramente apreciables con respecto a SIC. Se observa como en la zona de tráfico local SAC logra sacar una ventaja considerable a ambas propuestas (en torno a 5*phits* por ciclo de red). Sin duda la capacidad del arbitro para gestionar más peticiones facilita que las diferencias sean considerables. Notar que esta situación, prácticamente, se corresponde a la configuración en la que evaluábamos el rendimiento de cada encaminador de forma aislada.

Sin embargo, como ya sabemos, estos datos están distorsionados ya que el eje de ordenadas de esas figuras no está normalizado. Si normalizamos los resultados, representando la productividad en cada caso en *phits* consumidos por ciclo de procesador, las diferencias pasan a ser elevadas, como se puede ver en la Figura 4-26. De nuevo ocurre algo muy similar al caso de *FFT* y OAC logra superar, con un margen bastante elevado, tanto a SIC como a SAC tanto en 2-D como en 3-D. Aparentemente, las diferencias tienden a ser superiores que en el caso *FFT*, lo que parece justificar el incremento en las diferencias en el tiempo de ejecución en cada caso. De la misma manera se aprecia como la latencia de los mensajes es superior. Sin duda, al ser una aplicación más demandante que la anterior, la latencia promedio, en número de ciclos de red, es superior y por ello el impacto de las diferencias en el tiempo de ciclo de los encaminadores en el tiempo de ejecución de la aplicación, también es mayor. De la misma forma, al pasar de 2-D a 3-D, la presión ejercida por la aplicación se acerca más al límite de la red, lo que conlleva un mayor incremento en la latencia de los mensajes que en el caso anterior. Al tener en cuenta cada tiempo de ciclo, el resultado definitivo es que, en el caso de los dos encaminadores mas complejos, se produce una caída importante de rendimiento.

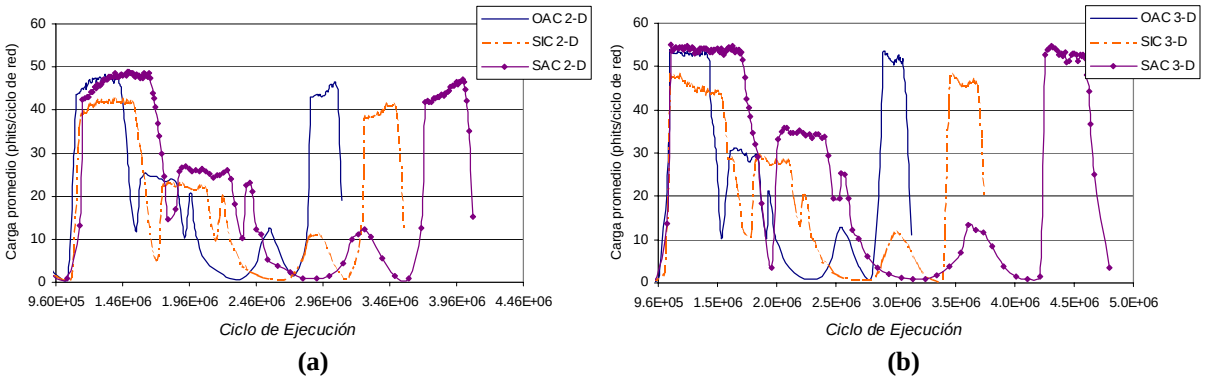


Figura 4-25. Evolución de la carga promedio (en *términos estructurales*) sobre la red de peticiones en la ejecución de Radix: (a) redes 2-D, (b) redes 3-D.

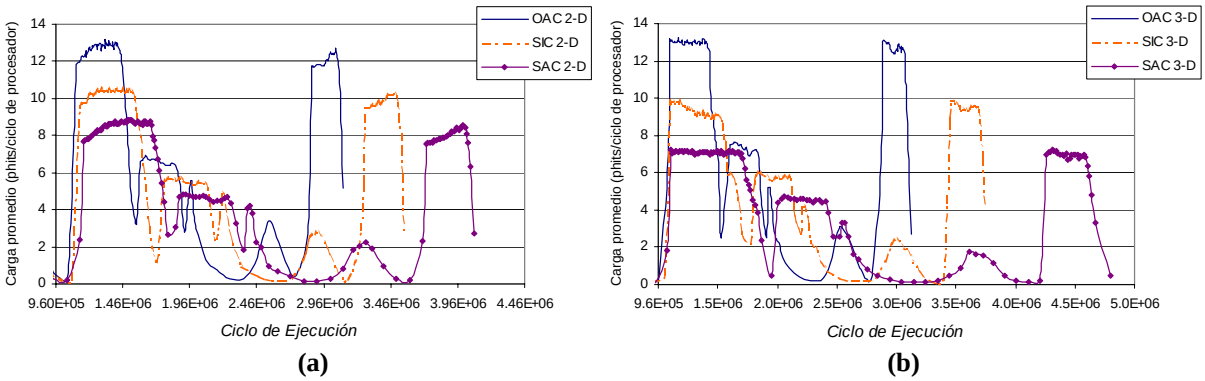


Figura 4-26. Evolución de la carga promedio (en *términos reales*) sobre la red de respuestas en la ejecución de Radix: (a) redes 2-D, (b) redes 3-D.

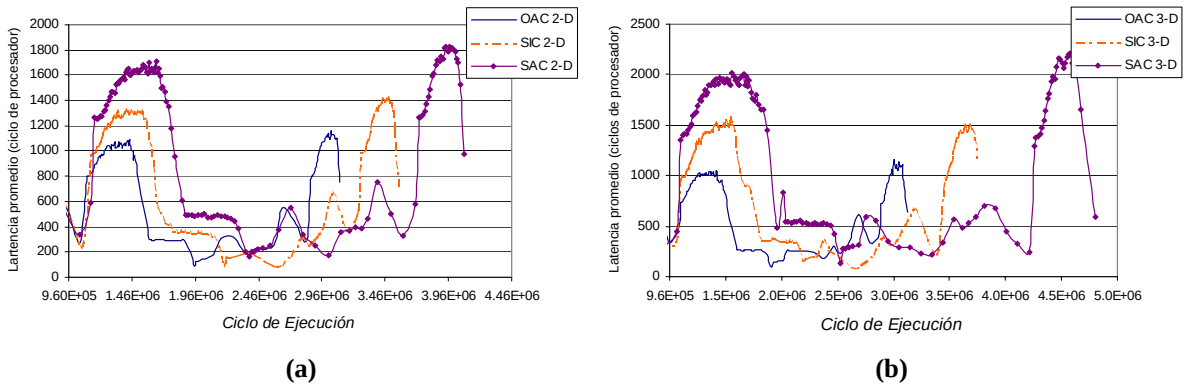


Figura 4-27. Evolución de la latencia promedio (en *términos reales*) sobre la red de respuestas en la ejecución de Radix: (a) redes 2-D, (b) redes 3-D.

4.5.4 Análisis Comparativo para LU.

Por último, analizaremos cual es el comportamiento de cada uno de los encaminadores en el caso de *LU*. El tamaño de problema seleccionado es el mismo que el empleado en el capítulo anterior, es decir, la matriz a factorizar tendrá un tamaño de 256x256. Como en aquel caso, la reducción en el tamaño de problema por defecto de la aplicación, viene motivado por lo demandante que es en tiempo de simulación. El tiempo de ejecución obtenido con cada encaminador alternativo y número de dimensiones se muestra de forma normalizada en la Figura 4-28. Como se puede apreciar, las diferencias entre SIC y OAC no son tan acusadas como en las aplicaciones previas. De cualquier manera, SAC sigue siendo la estructura que peor rendimiento muestra, llegando a hacer que la aplicación sea ejecutada hasta un 20% mas lenta que OAC en el caso 2-D y hasta un 35% en el caso 3-D.

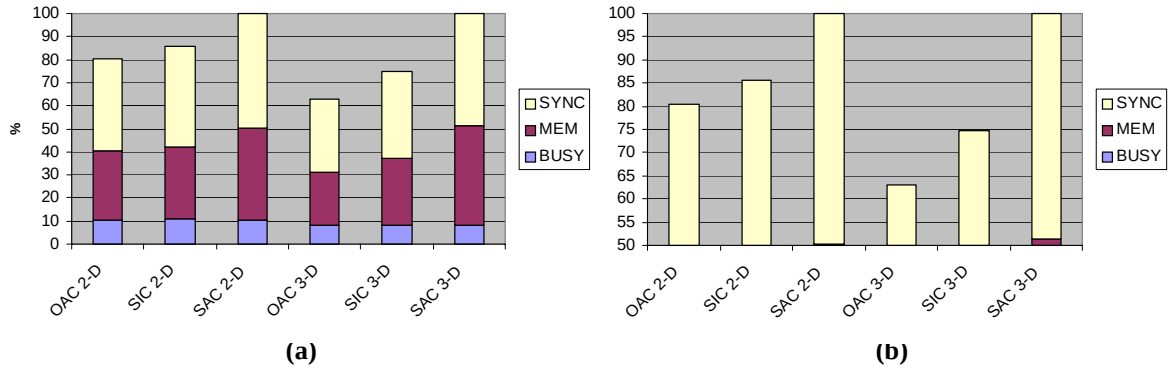


Figura 4-28. (a)Tiempos de ejecución de Radix para 64 procesadores en un Toro 8x8 y 4x4x4, (b)Detalle

Para comprender las diferencias mostradas en las Figuras 4-29, 4-30 y 4-31 se representa la evolución del estado de la red de respuestas para cada encaminador y número de dimensiones. Como podemos apreciar en la Figura 4-29 la presión ejercida sobre la red en términos generales es bastante inferior a las aplicaciones anteriores. Sabemos que el tipo de tráfico asociado a esta aplicación se caracteriza por generar puntos calientes de carga en los nodos encargados de procesar los bloques correspondientes a la diagonal de la matriz a factorizar. Como ya indicábamos en el capítulo anterior, la adaptatividad juega un papel muy importante en este sentido y el efecto inmediato de este hecho es que, en promedio, el número de adaptaciones que sufrirán los paquetes en su camino al destino será elevado. Por todo ello, es lógico que las diferencias entre SIC y OAC en términos estructurales sean menores que en las aplicaciones anteriores, ya que sabemos que el primero tiende a favorecer la adaptación de los paquetes a las condiciones del tráfico. Por ello, es completamente lógico que las diferencias en el tiempo de ejecución entre las dos propuestas sean menores que en *Radix* y *FFT*.

Por otro lado, SAC sigue ofreciendo, como era de esperar, mejor rendimiento en términos estructurales que cualquiera de la demás propuestas. Sin embargo, al tener en cuenta el rendimiento real de la red (Figura 4-30) vemos como el tiempo de ciclo vuelve a ser determinante. Dado que la carga que genera la aplicación es reducida, también las diferencias finales entre cada uno de los encaminadores son menores. Algo muy similar ocurre con la latencia promedio de los mensajes (Figura 4-31).

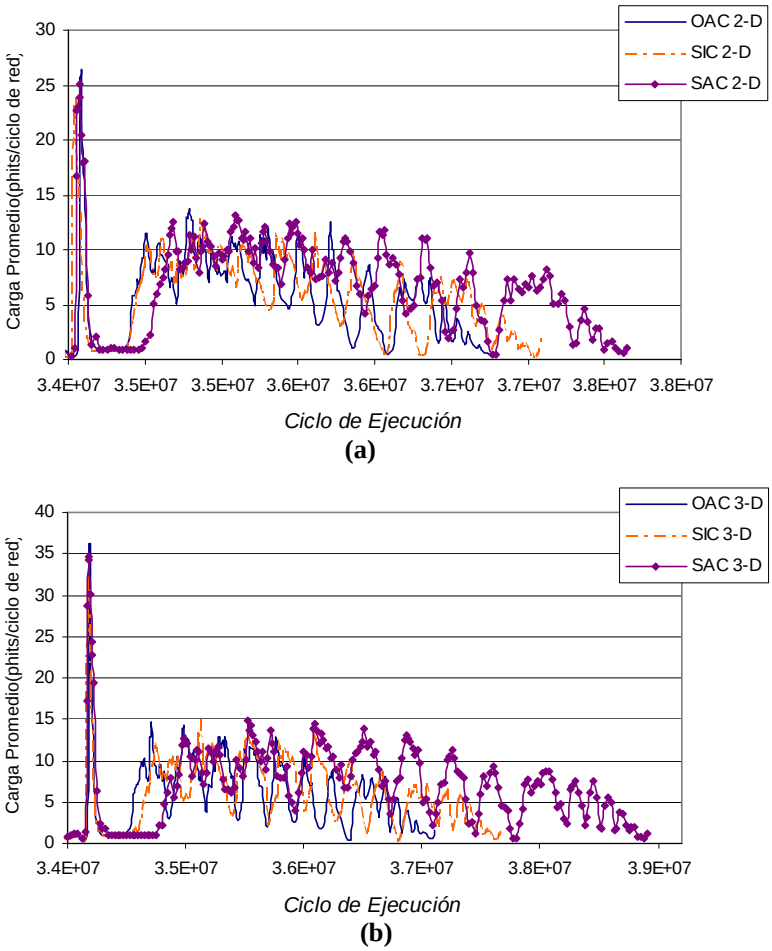
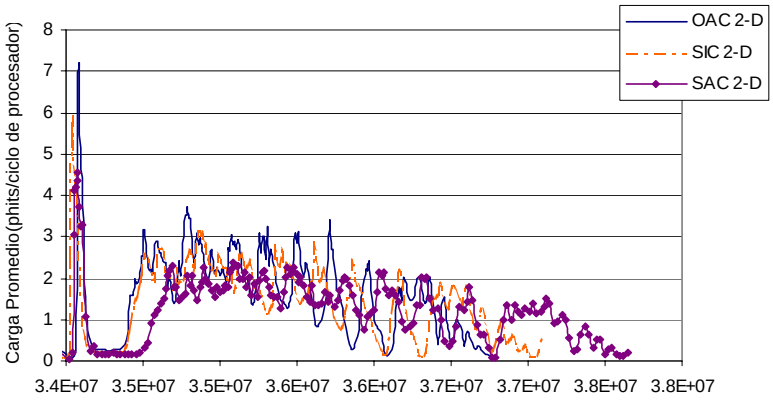


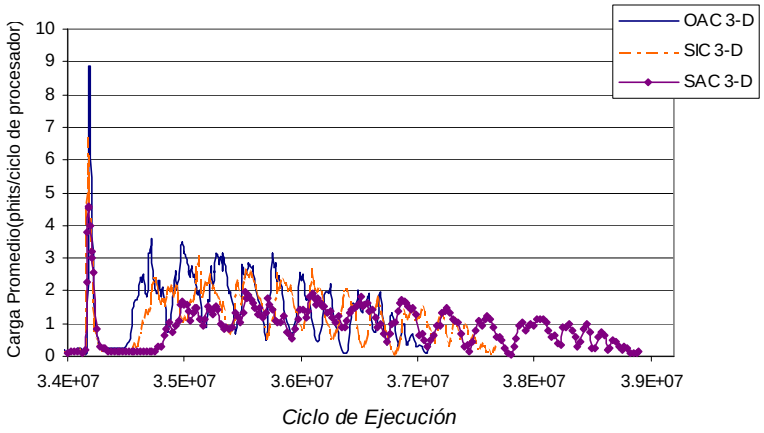
Figura 4-29. Evolución de la carga promedio (en *términos estructurales*) sobre la red de peticiones en la ejecución de LU:(a) redes 2-D, (b) redes 3-D.

Por otro lado, se observa que, en general, todos los encaminadores provocan que la aplicación muestre un tiempo de ejecución más lento para las redes 3-D que para las 2-D. Como se puede apreciar la latencia de los mensajes en el primer caso siempre es superior. De nuevo, la causa que provoca el incremento en la latencia no es la carga elevada que ha de soportar la red sino los puntos calientes. Pese a que, en el caso 3-D, las opciones de adaptación son mayores, esto no es suficiente para compensar el incremento en el tiempo de ciclo de los encaminadores. De cualquier modo, OAC solo empeora en un 5%, SIC en un 12% y SAC en un 30% el tiempo de ejecución con respecto al caso bidimensional. De esta forma, al influir más el tiempo de ciclo

de los encaminadores que en aplicaciones previas, es lógico que las diferencias entre las tres propuestas se acentúen.



(a)



(b)

Figura 4-30. Evolución de la carga promedio (en *términos reales*) sobre la red de respuestas en la ejecución de LU: (a) redes 2-D, (b) redes 3-D.

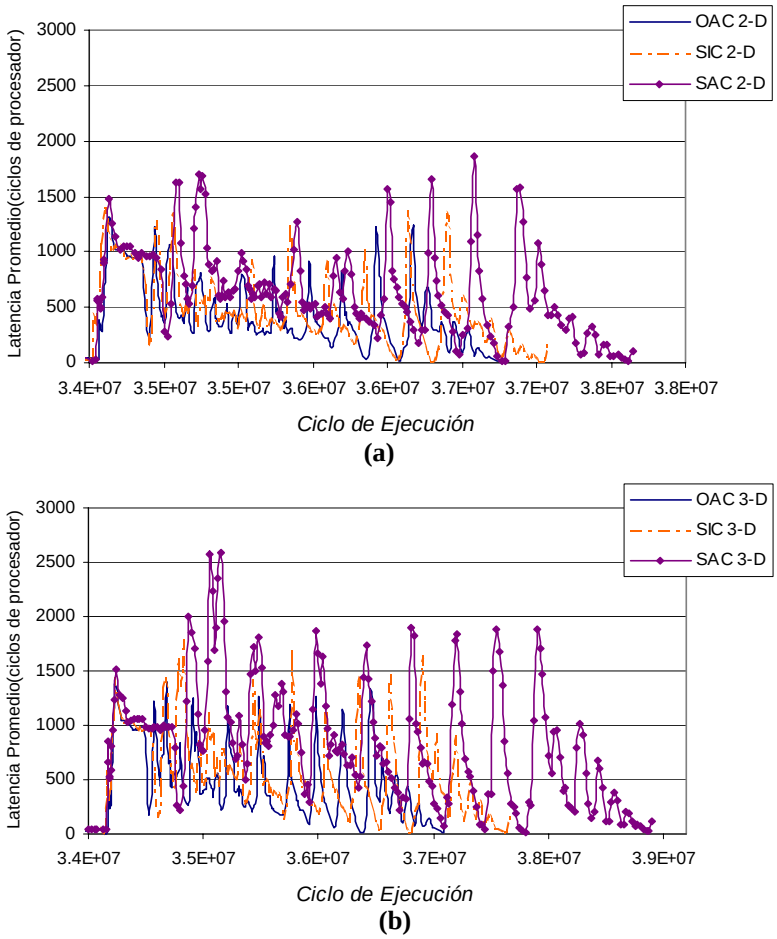


Figura 4-31. Evolución de la latencia promedio (en *términos reales*) sobre la red de respuestas en la ejecución de LU: (a) redes 2-D, (b) redes 3-D.

4.6 Conclusiones.

Las principal conclusión que muestra el estudio presentado en este capítulo es que los efectos dependientes de la implementación hardware de los encaminadores influyen de manera notable en el rendimiento de las redes de interconexión. Es por ello imprescindible ser conscientes de que al proponer arquitecturas evolucionadas desde el punto de vista funcional (como SAC) es necesario analizar cuales son los costes subyacentes. Los resultados finales son muy dependientes del *hardware* asociado. En el caso de no tener en cuenta este tipo de factores es muy probable que los resultados del análisis sean completamente erróneos.

Por otro lado, hemos propuesto un sistema de arbitrio que si bien desde el punto de vista funcional no parece muy adecuado, los condicionantes tecnológicos lo convierten en la arquitectura más oportuna. Básicamente, la idea perseguida con este diseño es hacer que el encaminador opere como lo haría uno determinista para rangos de carga baja y que los costes añadidos de por

la adaptatividad solo aparezcan a rangos medios-altos. Como en el capítulo precedente, la motivación es clara: buscar la máxima simplicidad en el encaminador sin renunciar a las ventajas de la adaptatividad. Al comparar esta nueva propuesta con esquemas tradicionales como SIC, los resultados muestran un rendimiento mejor, debido fundamentalmente a que la contención que se produce en la etapa de arbitraje es mayor que en nuestra propuesta. La razón de este comportamiento es clara e indica que es más importante favorecer que varios canales de entrada puedan solicitar de forma simultánea una salida al árbitro del encaminador que favorecer las adaptaciones de los paquetes. En promedio, el número de adaptaciones que sufren los paquetes en el camino a su destino será menos relevante que la posibilidad de que varios paquetes se encuentren esperando la atención del árbitro. El primer caso es el que favorece la propuesta SIC y el segundo caso es favorecido por OAC.

Independientemente de las estructuras planteadas, una conclusión clara que se extrae de este estudio es que el tiempo de ciclo del encaminador puede afectar de modo más que considerable al rendimiento del sistema global. Como ha quedado claro, el impacto puede llegar a ser muy elevado fundamentalmente debido al hecho de que la frecuencia de reloj de la red de interconexión influye tanto en su latencia como en su productividad. De esta forma, un tiempo de ciclo elevado implica una reducción en el *throughput* real que puede manejar la red y un incremento de la latencia de los mensajes a cargas bajas. En consecuencia, es lógico que el impacto sobre el sistema sea tan acusado. Nótese que la influencia puede ser muy superior a la correspondiente a variaciones estructurales en el encaminador.