Reducing the Interconnection Network Cost of Chip Multiprocessors

Pablo Abad, Valentín Puente, José Ángel Gregorio
Universidad de Cantabria
{abadp, vpuente, monaster}@unican.es

Abstract

This paper introduces a cost-effective technique to deal with CMP coherence protocol requirements from the interconnection network point of view. A mechanism is presented to avoid the end-to-end deadlock that arises from the dependency chains created at the network interfaces between the different message types handled by coherence protocols. Our proposal is designed to guarantee a fraction of end-to-end bandwidth for the highest priority messages and makes it unnecessary to employ several virtual networks or complex mechanisms for dealing with the limited capacity of the endpoint buffers. The presented approach uses the Rotary Router as its starting point, extending the original mechanism for the routing-dependent deadlock to the message-dependent deadlock. We also propose a solution that guarantees point-to-point message ordering in this router, which is a common requirement in some coherence protocols. Results for synthetic and parallel applications show that the proposal improves the performance of previous solutions with a much lower hardware cost.

1. Introduction

CMP systems usually assume the presence of some cache coherency mechanism in order to guarantee the communication between processors via shared-memory. The coherence protocol keeps caches transparent to the software, guaranteeing the coherence invariants at each cache block.

In order to ensure coherence invariants, each memory controller must react to each coherence transaction received from the processor or network, in some situations sending new messages to other controllers in the system. In consequence, a hierarchy of protocol messages must coexist in the network. The reactive nature of these different types of messages and the limited capacity of consumers can generate end-to-end deadlock. When a particular class of messages overflows the consumption queue in a node, the progression of subsequent actions at the memory controller cannot be guaranteed because the network cannot guarantee the delivery of the associated messages. The solutions for this problem vary from routing each network-traffic class through different virtual [14] [20] or physical networks [10]; extending the end-point queues into a main memory buffer [3]; employing some deadlock detection-recovery technique or a combination of detection and avoidance based on virtual networks[21][9]; or even, providing enough buffer space in each node's network interface message queues to hold at least as many messages as can be supplied, as in [22]. Today, the most common technique for dealing with this problem in CMP systems [19] or system level multiprocessors [14] is to employ different virtual networks for each kind of traffic. To be conscious of the cost of this technique, let us focus our attention in the Alpha 21364 router [14]. This router employs adaptive routing in a bi-dimensional torus network, which requires three virtual channels per physical link in order to avoid routing deadlock [7]. In addition, the coherence protocol employed requires seven different classes of messages, which makes it necessary to employ seven different virtual networks. In the end, the router employs 19 virtual channels per physical link. This configuration requires separate buffering and signaling for each channel, besides an increment in the arbitration cost even with a multiplexed crossbar. This might not be a problem in system level multiprocessor systems, such as the Alpha 21364, but it clearly is for a CMP system. In a CMP, this approach requires an extremely small buffer size if the implementation has to be feasible in terms of silicon area.

In this paper, we propose an effective solution for this problem, with an almost zero implementation cost. Starting from the Rotary Router [1], the proposed idea extends the originally designed to avoid routing deadlock to avoid also message-dependent deadlock. The idea is based on restricting the amount of resources available per router for each kind of traffic. The solution does not need to increase neither the number of virtual networks nor buffering resources of the original router, and it requires only some slight modifications in control logic. In contrast to conventional solutions, our idea allows a better buffer utilization. Thus, the implementation cost is reduced and the router performance could be improved.

Moreover, in some protocols, consecutive messages between two different controllers must arrive at their destination in the same order as they enter the network. This requirement is referred to as in-order delivery or point-to-point order. For example, in token coherence protocol, persistent request deactivation must arrive at destination after the activation [13]. In other situations, maintenance traffic must follow a predefined path from origin to destination [14][20]. The percentage of messages that must follow that order is extremely low and usually this is not a difficult problem. Deterministic routing solves this problem by construction, as the path between any origin-destination pair is fixed. In the adaptive routing case, the messages that require point-to-point ordering could be tagged at origin and the use of adaptive resources disallowed. Unfortunately, for the Rotary Router the solution is not so straightforward. The inherent buffering scheme makes it hard, at first sight, to achieve point-to-point ordering. However, we have found that it is possible to ensure that order with little effort and with low performance impact by restricting network resources availability combined with a port-to-port bookkeeping system.

The rest of the paper is organized as follows: Section 2 summarizes the operating mode of the Rotary Router. Section 3 explains the solution proposed to avoid message-dependent deadlock. Section 4 shows how the Rotary Router is capable of guaranteeing in-order delivery. Section 5 presents the performance results and, finally, Section 6 states the main conclusions of the paper.

2. Previous work

A router architecture able to deal with most of the requirements of CMP interconnection networks was proposed in [1]. The Rotary Router completely removes centralized structures, such as global arbiters or crossbars, is fully adaptive, and avoids the Head Of Line Blocking (HOLB). In addition, the Rotary Router is able to deal with some important requirements imposed by CMP environments, such as higher wire availability [4] and power and area efficiency. The capability to make such optimizations comes from the way packets are moved inside the router. The operation of the Rotary Router is based on two internal rings where packets circulate in opposite directions, looking for a suitable output port. This movement simplifies output arbitration and reduces contention. An outline of the Rotary Router can be seen in Figure 1. The router is built through the replication of three different structures named Input Stage, Output Stage and Buffering Segment Stage. Packets enter the router through the Input Stage, where pre-routing decisions are made. Additionally, the ring in which the packet will circulate is chosen depending on both the path to the closest profitable output port and the ring occupancy. After header update, packets start moving towards their output port through the Buffering Segment Stage. This stage is made up of Dual-ported FIFO Buffers (DFB) interconnected forming two independent rings and connected to each input and output port. Once a packet reaches a buffer connected to a profitable output port, there are two possible cases: i) if the packet loses arbitration, it will keep on circulating the ring until reaching another profitable output port; ii) if arbitration is won, the packet will move to that Output Stage of the router. This stage is in charge of sharing the only physical link between the packets coming from the two buffer rings.

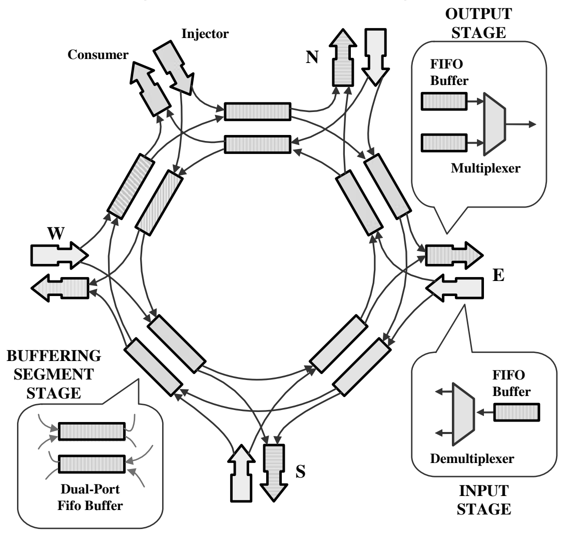

Figure 1. Rotary Router Sketch

Rotary Router avoids deadlock for any network topology, and performs adaptive routing without virtual channels. Different flow controls are applied to packets entering a buffer ring from a transit port or from an injection port. Control applied to in-transit packets limits the number of packets that can be in the router rings simultaneously, avoiding deadlock appearance inside the router. Control applied to injection ports guarantees the existence of enough holes in the network to move packets between nodes, which, added to the capacity of packets to do miss-routing, makes the network deadlock free. A formal proof of this claim can be found in [23].

The Rotary Router has proved to perform better than more classic structures as part of a CMP interconnection network. The performance advantage and power efficiency of this router is clear. However, for the performance study carried out in [1], for the sake of simplicity, message dependent deadlocks were solved by the network interface assuming unlimited storage capacity at consumption queues. In-order delivery was unnecessary in the evaluation framework employed. These assumptions do not invalidate the performance and energetic efficiency achieved, but for a practical usage of Rotary Router, these issues must be addressed.

3. Message-Dependent Deadlock

In a CMP cache coherence protocol, the messages involved in a memory transaction depend one upon the other, because the generation of some message types is a consequence of the delivery of other types. For example, when a cache controller sends a message requesting a cache line, this causes at least the answer from another cache controller providing that block. Therefore, there is a second message *subordinate* to the first one. This relationship is known as the message dependency chain [21]. This dependency between messages can cause the appearance of a deadlock other than the routing deadlock. This new kind of deadlock, known as message-dependent deadlock or end-to-end deadlock, appears at the endpoints of the interconnection network because of the limited capacity of the consumption queues. As a simple example, consider an interconnection network with a request-reply protocol. If an application exhausts network resources with request messages due to consumption queue overflow, reply messages will not be able to make progress, stopping the processing of pending requests at consumption queues and blocking therefore traffic advance. This cyclic dependency could lead to a message deadlock. Coherence protocols will have deeper dependencies among messages (longer dependency chains) than a request-reply communication protocol, because depending on the state which a cache line is in, different operations will be performed in order to read/write the data. As an example, the communication protocol in [14] has a dependency chain of seven messages, which means that some operations need seven messages to complete.

Most of the previous methods used to avoid message-dependent deadlocks are mainly based on the replication of traffic paths via the inclusion of extra hardware resources, such as extra virtual or physical networks able to break resource and message dependencies [6][9][14][20]. Different message types travel through different hardware resources. This way, different message[1] traffics never can block each other. The complete avoidance of message-dependent deadlocks requires a number of replications equal to the dependency chain length. The additional hardware resources increase network area and arbitration complexity and obviously, this could have a significant performance and cost impact. In fact, in some real machines, like the SGI Origin 2000, with the aim of reducing the number of necessary virtual networks imposed by the protocol, a detection-recovery technique is employed to reduce the three virtual networks imposed by the protocol to only two. In other cases, like the solution adopted in the Alpha 21364 router where seven virtual networks are employed solely for the purpose of eliminating this anomaly, the importance of the problem can be clearly appreciated.

Fortunately, the special Rotary Router characteristics, summarized in the previous Section, provide the opportunity to deal with this problem without requiring extra hardware resources. This technique can only be applied if the buffering strategy allows packet overtaking inside the router, as occurs in the Rotary Router.

3.1 End-to-end Deadlock avoidance for Rotary Router

The Rotary Router is free of Head of Line Blocking because a blocked message cannot indefinitely delay the access of other messages to an available output port. The same property, adequately employed, will allow us to avoid message-dependent deadlock without hardware replication. This method, named as *filling control*, is explained below.

It could seem that in order to reserve exclusive buffering resources for each message type the Rotary Router would need the inclusion of two additional DBF rings per type. But it is not the case. To avoid buffering replication, every message type is forced to move through the same buffering space, and message advance is guaranteed through the control of the cumulative amount of resources used in all the

[1] *As we are employing cache coherent protocol, each message is composed of just one packet (a command or a command plus a cache block) and consequently we will indistinctly employ the terms "message" or "packet"*

buffering space. The filling control method does not supervise the buffers used by a message type, but rather the fraction of the whole buffering resources in use. The amount of buffering used by each message type will be limited, in accordance with the priority of the traffic. Each subordinate message is able to occupy a bigger portion of area than its predecessor. The top class or terminating class of traffic will be able to utilize all the buffering space in the network, with the exception of the resources required to guarantee that the network is routing-deadlock free. This way, every time a message tries to access any buffer ring, the number of messages which are already in both router rings is checked. If this number is below the limit imposed for a particular message type and router input port, the message is allowed to enter the ring. Otherwise, the message must wait until the necessary amount of buffering resources becomes available.

Let us explain the method with an example. Assume we have a protocol with four different message types, m_1, m_2, m_3 and m_4, being m_1 the first message type in the dependency chain and m_4 the terminating one. Filling control will restrict the first message type m_1, only allowing it to fill up to 25% of a router's buffering resources. When a router exceeds this limit, the situation is communicated to the injector and to the neighboring routers, through specific stop protocol lines. From that moment, no more m_1 messages are allowed to enter in the router from any input port (injection or transit). m_1 subordinate messages (m_2) should be able to advance even if m_1 messages are stalled due to an overflow of m_1 at any consumption queue. In a similar way, m_2 should not be allowed to occupy the whole buffering space, because there are still two kinds of messages with higher priority. For this reason, it is safe for m_2 messages to make use of up to 50% of ring buffering. The limit for the third message type will rise 75%, and finally, the terminating message type will be the only one allowed to make use of 100% of buffering resources (obviously, this 100% does not include the buffering resources needed by the routing-deadlock avoidance method). Terminating messages do not generate new messages, therefore they will not block any other message.

However, applying the same occupancy limits to injection and in-transit ports could give rise to deadlock situations. For instance, if every router in the network reaches the limit to inject m_1 messages simultaneously, this class will not block subordinate messages, but it will not be able to advance. To circumvent this situation, a solution equivalent to the one used in routing-deadlock avoidance mechanism has been adopted. In order to reach a situation in which every router in the network has reached its limit to inject m_1 messages, last messages must come from an injection queue (in-transit queues move messages between routers, but do not increase the total amount of messages in the network). Applying a harder limit to injection queues we ensure that we will never reach the situation described before. This way, in a worst case scenario at least one of the routers will have an occupation level lower than the limit for m_1 messages.

Notice that this limit difference must be applied to each message type independently. For example, if we had a network with two message types (m_1 and m_2) and a router capacity of N messages, in-transit queue limits will be $N/2$ for m_1 messages and N for m_2 , while injection queue limits will be $N/2\text{-}\delta$ and $N\text{-}\delta$. The δ extra holes generated by the method avoid blocking situations.

The filling control mechanism does not require data-path replication to deal with message-dependent deadlocks, but some control logic needs to be added. This mechanism has to be applied both to the messages in transit through the network and to the messages injected from a coherence controller. For this reason, this flow control has to be handled by a centralized structure per router. This structure, made up of a counter for each message type, will be in charge of sending stop signals to the injector and to the neighboring routers. Counters update is performed in two phases; first, every new message entering the input stage or at the injector will be checked and the appropriate counter will be incremented. This operation is performed in parallel with the pre-routing and ring selection process. In the second phase, packets leaving the router rings are also checked, and counters decremented. This operation is performed when the message is in the output stage. Once every counter has the proper value, flow control signals are generated and sent to the injector and neighboring routers. These control signals are processed by the control logic of the DFBs in the neighboring router in order to make the message advance inside the ring or to the next router.

It should be noted that this approach requires a modest amount of hardware: merely one counter per class of traffic, some modification in the DFB control and to increase the wiring between routers with a stop signal per message type. In contrast, conventional solutions need to multiply the virtual channels per link by the number of messages types. This implies additional buffers per physical channel and complexity increment in crossbar arbitration. Besides, independent inter-router protocol lines per traffic class are required.

Moreover, our approach tends to favor the advance of high order messages type, being the traffic with the biggest advantage the terminating traffic. For a coherence protocol, this behavior is the most desirable, and in contrast to our router, conventional routers must prioritize artificially the traffic at crossbar arbitration [14], increasing its cost.

3.2 Correctness in Corner-case situations

To check the correctness of the proposal, specific synthetic traffic tests have been developed for emulating the environment in which message-dependent deadlocks are likely to occur.

Basically, a reactive traffic pattern is applied to the network and after the steady state network consumers will be enforced to stop accepting one of the message types, emulating the behavior of a network interface with finite resources. From that moment, no more messages subordinated to the type stopped will be generated. The filling control method must guarantee the delivery as soon as possible of the subordinate messages which are actually in transit or waiting at injection queues.

The main network parameters will be the same for every simulation. The topology considered will be a 2-Dimensional torus with 64 nodes (8×8). For this experiment, every message-type will have a fixed length of 5 phits. The router buffering space will be kept constant at a value of 60 packets per router. Traffic analyzed will be reactive with uniform destination pattern. The simulator generates the first message type in the dependency chain at a rate of one phit/cycle/router. The generation of the rest of subordinate messages is done automatically upon completion of servicing messages at end nodes. Six different protocols will be simulated, each of them with a different dependency chain length, varying from two to seven. Each protocol will be evaluated, stopping the consumption of different message types at a fixed simulation cycle. For every simulation, we will analyze the throughput obtained for the last class of messages in the dependency chain. To do so, the time the last terminating message takes to reach its destination will be measured. Simulations were repeated several times with different seeds for traffic generation, and for every simulation done, every terminating message reached its destination. If a deadlock had happened, some terminating messages would have never reached their destination. The results of the experiment are shown in Figure 2. The X-axis represents the message type stopped for each simulation and the Y-axis represents the throughput of terminating messages remaining in the network, normalized to each protocol's total throughput. Note that while for a 7-type protocol we can stop 6 different message types, for a 2-type there is only one message type that can be stopped. As can be seen in the graph, terminating messages throughput remains constant independently of the message type stopped. As the protocol has a higher number of message types, less buffering space is assigned to each type and obviously terminating traffic will have lower throughput, but every message belonging to the highest priority type arrives at its destination.

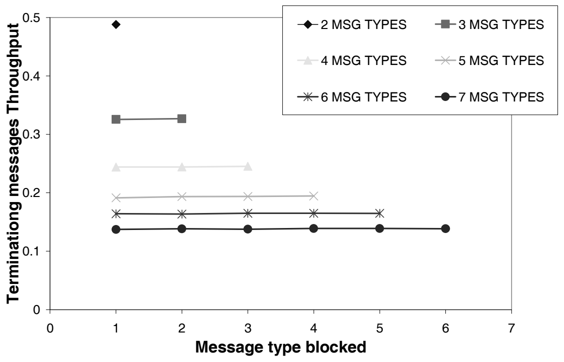

Figure 2. Terminating messages throughput when consumers of intermediate message classes are artificially blocked

4. In-Order delivery support

Some memory coherence protocols or maintenance tasks require in-order delivery support for a small fraction of the network traffic. Fulfilling this requisite is extremely simple for input buffered routers. A specific routing algorithm can be applied to ordered messages, forcing them to follow a fixed path to destination. As the buffers in this type of routers do not allow packet reordering inside each router, in-order delivery will be guaranteed. The Rotary Router can forward packets between different router buffering elements, which imply a chance of reordering. Consequently, in the Rotary Router some special actions need to be taken to ensure the correct order at packet delivery.

The routing deadlock avoidance mechanism in the Rotary Router is based on the ability of packets to be miss-routed and on the injection restriction. In-order messages cannot make use of that approach, so they need a different methodology to avoid routing-deadlock. As the Rotary Router does not divide network resources into multiple virtual channels, the routing algorithm chosen for in-order messages has to be deadlock free when no virtual channels are available. We could guarantee in-order delivery for that traffic if the network resource availability is

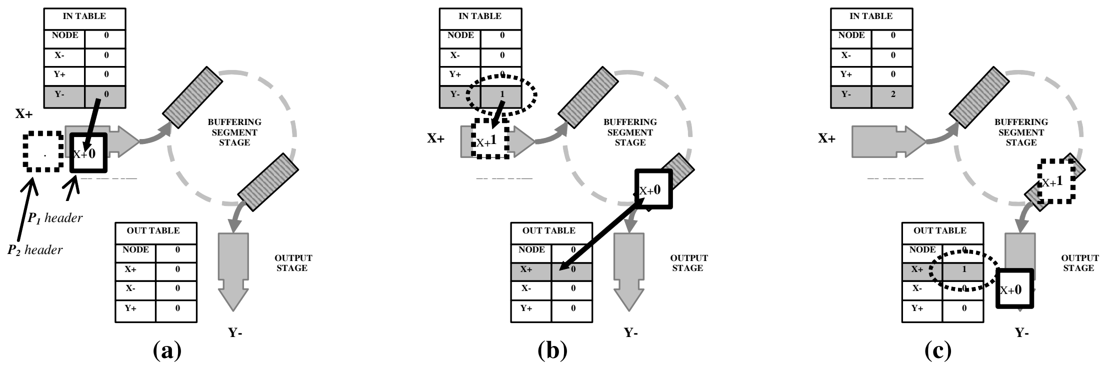

Figure 3. In-order routing, with two packets from the same traffic class. (a) Entering packet p1 exit index is updated using input table value and X+ input port (b) Input table is updated, subsequent packet p2 exit index is updated. Exit index of p1 is compared against output table value for packet input port, (c) p_1 is allowed to leave the router, next exit index in output table is updated for X+ inputs after the message leaves the ring.

restricted to a sub-topology. Although the Rotary Router is topology agnostic, in this work we will assume a suitable topology for CMP systems, such as bi-dimensional torus. Under these circumstances, it is enough that the in-order messages do not make use of the wraparound links and that they must be routed following a strict dimension order, to guarantee deadlock freedom for in-order traffic. Although sub-optimal, in general it is possible to keep the in-order delivery implementation topology agnostic, employing up/down* routing algorithm.

In an input buffered router, messages traveling in opposite directions of the same dimension use independent resources. However, in the Rotary Router every direction and dimension can share the buffering space (originally, there is no restriction to using both internal rings). This could produce a deadlock between two neighboring routers if a router is full of in-order traffic and all the messages try to advance towards a neighbor where the same situation is happening. To avoid this circumstance, the two buffer rings inside the Rotary Router will be used for in-order traffic flowing in opposite directions. The selection of the router ring in which a packet will advance will be based on the packet direction. Packets traveling in opposite directions will make use of different rings, never sharing the same buffering space. This way, this kind of deadlock is avoided because no cyclic dependences between buffers will occur.

Under the above conditions, it can be ensured that packets with the same source and destination will advance through the same path. But we still need to ensure that packets do not change their order while moving inside the buffer rings of the routers. A mechanism based on table lookup is proposed for this purpose. The operation of this mechanism is shown in Figure 4. Each input stage will hold a small table with the order value (named *exit index*) for each output of the router. Once pre-routing is computed and it is determined that the packet is tagged for in-order delivery, the table value corresponding to the profitable output port is copied to the header of the packet, and the value in the table is incremented by one unit. This value at the header of the packet indicates the exit order. At every output stage, there will be a complementary table with information about the exit index expected for the next in-order packet that must leave the router, coming from each input port. In this way, when a packet is being arbitrated at a DFB, the index at the header of the packet p_i must coincide with the value in the output stage table corresponding to the input port where the packet came from. If both fields match, the packet p_i can leave the router ring, and the corresponding output port table will be updated. Otherwise, the other packet p_j that entered the ring before p_i is still in the ring and consequently p_i must be kept on in the current router, and will be forwarded to the next DFB of the ring, even when its profitable output port is available.

Note that this ordering method does not order the packets according to the pair source-destination, but to their router input-output ports. This can introduce unnecessary delay to some packets, because they have to maintain the order with respect to packets from different traffic flows. The main reason for applying this method is to minimize implementation cost. If packets are classified according to their source and destination, table sizes will be increased in an unsustainable way with the network size. With this method, tables size is proportional to the number of input ports of the router. As an example, if we have a router with five input ports and a buffering space of 60 packets in the rings, we will only need ten tables with four rows each (it is not possible for an in-order packet to leave the router through the entering port). The exit index on each row does not need to grow indefinitely; it is enough to let it reach a value equal to the buffering space (in packets). Thus, for this example each row would only need six bits to store a maximum value of 60. Therefore, each table needed would have a size of 24 bits (30 bytes per router).

The Rotary Router was not conceived to deal with in-order packet routing, and therefore its performance falls when this kind of traffic predominates. For these packets, the output port accessibility is much more challenging than in conventional traffic, and can produce router performance degradation for in-order traffic. This method is only useful when in-order packets represent a small portion of the whole network traffic. Under this condition, the method guarantees in-order delivery without using solutions with a higher implementation cost. In-order packets are able to share resources with the rest of the packets, only routing and arbitration needs to be modified.

Note that the combination of this method with the end-to-end deadlock avoidance mechanism is not completely orthogonal. This means that if we have in-order traffic belonging to different classes of traffic, we need to expand the size of the tables, maintaining an exit index per class (the cost increment is proportional to the number of message classes that could require in-order delivery). Otherwise, low priority in-order traffic can block point-to-point ordered packets with higher priority, which can generate deadlock. Each class of traffic must be in-order only with packets belonging to its own class. In most cases, the in-order traffic is restricted to only one or a few message classes [13][14].

5. Performance Evaluation

The solutions presented for message-dependent deadlock and in-order routing must be evaluated to see how they affect performance. For this purpose, three counterpart router architectures were compared against the Rotary Router. The first one was the Adaptive Bubble Router (BADA) [17]. This router will have higher buffer requirements than the rest of the routers, because it needs two virtual channels to avoid routing-deadlock. Adaptive Bubble Router has proved to be a good off-chip proposal and in fact its deadlock avoidance methodology is used in the interconnection network of the BlueGene/L supercomputer [2]. The second router evaluated was the Deterministic Bubble Router (BDOR). This router shares the same flow control as the one above, but does not perform adaptive routing, making unnecessary the employment of virtual channels to avoid deadlock. Finally, the last router evaluated (LOW-LAT) makes use of speculative arbitration and buffer bypass to achieve a base latency of a single cycle, similar to the router presented in [15]. This is a deterministic router, where packets traveling in the same dimension have a latency of one cycle, while injected, consumed or turning packets have to go through the whole router pipeline. As link traversal is also made in the same cycle, in order to achieve a reasonable cycle time this last router is only suitable for topologies with low link lengths, such as mesh networks. Message-dependent deadlocks are avoided in the three routers using different virtual channels for each message type. In-order messages do not need special actions in the second and third routers. For the Adaptive Bubble Router, in-order messages will not be routed through adaptive paths. To carry out this evaluation, no implementation based on recovery methods for message dependent deadlock, such as [21], was employed because this method cannot support the in-order delivery required by the real system evaluation carried out in subsequent sub-sections.

5.1 Synthetic Scenario

In this first phase of the evaluation, we will show the effect of synthetic traffic on network performance. The tool used for this evaluation will be the interconnection network simulator SICOSYS [18]. Two different networks have been considered for the experiment. The topology chosen for LOW-LAT router is an 8x8 mesh, and 8x8 torus for the rest of the routers.

As we are trying to emulate a memory coherency protocol, we will simulate a workload in which data transactions are started at every injector and measure the time required to complete a fixed number of transactions. This will provide more useful information about how the network will behave when making a full system evaluation. Data transactions will be generated as follows; the simulator generates the first message type in the dependency chain at a rate of 1 phit/router/cycle, and subordinate messages are generated automatically as messages arrive at their end nodes. Messages are classified in classes and these are numbered (starting from 1) according to their position in the dependence chain. The packet size selected for the experiment has a bimodal distribution, being 5 phits (128 bits links and 64+16 bytes messages) for messages belonging to odd classes and 2 phits for those belonging to even classes (128 bits links and 16 bytes messages). To mimic the reactive characteristic of the traffic, destination of an odd-class message is chosen according to the pattern traffic selected, whereas those messages from even classes are sent back to the sources from which the originating messages were received.

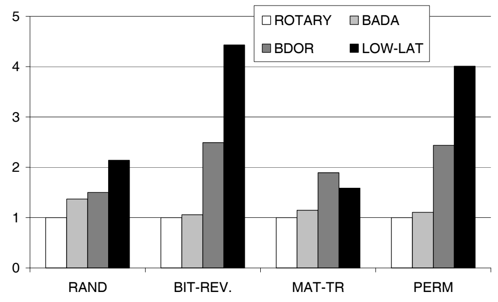

Figure 4. Normalized required time for consuming 32,000 terminating messages.

The first experiment is designed to observe the raw performance of all routers. Assuming only three message classes, synthetic patterns Random, Transpose Matrix, Perfect Shuffle and Bit Reversal [8] are employed for the destination of class-1 and class-3 messages. Results have been obtained simulating the traffic triggered by 32,000 messages (500 from each router) belonging to the class 1 and similar buffer capacity is assumed in each router. The BADA router has 15 phits FIFO queues, BDOR and LOW-LAT routers have 30 phits per FIFO. In the Rotary Router, buffer capacity is 20 phits in DFB and 10 phits in each one the input and output stages. In this way, the total storage capacity per router is 320 phits in conventional routers and 300 phits in the Rotary Router. The sizes chosen are those where no significant improvements in throughput were observed for larger sizes.

As can be seen in Figure 4, the Rotary Router reduces the time required to finish every data transaction for almost all traffic pattern analyzed. Both adaptive routers obtain close results, the two deterministic ones employ up to four times more execution time in the presence of non-uniform traffic patterns. The differences between the BDOR and the LOW-LAT router are mainly caused by the different network topology chosen for each router. The small link length restriction present in the LOW-LAT router harms its performance.

The second experiment is designed to show how each router performs when the number of message classes vary form 2 to 4, keeping the router storage capacity constant at 300/320 phits. In order to meet this requirement, we must modify the phits per FIFO to 20 (2 classes) and 10 (4 classes) phits respectively in BADA and 40 (2 classes) and 20 (4 classes) phits for BDOR and LOW-LAT. No change is required in the Rotary Router storage distribution. We are considering no implementation cost increment in any of the routers, which is true for the Rotary Router but false in the conventional routers, harming our proposal in this comparison. Anyway, even without taking into account this important fact, the performance advantage of the Rotary Router, as it can be seen in Figure 5, is clear. For every traffic pattern and every protocol analyzed, the Rotary Router obtains better throughput results. In addition, the number of message types in the network does not affect Rotary Router performance, and maximum throughput remains nearly constant. Deterministic routers also maintain their maximum throughput levels constant, but at much lower levels. The BADA router suffers important throughput losses when moving from 3 to 4 message types. The reason behind this behavior is the reduction of buffer size. As buffering space was kept constant, buffers in the BADA are extremely small (2 packets) for a 4-types protocol, which reduces router throughput, being even worse than the deterministic routers in some cases.

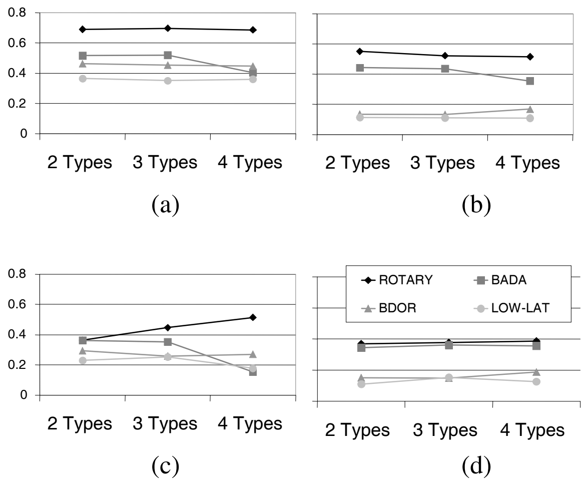

Figure 5. Normalized Maximum Sustained Throughput for different number of types

5.2 Real Scenario

In this section, we will show the effect of the message dependent deadlock avoidance mechanism and in-order packet routing under a realistic situation. For this purpose, the complete system simulator Simics [11] will be used, extended with the GEMS timing infrastructure [12]. GEMS provides detailed models of both the memory system and a state-of-the-art processor. SICOSYS has been integrated into the simulator GEMS, replacing its original network simulator. The simulated system is a 16-processor

CMP with shared S-NUCA L2 based on [5]. The protocol, based on Token Coherence [13], requires a hierarchy of five classes of messages to be implemented. In this protocol, persistent request activation and deactivations must be point-to-point ordered. In this way, we will expose the advantages and correctness of our proposal in terms of performance, with a large number of message classes, and correctness, requiring in-order traffic. Main parameters of the simulated system are shown in Table 1.

Table 1. Main simulation parameters

Number of Cores	16
Window Size / outstanding req. per CPU	256 / 16
Issue Width	4
L1 I/D cache	Private, 32KB, 2-way, 64Bytes block, 1-cycle
Direct Branch Predictor	4KB YAGS
Indirect Branch Pred.	256 entries (cascaded)
L2 cache	SNUCA, token coherence protocol, 16x16 banks, 4 per router
L2 cache bank	128KB, 16-way, 3-cycles, Pseudo LRU, 64 Bytes block
Main Memory	4GB, 260 cycles, 320 GB/s
Command size	16 bytes
Network Topology	8×8 torus
Network Link	128 bits / 1 cycle latency

The applications considered in this study are four transactional and three scientific workloads. The server workloads used are a Static Serving Web server benchmark based on SURGE running on top of an Apache web server (HTTP1) and Zeus web server (HTTP2), SPECjbb2000 (Java), and an online-transactions processing TPC-C like benchmark (OLTP). The numerical workloads used are LU, FT and IS from NAS Parallel Benchmark, using the OpenMP implementation, Version 3.3. In all applications, a variable number of runs are performed with pseudo-random perturbation in access memory times in order to estimate workload variability.

Figure 6 presents the results with expected average execution time. The confidence interval is 95%. As can be seen, the Rotary Router outperforms the rest of the routers in all the applications simulated. The adaptive and deterministic Bubble Routers obtain similar results for every application. This means that each virtual network is not stressed enough to take advantage of adaptive routing. The LOW-LAT router exhibits a very low latency under low traffic conditions. As can be seen in some applications, especially server based ones, it outperforms bubble routers, because of its smaller low load pipeline length. Notwithstanding, with other applications this router has a poor performance. This behavior is mainly due to the topology employed (it is the only mesh topology) and routing simplicity. Both factors will cause a low maximum achievable throughput. It should be pointed out here that in the experiments carried out, the LOW-LAT router has an unfair advantage because we are assuming the same cycle time for all the routers. For example, according to [1], the Rotary Router cycle time will be approximately 20FO4 whereas LOW-LAT router will have 35FO4 [16].

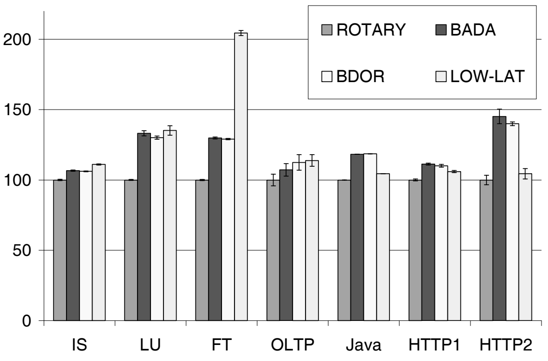

Figure 6. Normalized execution time of real workloads.

The reason behind the better results of the Rotary is the lower latency of the protocol messages with highest priority. Due to the special flow control of in-transit messages in the Rotary Router, messages with high priority advance faster, because they can occupy a bigger portion of router buffering resources. In the case of input buffered structures, an amount of buffering space is fixed for each message type, so latencies are not able to adapt to message priority. In some cases, the advantage is close to 30% with respect to the closest alternative.

6. Conclusions

In this work, we have presented efficient solutions for well-known network problems. The mechanism designed to alleviate the HOL blocking in the Rotary Router has been extended to implement a mechanism able to deal with message-dependent deadlocks without hardware replication. This proposal allows us to keep router complexity constant, independently of the number of message classes of the coherence protocol and we consider this to be of great importance for current and future CMP architectures. In addition, a solution for point-to-point ordering was presented. An efficient mechanism able to deal with in-order messages with few control logic add-ons has been developed. The idea allows ordered messages to use the same resources as the rest of message types.

Performance results from a wide range of loads demonstrate that the mechanism used for deadlock avoidance presents advantages over conventional approaches. The buffer utilization of our proposal is more flexible, which implies a performance boost and messages with a higher priority travel faster through the network, thus accelerating application execution times.

7. Acknowledgements

This work has been supported by the Ministry of Education and Science of Spain, under contracts TIN2004-07440-C01-01 and TIN2007-68023-C02-01 and by the HiPEAC European Network of Excellence.

8. References

[1] P. Abad, V. Puente, P. Prieto, J.A. Gregorio, "Rotary Router: An Efficient Architecture for CMP Interconnection Networks", International Symposium on Computer Architecture (ISCA), 2007.

[2] N.R. Adiga, et al., "An Overview of the BlueGene/L Supercomputer", Supercomputing 2002.

[3] A. Agarwal, R. Bianchini, D. Chaiken, K. Johnson, D. Kranz, J. Kubiatowicz, B.H. Lim, K. Mackenzie, D. Yeung, "The MIT Alewife Machine: Architecture and Performance", International Symposium on Computer Architecture (ISCA), 1995.

[4] J. Balfour, W. Dally, "Design Tradeoffs for Tiled CMP On-Chip Networks", International Conference on Supercomputing (ICS) 2006.

[5] B. Beckmann and D. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches", 37th International Symp. on Microarchitecture (MICRO), December 2004.

[6] D. Burger, S. Keckler, K. McKinley, M. Dahlin, L. John, C. Lin, C. Moore, J. Burrill, R. McDonald, W. Yoder "Scaling to the end of Silicon with EDGE Architectures" IEEE Computer. Volume 37, No 7, pp.44-55, July 2004.

[7] W. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", Design Automation Conference (DAC) 2001.

[8] W. Dally, B. Towles, "Principles and Practices of Interconnection Networks". Morgan Kaufmann, 2004.

[9] J. Laudon, D. Lenosky, "The SGI Origin: A ccNUMA Highly Scalable Server", International Symposium on Computer Architecture (ISCA), 1997.

[10] D. Lenoski et al., "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", International Symposium on Computer Architecture (ISCA), 1990.

[11] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, F. Larsson, A. Moestedt, B. Werner, "Simics: A Full System Simulation Platform". Computer, Vol. 35, No.2, pp. 50-58, February 2002.

[12] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, D. Wood, "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset", SIGARCH Comput. Archit. News, Vol.33, No.4, pp.92–99, November 2005.

[13] M. Martin, M. Hill, and D. Wood, "Token Coherence: Decoupling Performance and Correctness", International Symposium on Computer Architecture (ISCA), June 2003.

[14] S. Mukherjee, P. Bannon, S. Lang, A. Spink, D. Webb, "The Alpha 21364 Network Architecture", IEEE Micro, vol. 22, no. 1, pp 26-35, Jan-Feb 2002.

[15] R. Mullins, A. West, S. Moore "Low-Latency Virtual-Channel Routers for On-Chip Networks", International Symposium on Computer Architecture (ISCA), 2004.

[16] R. Mullins, A. West, S. Moore, "The design and implementation of a low-latency on-chip network", ASP-DAC 2006

[17] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo, J.M. Prellezo, "The Adaptive Bubble Router", Journal of Parallel and Distributed Computing, Vol. 61, No. 9, September 2001.

[18] V. Puente, J.A. Gregorio, R. Beivide, "SICOSYS: An Integrated Framework for studying Interconnection Network in Multiprocessor Systems", Euromicro Workshop on Parallel and Distributed Processing, 2002.

[19] K. Sankaralingam et. al. "Distributed Microarchitectural Protocols in the TRIPS Prototype Processor", International Symposium on Microarchitecture (MICRO), 2006.

[20] S. Scott and G. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus", Hot Interconnects IV, August 1996.

[21] Y.H. Song, T.M. Pinkston, "A Progressive Approach to Handling Message-Dependent Deadlock in Parallel Computer Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 14, No. 3, pp 259-275, March 2003.

[22] C.B. Stunkel et al. "The SP2 High-Performance Switch", IBM Systems J. Vol. 34, No. 2, pp. 185-204, 1995.

[23] http://rotaryformalproof.googlepages.com/publ_109.pdf