Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 95859, 10 pages
doi:10.1155/2007/95859

Research Article

Avoiding Message-Dependent Deadlock in Network-Based Systems on Chip

Andreas Hansson,[1] Kees Goossens,[2,3] and Andrei Rădulescu[3]

[1] *Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[2] *Computer Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2600 GA Delft, The Netherlands*
[3] *SOC Architectures and Infrastructure, Research, NXP Semiconductors, 5656 AE Eindhoven, The Netherlands*

Received 16 November 2006; Accepted 6 February 2007

Recommended by Maurizio Palesi

Networks on chip (NoCs) are an essential component of systems on chip (SoCs) and much research is devoted to deadlock avoidance in NoCs. Prior work focuses on the router network while protocol interactions between NoC and intellectual property (IP) modules are not considered. These interactions introduce *message dependencies* that affect deadlock properties of the SoC as a whole. Even when NoC and IP dependency graphs are cycle-free in isolation, put together they may still create cycles. Traditionally, SoCs rely solely on request-response protocols. However, emerging SoCs adopt higher-level protocols for cache coherency, slave locking, and peer-to-peer streaming, thereby increasing the complexity in the interaction between the NoC and the IPs. In this paper, we analyze *message-dependent deadlock*, arising due to protocol interactions between the NoC and the IP modules. We compare the possible solutions and show that deadlock avoidance, in the presence of higher-level protocols, poses a serious challenge for many current NoC architectures. We evaluate the solutions qualitatively, and for a number of designs we quantify the area cost for the two most economical solutions, *strict ordering* and *end-to-end flow control*. We show that the latter, which avoids deadlock for *all* protocols, adds an area and power cost of 4% and 6%, respectively, of a typical Æthereal NoC instance.

1. INTRODUCTION

Networks on chip (NoCs) have emerged as the design paradigm for design of scalable on-chip communication architectures, providing better structure and modularity while allowing good wire utilisation through sharing [1–4]. By providing *services* for intermodule communication [5] over a mix of different sockets, NoC enables intellectual property (IP) reuse [3, 6, 7] and enhances system-level composability [6]. The services must be implemented robustly and efficiently.

Deadlock is catastrophic to as SoC and a serious threat to the robustness of the communication services offered by the NoC. Therefore, the importance of deadlock-free operation is stressed as a key research problem in NoC design [8] and much work is focused on providing deadlock-free routing in NoCs [9–11].

Deadlock freedom in the router network, henceforth just network, relies on the *consumption assumption* [12]: the network accepts and delivers all messages sent by the network interfaces (NIs) as long as they promise to consume all messages from the network when they are delivered. Routing algorithms that rely on this assumption, which to the best of our knowledge is true for all nonlossy routing algorithms currently used in NoCs, are still susceptible to deadlock arising from protocol interactions in the NIs. The IP blocks create *message dependencies* between buffers in the NIs that, when transferred to the router network, can lead to *message-dependent deadlocks* [12].

The SoC comprises IP modules with two different types of ports: masters (initiators) and slaves (targets) [3]. Masters initiate *transactions* by issuing *requests*. One or more slaves receive and execute each transaction. Optionally, a transaction also includes a *response*, returning data, or an acknowledgement from the slave to the master. This transaction model subsumes both a distributed shared memory (DSM) and message passing (MP) communication paradigm. As we will see, this model of on-chip communication can lead to *four types* of message dependencies, *request-response*, *response-request*, *request-request*, and *response-response*, depending on the behavior of the IP modules.

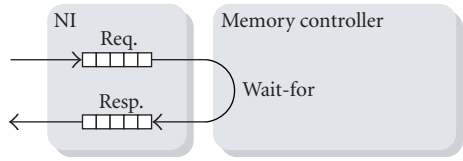


Figure 1: Request-response dependency at a memory.

These dependencies arise as a consequence of the IP modules' *desired* behavior. For example, the memory controller in Figure 1 is expected to respond to requests, and thus creates a request-response dependency.

Even when NoC and IP dependency graphs are cycle-free in isolation, put together they may still create cycles due to these dependencies. Traditional NoC architectures rely solely on request-response protocols, and consequently only have to address these dependencies. However, higher-level protocols are being adopted in emerging SoCs for cache coherency [13, 14], slave locking [14, 15], and peer-to-peer streaming [16]. These higher-level protocols introduce additional dependencies that must be addressed to provide deadlock-free operation.

The main contribution of this paper is an analysis of the message-dependent deadlocks that commonly used programming models and coherency schemes can cause network-based SoCs. We evaluate the possible solutions and show that many NoCs do not consider [6, 17–20] or only partially solve [21–23] the problem. These NoCs can only guarantee deadlock-free operation for a limited set of protocols. Furthermore, we show how the Æthereal [3] and FAUST [24] NoC, both employing *credit-based end-to-end flow control* [25], avoid message-dependent deadlock irrespective of the communication protocols used. Alternative approaches, for example multiple networks, have not been shown for NoCs.

For a number of designs, we quantify the area cost for the two most economical solutions, *strict ordering* and end-to-end flow control. We show that the latter, which avoids deadlock for *all* protocols, has an area and power costs of 4% and 6%, respectively, of a typical Æthereal NoC instance.

Related work is introduced in Section 2. The architectural platform is presented in Section 3. Next, the problem is introduced in Section 4 and the different message dependencies are covered in depth in Section 5. Solutions used in NoCs are presented in Section 6. An evaluation of the different solutions is given in Section 7 together with a quantitative analysis of the two prominent techniques, strict ordering, and end-to-end flow control, applied to Æthereal. Finally, Section 8 concludes the paper and presents directions for future work.

2. RELATED WORK

Key research problems in NoC design are presented in [8]. The authors stress the importance of deadlock-free operation but identify it only as a routing problem, not considering the protocol interactions between the IPs and the NoC at the network endpoints.

Deadlock recovery is a popular resort in parallel computers [12] and is used in the Proteo [26] NoC that drops packets on overflow. The majority of NoCs [3, 6, 17–24], however, avoid deadlock, as deadlock detection and recovery mechanisms are expensive [8] and complicate the provision of guarantees. Deadlock avoidance is also the focus of this paper.

An NI that offers high-level services is presented in [3]. End-to-end flow control, important as we will see in Section 6.2, is part of the basic functionality offered by the design and the added bandwidth for an MPEG-2 decoder is evaluated. However, as with [20, 24] that also use end-to-end flow control, message-dependent deadlock is not discussed.

Many NoCs [21–23] break request-response dependencies by introducing separate physical networks for the two message types. Virtual, instead of physical, networks are used in [27, 28] to avoid deadlock in a higher-order configuration protocol and a forwarding multicast protocol, respectively. All the solutions are protocol-specific and none address the dependencies that can arise when IPs have both master and slave ports.

The possibility of considering message types in the topology synthesis is explored in [29]. The work presents a methodology that tailors the NoC to a particular application behavior while taking message-dependent deadlock into account. In contrast to what we advocate in this work, the NoC architecture is inherently coupled to the application and assumes that the NoC can be redesigned if the application or its binding to the NoC should change.

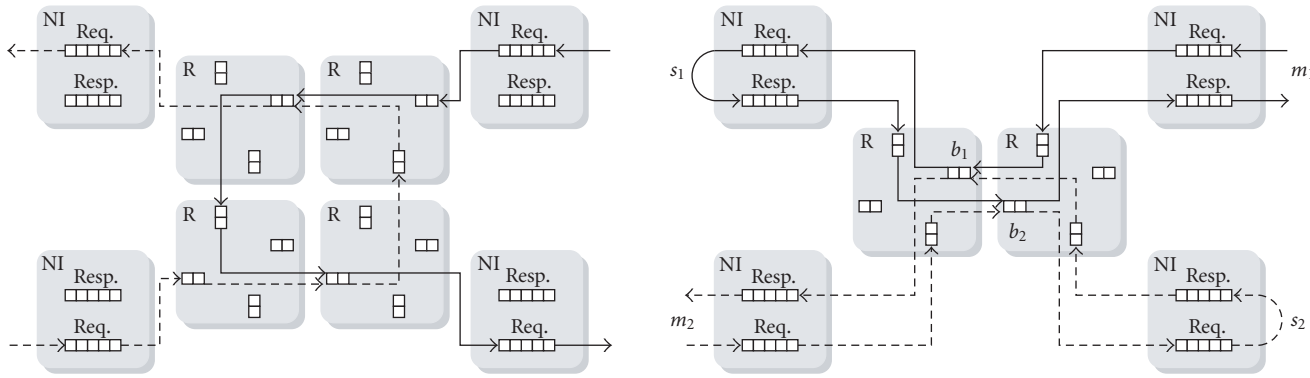
A comprehensive survey on methods for handling message-dependent deadlocks in parallel computer systems is given in [12]. In contrast with the computer networks and multiprocessor environments studied in the work, NoC storage and computation resources are relatively more restricted, and the protocol stack is entirely implemented in hardware. Hence, design constraints and optimization goals are fundamentally different.

In this work, we present the implications regarding deadlock that arise in a network-based SoC due to the *interactions between the NoC and the IP modules*. Furthermore, we evaluate the area and power cost of a NoC architecture, applied to a number of representative SoCs, that avoid *all* potential message-dependent deadlocks through the use of credit-based end-to-end flow control.

3. ARCHITECTURAL PLATFORM

We assume that NoCs comprise two components: routers (R) and network interfaces (NIs), as depicted in Figure 2. The routers can be randomly connected amongst themselves and to the NIs, that is, there are no topology constraints, although the routing is assumed to be deadlock-free. The routers transport packets of data from one NI to another.

The NIs enable end-to-end services [3] to the IP modules and are keys in decoupling computation from communication [6, 7]. The NI allows the designer to simplify communication issues to local point-to-point transactions at IP module boundaries, using protocols natural to the IP [7], for example, AXI [15] or OCP.



(a) Routing-dependent deadlock

(b) Message-dependent deadlock

Figure 2: Different levels of deadlock.

Master and slave IP ports are connected to slave and master NI ports, respectively. The term *connection* is used throughout this paper to denote a unidirectional peer-to-peer interconnection between a master and a slave, either carrying requests from master to slave, or responses from slave to master, but not both. In Section 6.4, we return to the differences with the *looped containers* [18] of Nostrum.

Throughout this paper, data integrity, lossless data delivery, and in-order data delivery are assumed to be services inherent to the router network. Freedom of reassembly deadlock and resequence deadlock is thus guaranteed [30]. In Proteo [26] that is lossy, and Nostrum [18] that uses adaptive (hot-potato) routing, additional care must be taken to recover from and avoid deadlock, respectively.

4. PROBLEM DESCRIPTION

In this paper, we assume freedom of *routing-dependent deadlock* [12], depicted in Figure 2(a). All NoCs we are aware of solve this kind of deadlock, mostly by assuring acyclic resource dependencies in the router network [3, 6, 17–24]. A dependency cycle involving *only* the routers, as shown in the figure, can hence not occur.

Although acyclic routing algorithms assert that no deadlock occurs, they do so under the *consumption assumption*. This assures that delivered messages are, in a finite time, sunk by the NIs. By induction, because the network dependencies are acyclic, all buffers are eventually emptied.

Unconditional consumption requires that delivery of one message is not coupled to the injection or reception of another message [12]. Regardless of whether the DSM or MP communication paradigm is used, IP modules often violate this assumption as a result of their *normal desirable behavior*, for example, a slave module that responds to incoming requests and thereby introduces a request-response dependency. Together with the dependencies of the network, the message dependencies can again cause dependency cycles and introduce message-dependent deadlock, as shown in Figure 2(b).

Taxing the IP modules with the responsibility of correctness (e.g., by employing end-to-end flow control on the application level) is not desired as it necessitates modification of existing IPs [31] and frustrates reuse [7]. Therefore, the onus of consumption is placed on the NIs. In the following sections, we show how the IP behavior determines the type of dependencies that arise, and in Section 6 we present solutions that guarantee consumption in their presence.

Besides the router-dependent and message-dependent deadlocks, we also address *application deadlock* [16]. This third level of deadlock, involving the IPs only, is as important as the two lower levels. It is, however, independent of the behavior of the NoC and is out of the scope of this paper.

5. MESSAGE DEPENDENCIES

We adopt the terminology used in [12]. A *message dependency chain* represents a partially ordered list of message types m_1 through m_n, where $m_i \prec m_j$ if and only if m_j can be generated by a node receiving m_i. The *chain length* denotes the number of types n in the chain. We refer to a protocol with such a message dependency chain as a n-way protocol. A message of type m_n is said to be *terminating* as it does not introduce any new messages.

5.1. Request-response dependency

A dependency that is frequently occurring in contemporary SoCs is the *request-response dependency*. As we have seen, this dependency arises in a slave module, such as a memory controller, that awaits a request and upon reception processes the request and sends a response. The protocol is clearly two-way with a message dependency chain *request* $\prec$ *response*.

The coupling between reception of request and generation of responses introduces a dependency between the request and response buffers in the NI, as depicted in Figure 2(b). Transferred to the network, this dependency can cause deadlock. In the figure, two master and slave pairs communicate via two shared input buffered routers. The two connections between m_1 and s_1 are drawn with continuous

lines and the connections of m_2 and s_2 with dashed lines. Note that dimension-ordered routing is used and that the network is clearly acyclic. Moreover, the individual master and slave pairs do not introduce cycles as there is only a message dependency on the slave side. However, a dependency cycle is formed over the two slave modules. Responses from s_1 enter the network, turn east, and end up in b_2. This buffer is shared by responses destined for m_1 and requests going to s_2. From b_2, the dependencies continue through the slave s_2, and the shared buffer b_1, back to s_1, closing the cycle. As a result, a deadlocked situation, where none of the involved connections make progress, can occur.

As we will see in Section 6.3, one way to resolve the dependencies of b_1 and b_2 is to use separate request and response networks, or at least separate buffer classes.

5.2. Response-request dependency

In contrast to what most NoC designs suggest, many protocols create more than just request-response dependencies. For example, when a master reacts on the response from a slave by sending an additional request, it creates a *response-request dependency*. Consider for example an implementation of atomic access through read-modify-write (RMW) [14, 15]. A read request is issued by the master which acquires exclusive ownership and receives a response from the slave. Finally, the master issues a write request which upon completion releases the lock. This protocol creates a message chain $read \prec response \prec write$.

Examples of more specialized protocols that have response-request dependencies are given in [27, 28, 32]. In these works, interconnections and multicast groups are established through a three-way resource reservation protocol: (1) a master sends a setup request, (2) the slave responds with a positive or negative acknowledgement, and in the latter case (3) the master restores the reservations done by sending a tear-down. The message dependency chain thus comprises three types: $setup \prec ack/nack \prec teardown$.

5.3. Request-request and response-response dependencies

The aforementioned dependencies involve only dedicated master and slave modules. This is also an assumption made by most existing solutions to message-dependent deadlock in NoCs [21–23, 27]. With the introduction of IP modules with both master and slave ports, for example, a processor or direct memory access (DMA) engine, two additional dependencies may arise: *request-request* and *response-response*.

Request-request dependencies, as depicted in Figure 3, are created when reception of a request on the slave side is coupled to the generation of a request on the master side. This occurs when IP modules process a certain input that is sent to them by the preceding module and then write their output to the succeeding module, as done in *peer-to-peer streaming* and in protocols that, in the interest of performance, use *forwarding* [12, 33].

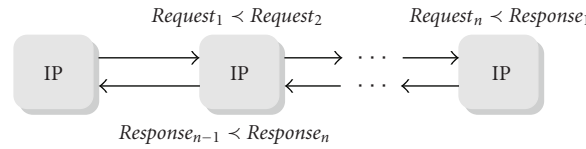


Figure 3: Message forwarding.

5.3.1. Forwarding

In a forwarding protocol, an initial request passes through a number of intermediate IPs, generating new requests until the final destination is reached. Potentially, a response is travelling in the other direction, creating response-response dependencies on the way back. Two prominent examples of forwarding protocols are *cache coherency protocols* [33] and *collective communication* [34], such as multicast and narrowcast [3, 28].

Cache coherency in network-based SoCs is typically implemented using a directory-based protocol as the medium does not lend itself to snooping [13, 14]. These protocols, in general, do not adhere to strict request-response protocols, as they strive to reduce the number of network transactions generated per memory operation [33]. Both *reply forwarding* and *intervention forwarding* manifest request-request dependencies, and the latter introduces also response-response dependencies.

Multicast and narrowcast are used in NoCs to implement DSM on a single interconnection [3], and in parallel systems also for cache invalidation, acknowledgement collection, and synchronization [34]. These higher-order interconnections give rise to both request-request and response-response dependencies when implemented using forwarding [28]. The latter is used to avoid sending a unicast message for every destination, which causes congestion at the source [35].

5.3.2. Streaming

A streaming protocol, where data is pushed from producer to consumer, is beneficial in *dataflow applications* [16, 36] comprising a chain of modules, such as the video pixel processing pipeline [37] depicted in Figure 4.

The advantage of pushing (writing) data instead of pulling it from the producer is that it greatly reduces the impact of network latency. When pulling, as suggested by [38], then first a read request is sent whereafter the producer responds with the data, thereby doubling the latency by traversing the network twice. Note that the latter approach, where every IP reads and writes its input and output, respectively, reduces the protocol to strict request-response but has several drawbacks, further discussed in Section 6.3. An example of a SoC employing peer-to-peer streaming is presented in [39] where a commercially available SoC for picture improvement is extended with a NoC.

The peer-to-peer streaming protocol where IPs write their output to the next module, illustrated in Figure 5, has a message chain that is built only from (forwarded) requests:

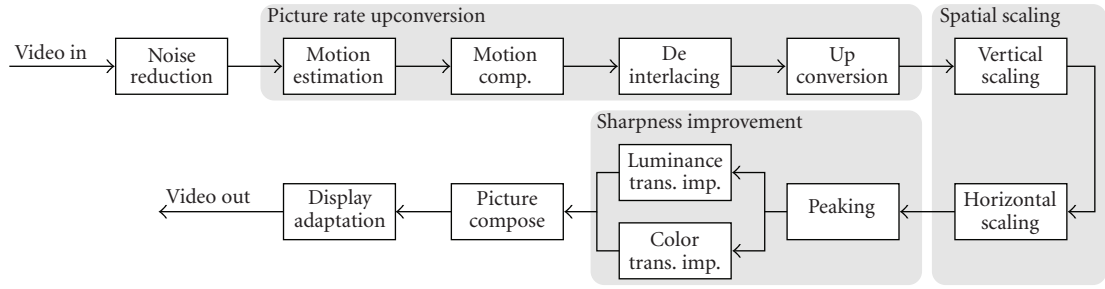


Figure 4: Video pixel processing application.

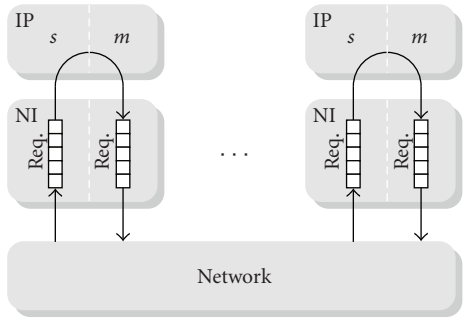


Figure 5: Dependencies created by peer-to-peer streaming.

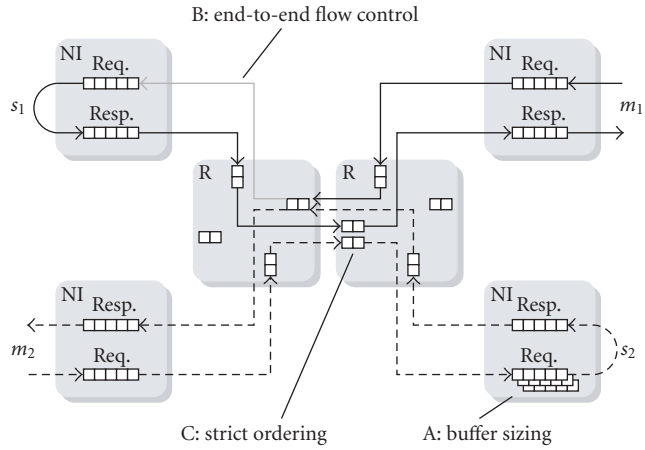


Figure 6: Various solutions.

$request_1 \prec \cdots \prec request_n$, where n is the number of modules in the processing chain. Consider for example the pipeline in Figure 4 that has 12 different types of request messages if all communications are implemented by peer-to-peer streaming.

6. SOLUTIONS

To provide a deadlock-free NoC, the consumption assumption must be fulfilled. As a first requirement, messages must be separated into different NI buffers based on their type. Having a separate NI buffer per message type is a *necessary but not a sufficient condition* to avoid deadlocks [12]. Message-dependencies together with dependencies in the router network can still introduce cycles.

As already outlined, the avoidance-based solutions to this problem fall within two categories. First, the consumption assumption can be implemented by designing the NIs such that NI buffers are guaranteed to consume all messages sent to them, regardless of the IPs. Buffer sizing (Section 6.1) and end-to-end flow control (Section 6.2) are instances of this technique. Alternatively, the NoC must guarantee that messages of one type never preclude the advances of its subordinate types indefinitely. Thereby, messages of the terminating type (guaranteed to sink upon arrival) reach their destination and its dominant types can follow suit. This technique is referred to as strict ordering (Section 6.3), and virtual circuits (Section 6.4) is a special case.

6.1. Buffer sizing

A first way to solve the deadlock problem is to ensure enough space by (over-)sizing the buffers. This requires a generous storage budget, determined by the maximum bounds on packet size and the number of outstanding transactions. The concept is shown in Figure 6 that revisits the case of a request-response protocol.

While extensively used in parallel computers [12], this method is prohibitively expensive in NoCs and is not used in any known architecture.

6.2. End-to-end flow control

Instead of adapting the buffer size to the maximum requirements, end-to-end flow control does the other way around: it assures that no more is ever injected than what can be consumed. This approach, end-to-end flow control, is used in the Æthereal and FAUST NoC. As illustrated in Figure 6, it removes a dependency edge from the network to the NI.

The simplest form of end-to-end flow control is the so-called *local flow control* [30], assuring that enough space is available to fit the response before sending the request. This local check solves response-response and response-request dependencies.

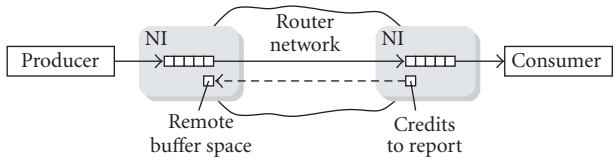


Figure 7: Credit-based end-to-end flow control per connection.

Requests-request and request-response dependencies are caused by transactions initiated by remote parties, and thus require end-to-end flow control. As buffer space is the critical resource, a windowing mechanism must be used. An example of such a mechanism is credit-based flow control, as illustrated in Figure 7. A rate-based mechanism, such as the one used in [40], is insufficient as it does not bound buffer usage.

Just as buffer sizing, end-to-end flow control solves *all* potential message dependencies. It does so without placing any restrictions on the amount of sharing in the router network. Furthermore, routers need not know message types or the number of connections, and can thereby be simplified in complexity and optimized for other important or otherwise critical features [12]. However, credit-based end-to-end flow control carries three major downsides.

First, it requires extra buffering to hide the round-trip latency of the credits. The amount of buffering is determined by the performance requirements [41] and it is evaluated in Section 7.1.

Second, communication of credits consumes bandwidth and hence power. The closed-loop nature requires state to be communicated between the producer and consumer NI. The additional bandwidth, quantified in Section 7.1, can be reduced with 20% by piggybacking credits on the data packets [3].

Third, it requires dedicated NI buffers per connection. Alternatively, if many sources share a common destination buffer they need collective knowledge of the destination and each other, something that cannot be implemented in a cost-efficient way.

6.3. Strict ordering

Another way of assuring freedom of message deadlock is by ordering network resources. This is done by introducing logically independent networks, physical or virtual, for each message type. Arteris [21], STbus [22], and SonicsMX [23] fit in the first category by having two physical networks for requests and responses, respectively. The methods used to break request-response dependencies in [27, 28] fit in the latter category as they both use one buffer class per message type. This approach is illustrated in Figure 6 where a buffer is added to break the dependency cycle.

A major drawback of the strict ordering is that buffers cannot be shared between the different message classes, increasing the amount of buffering required. The partitioning into logical networks leads to inefficient utilization of network resources [33] and increased congestion due to unbalance [12]. These effects increase with the number of networks required. In [22], the authors argue that the size of the request and response networks can be made different. The size is however static, and use-cases (modes) with different traffic characteristics magnify the problem.

Having virtual instead of physical networks mitigates the aforementioned problem. However, the router complexity increases as it must forward messages considering message type [12].

The major limitation with strict ordering is the inherent coupling between the NoC and the IP modules. A NoC with n logical networks can only be used by IP modules employing protocols with n or fewer message types. In multiprocessor designs, like the Alpha 21364 [42], this entanglement of concerns is not an issue. The router network is tailored to the protocol with seven virtual networks, one for each message type. For a NoC design, however, the coupling between IPs and the NoC architecture severely limits the reusability. Consider for example the implementation of a forwarding protocol [28] where the number of buffers determines the maximum number of multicast groups.

Higher-order protocols require either a redesign of the NoC or a reduction of the protocol to n ways. IP modules using peer-to-peer streaming communication hence cannot use the NoCs in [21–23] as they only support two-way protocols. The protocol has to be reduced to pure request-response and communication must go via memory. This adds complexity, requires additional bandwidth, introduces latency, increases congestion, and consumes more power.

6.4. Virtual circuits

Virtual circuits represent the extreme case of strict ordering as every connection has its own logical network. This way of implementing unconditional delivery is found in the guaranteed service networks of Æthereal [3], MANGO [43], and Nostrum [18]. The implementations differ, but all rely on predetermined spatial and/or temporal multiplexing.

The deadlock freedom comes at a price of exclusively reserved resources coupled with decreased utilization. Furthermore, in all three NoCs the maximum number of circuits supported by a router and NI is decided at design time. For all three NoCs, the number of buffers in the NI sets an upper bound for the number of circuits. The router is limited by the number of virtual channels (buffers) in MANGO, by the slot table size in Æthereal and by the number of temporally disjoint networks in Nostrum.

The lack of resource sharing and potentially low utilization is the major drawback of all three implementations. However, Nostrum is more severely limited than the other two, due to the *looped containers* [18] that do a round trip and reserve an equal amount of resources in both directions. As the progress guarantee requires that no circuit carries more than one message type, requests and responses must use separate circuits. A circuit may therefore only carry messages from master to slave or slave to master, not both. Thereby, Nostrum is, if no additional measures are taken, limited to maximally 50% utilization as only the forward

Table 1: Avoidance techniques used in NoCs.

NoC	Technique	2-way	n-way
aSOC	—	–	–
MANGO BE	—	–	–
Nostrum BE	—	–	–
×pipes	—	–	–
Arteris	Strict ordering	+	–
SonicsMX	Strict ordering	+	–
STbus	Strict ordering	+	–
MANGO GS	Virtual circuits	+	+
Nostrum GS	Virtual circuits	+	+
Æthereal GS	Virtual circuits	+	+
SPIN	End-to-end flow control	–	–
FAUST	End-to-end flow control	+	+
Æthereal BE	End-to-end flow control	+	+

path may carry messages. An alternative is to enforce a maximum number of outstanding transactions and a maximum transaction size and then size the buffers accordingly, as discussed in Section 6.1.

7. EVALUATION

As seen in Table 1, the best-effort network in MANGO and Nostrum, together with aSOC [17] and ×pipes [19], do not address message dependencies at all, leaving these networks susceptible to deadlock (livelock in the case of Nostrum). Hence, not even a two-way protocol can be safely implemented on these architectures without further measures.

Arteris, SonicsMX, and STbus all have separate request and response networks, which allows them to handle two-way protocols without deadlock. However, peer-to-peer streaming protocols or forwarding multicast cannot be used by the IP modules unless the NoCs are extended with additional logical networks. The pipeline in Figure 4, for example, requires ten more networks. Even then, the maximal pipeline length is still limited by the architecture. Furthermore, if one IP fails to consume its messages it can bring the entire network to a stall.

The guaranteed-service network in Æthereal, MANGO, and Nostrum all avoid message-dependent deadlocks, but do so at the price of (1) reduced resource sharing, and (2) a fixed number of connections supported by the router and NI architecture.

SPIN [20], FAUST, [24] and the best-effort network in Æthereal all employ credit-based end-to-end flow control. However, only the latter two fulfil the consumption assumption as SPIN issues more credits than the capacity of the receiving buffer. The additional credits are introduced to reduce latency, and the only consequence is said to be an increased possibility of contention in the network. However, consumption can no longer be guaranteed making the system susceptible to message-dependent deadlock. In FAUST and Æthereal, the consumption assumption is fulfilled and no message-dependency chain can introduce deadlock. The router architecture is oblivious to message types and the number of connections, but the latter is instead limited by the number of buffers in the NIs.

Table 2: Buffer cost (words).

	MPEG	$s_1m_1p_2$	$s_1m_2p_2$	$s_8m_1p_2$	$s_8m_2p_2$
Total	242	339	615	450	801
Per conn.	5.8	3.2	3.0	3.5	3.3

7.1. Cost analysis

In this section, we evaluate the cost associated with the two most resource-efficient solutions, namely strict ordering and end-to-end flow control. This is done for five different use-cases. The *MPEG* use-case is an MPEG codec SoC with 16 IP modules, tied together by 42 connections. The remaining four use-cases are internal video processing designs, all having a hot spot around a limited set of IPs (external memories) and 100 to 250 connections. These connections deliver a total bandwidth of 1-2 Gbyte/s to 75 ports distributed across 25 IP modules.

For each use-case, a NoC is dimensioned using the UMARS algorithm [44]. Given the performance requirements, NI buffer sizes are then calculated in two individual parts: (1) the amount required to decouple the IP and NI consumption and production without introducing stalls, and (2) the number of words that must be added to hide the round-trip latency of flow control [45]. The contribution of the latter is presented in Table 2.

As seen in Table 2, the average cost is merely three to six words per connection. The addition to the total NoC area is shown in Figure 8. The silicon area requirements are based on the model presented in [46], for a 0.13 μm CMOS process with full-custom FIFOs. The added NoC area is below 4% for all the applications. The mean value is 3.2%. Thus, in a network-based SoC, such as the one presented in [39], the area cost of end-to-end flow control is no more than 0.2% of the whole SoC.

To put the area cost of end-to-end flow control in contrast with strict ordering, we calculate an approximate cost of such an implementation. This is done by introducing an additional best-effort router network, identical to the one in place, thus having one network for requests and one for responses. Although we have an approximation, the results in Figure 8 suggest that the two methods are comparable in cost. The MPEG and $s_8m_2p_2$ designs have a more evenly distributed communication and less NIs per router than the other designs. As a result, close to 20% of the area is attributable to the routers in these two cases, which affects the cost of strict ordering negatively. The average area cost for strict ordering is slightly less than 3.9% of the NoC, only negligibly different from what is achieved with end-to-end flow control.

Note that we add only one router network for the comparison as all the use-cases employ a strict request-response protocol. With the introduction of higher-order

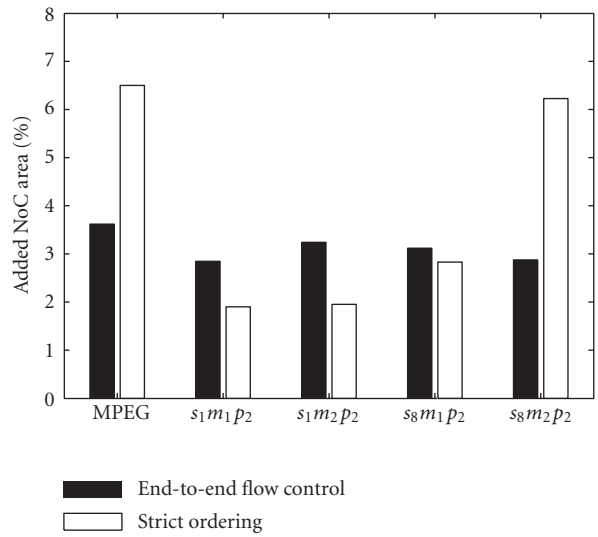


Figure 8: Comparison of added network area.

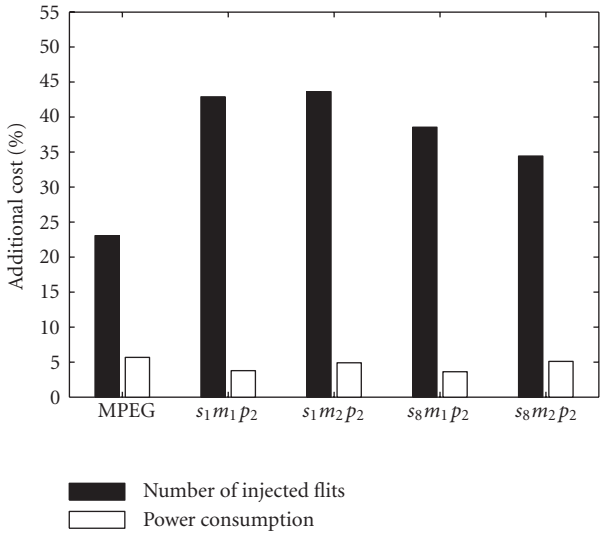


Figure 9: Additional traffic and power consumption.

protocols, the cost of end-to-end flow control remains constant, whereas the cost of strict ordering *increases linearly* with the number of logical networks (protocol stages). This is under the assumption that all network components are designed to handle all different message types. As proposed in [29], it is, for a given application, possible to reduce the cost by only introducing the additional buffer classes where strictly needed.

To asses the cost of the traffic introduced by the end-to-end flow control, we simulate each design 3×10^6 clock cycles in a flit-accurate SystemC simulator of the Æthereal NoC, using traffic generators to mimic core behavior. Figure 9 shows the additional cost in terms of injected flits and power consumption.

The additional amount of injected flits ranges from 23% up to 44%. The MPEG design has an average bandwidth (76 Mbyte/s) three times higher than the other designs, which results in less flits carrying only credits. A higher bandwidth (and larger burst size) increases the opportunities for piggybacking credits on data-carrying packets [3]. Furthermore, it also leads to a more bursty delivery of credits with more credits per packet. As a result, buffers grow (see Table 2), but less credit-carrying flits are injected.

As more flits are injected and routed through the network, also the power consumption increases. The contribution added by the credit-carrying flits is depicted in Figure 9. Note that the power estimation, calculated according to the model in [47], covers only the router network (without the NIs). In the reference case with no flow control, the flits that carry only credits and no data are treated as empty. Despite the amount of flits, the additional cost in power consumption is consistently below 6%, with an average of 4.6%.

8. CONCLUSION AND FUTURE WORK

In this paper we analyze *message-dependent deadlock*, arising due to protocol interactions between the NoC and the IP modules. We compare the possible solutions and show that deadlock avoidance, in the presence of higher-level protocols, for example, cache coherency, slave locking and peer-to-peer streaming, poses a serious challenge for many current NoC architectures.

Furthermore, we show how a NoC, such as the Æthereal and FAUST NoCs, employing credit-based end-to-end flow control, provides robust communication services for *all* potential communication protocols used. We show that the associated area and power cost represent 4% and 6%, respectively, of a typical Æthereal NoC instance.

Future work includes a more in-depth analysis of the costs associated with the various solutions in the presence of streaming peer-to-peer protocols.

REFERENCES

[1] L. Benini and G. de Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[3] A. Rădulescu, J. Dielissen, S. Gonzàlez Pestana, et al., "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, 2005.

[4] M. Sgroi, M. Sheets, A. Mihal, et al., "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 667–672, Las Vegas, Nev, USA, June 2001.

[5] M. Coppola, S. Curaba, M. D. Grammatikakis, G. Maruccia, and F. Papariello, "OCCN: a network-on-chip modeling and simulation framework," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 3, pp. 174–179, Paris, France, February 2004.

[6] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø, "An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip," in *Proceedings of International Symposium on System-on-Chip (SOC '05)*, pp. 171–174, Tampere, Finland, November 2005.

[7] D. Wingard, "Socket-based design using decoupled interconnects," in *Interconnect-Centric Design for SoC and NoC*, J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Eds., Kluwer, Dordrecht, The Netherlands, 2004.

[8] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in NoC design: a holistic perspective," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '05)*, pp. 69–74, Jersey City, NJ, USA, September 2005.

[9] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, 2000.

[10] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 688–693, Munich, Germany, March 2003.

[11] J. Hu and R. Marculescu, "DyAD—smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 260–263, San Diego, Calif, USA, June 2004.

[12] Y. H. Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 3, pp. 259–275, 2003.

[13] T. T. Ye, L. Benini, and G. de Micheli, "Packetized on-chip interconnect communication analysis for MPSoC," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 344–349, Munich, Germany, March 2003.

[14] F. Pétrot, A. Greiner, and P. Gomez, "On cache coherency and memory consistency issues in NoC based shared memory multiprocessor SoC architectures," in *Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD '06)*, pp. 53–60, Dubrovnik, Croatia, August-September 2006.

[15] *AMBA AXI Protocol Specification*, ARM, June 2003.

[16] M. Bekooij, R. Hoes, O. Moreira, et al., "Dataflow analysis for real-time embedded multiprocessor system design," in *Dynamic and Robust Streaming in and between Connected Consumer-Electronics Devices*, P. van der Stok, Ed., Kluwer, Dordrecht, The Netherlands, 2005.

[17] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *Proceedings of International Symposium on Asynchronous Circuits and Systems (ASYNC '05)*, pp. 54–63, New York, NY, USA, March 2005.

[18] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 890–895, Paris, France, February 2004.

[19] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. de Micheli, "×pipes lite: a synthesis oriented design library for networks on chips," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1188–1193, Munich, Germany, March 2005.

[20] P. Guerrier, "Un réseau d'interconnexion pour systémes intégrés," Ph.D. dissertation, Universite Paris VI, Paris, France, 2000.

[21] Arteris, "A comparison of network-on-chip and busses," White paper, 2005.

[22] S. Murali and G. de Micheli, "An application-specific design methodology for STbus crossbar generation," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1176–1181, Munich, Germany, March 2005.

[23] *SonicsMX Datasheet*, Sonics, 2005, http://www.sonicsinc.com/.

[24] Y. Durand, C. Bernard, and D. Lattard, "FAUST: on-chip distributed architecture for a 4g baseband modem SoC," in *Proceedings of IP Based SoC Design Conference and Exhibition (IPSOC '05)*, Grenoble, France, December 2005.

[25] A. Tanenbaum, *Computer Networks*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[26] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo networks-on-chip," in *Proceedings of International Symposium on Circuits and Systems (ISCAS '03)*, pp. 113–116, Bangkok, Thailand, May 2003.

[27] B. Gebremichael, F. Vaandrager, Z. Miaomiao, K. Goossens, E. Rijpkema, and A. Rădulescu, "Deadlock prevention in the Æthereal protocol," in *Proceedings of the 13th IFIP WG 10.5 Advanced Research Working Conference Correct Hardware Design and Verification Methods (CHARME '05)*, pp. 345–348, Saarbrücken, Germany, October 2005.

[28] Z. Lu, B. Yin, and A. Jantsch, "Connection-oriented multicasting in wormhole-switched networks on chip," in *Proceedings of IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, pp. 205–210, Karlsruhe, Germany, March 2006.

[29] S. Murali, P. Meloni, F. Angiolini, et al., "Designing message-dependent deadlock free networks on chips for application-specific systems on chips," in *Proceedings of IFIP International Conference on Very Large Scale Integration*, pp. 158–163, Nice, France, October 2006.

[30] M. Gerla and L. Kleinrock, "Flow control: a comparative survey," *IEEE Transactions on Communications Systems*, vol. 28, no. 4, pp. 553–574, 1980.

[31] P. Bhojwani and R. Mahapatra, "Interfacing cores with on-chip packet-switched networks," in *Proceedings of 16th International Conference on VLSI Design*, pp. 382–387, Las Vegas, Nev, USA, June 2003.

[32] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[33] D. E. Culler, J. P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 1999.

[34] R. Sivaram, R. Kesavan, D. K. Panda, and C. B. Stunkel, "Where to provide support for efficient multicasting in irregular networks: network interface or switch?" in *Proceedings of International Conference on Parallel Processing (ICPP '98)*, pp. 452–459, Minneapolis, Minn, USA, August 1998.

[35] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource deadlocks and performance of wormhole multicast routing algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 6, pp. 535–549, 1998.

[36] G. Kahn, *Information Processing*, North-Holland, New York, NY, USA, 1974, ch. The Semantics of a Simple Language for Parallel Processing.

[37] O. P. Gangwal, J. Janssen, S. Rathnam, E. Bellers, and M. Duranton, "Understanding video pixel processing applications for flexible implementations," in *Proceedings of Euromicro Symposium on Digital System Design*, pp. 392–401, Belek-Antalya, Turkey, September 2003.

[38] H. Nikolov, T. Stefanov, and E. Deprettere, "Multi-processor system design with ESPAM," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '06)*, pp. 211–216, Salzburg, Austria, September-October 2006.

[39] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R. Peset Llopis, "Networks on chips for high-end consumer-electronics TV system architectures," in *Proceedings of Design, Automation and Test in Europe (DATE '06)*, vol. 2, pp. 1–6, Munich, Germany, March 2006.

[40] S. Murali, L. Benini, and G. de Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, vol. 1, pp. 27–32, Shanghai, China, January 2005.

[41] O. P. Gangwal, A. Rădulescu, K. Goossens, S. Gonzàlez Pestana, and E. Rijpkema, "Building predictable systems on chip: an analysis of guaranteed communication in the Æthereal network on chip," in *Dynamic and Robust Streaming in and between Connected Consumer-Electronics Devices*, Kluwer, Norwell, Mass, USA, 2005.

[42] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture," *IEEE Micro*, vol. 22, no. 1, pp. 26–35, 2002.

[43] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of Design, Automation and Test in Europe (DATE '05)*, vol. 2, pp. 1226–1231, Munich, Germany, March 2005.

[44] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '05)*, pp. 75–80, Jersey City, NJ, USA, September 2005.

[45] M. Coenen, S. Murali, A. Rădulescu, K. Goossens, and G. de Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES, ISSS '06)*, pp. 130–135, Seoul, Korea, October 2006.

[46] S. Gonzàlez Pestana, E. Rijpkema, A. Rădulescu, K. Goossens, and O. P. Gangwal, "Cost-performance trade-offs in networks on chip: a simulation-based approach," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 764–769, Paris, France, February 2004.

[47] J. Dielissen, A. Rădulescu, and K. Goossens, "Power measurements and analysis of a network-on chip," Tech. Rep. NL-TN-2005-0282, Philips Research Laboratories, Eindhoven, The Netherlands, 2005.