Journal of Systems Architecture 133 (2022) 102766

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Comparing timed-division multiplexing and best-effort networks-on-chip

Jens Sparsø [a,*], Hans Jakob Damsgaard [a,b], Dimitrios Katsamanis [c], Martin Schoeberl [a]

[a] *Technical University of Denmark, Department of Applied Mathematics and Computer Science, 2800 Kgs. Lyngby, Denmark*
[b] *Tampere University, Electrical Engineering Unit, 33720 Tampere, Finland*
[c] *ARM Sweden A/B, Lund, Sweden*

ARTICLE INFO

Keywords:
Multi-core/single-chip multiprocessors
On-chip interconnection networks
Time-division-multiplexing
Performance analysis
Queuing theory model

ABSTRACT

Best-effort (BE) networks-on-chips (NOCs) are usually preferred over time-division multiplexed (TDM) NOCs in multi-core platforms because they are work-conserving and have lower (zero-load) latency. On the other hand, BE NOCs are significantly more expensive to implement than TDM NOCs because of their virtual channel buffers, allocators/arbiters, and (credit-based) flow control; functionality that a TDM NOC avoids altogether.

The objective of this paper is to compare the performance of BE and TDM NOCs, taking hardware cost into consideration. The networks are compared using graphs showing average latency as a function of offered load. For the BE NOCs, we use the BookSim simulator, and for the TDM NOCs, we derive a queuing theory model and an associated TDM NOC simulator.

Through experiments with both router architectures, packet length, link width, and different traffic patterns, we show that for the same hardware cost, a TDM NOC can provide higher bandwidth and comparable latency. We also show that the packet length is the most important factor affecting the TDM period, which again is the primary factor affecting latency. The best TDM NOC design for BE traffic uses single flit packets, wide links/flits, and a router with two pipeline stages: link and router traversal.

1. Introduction

Time-division multiplexing (TDM) networks-on-chips (NOCs) [1–5] are intended for use in hardware platforms for real-time systems where the ability to guarantee latency and throughput of individual processor-to-processor communication-flows is crucial. In the bigger picture, NOCs that offer such hard service guarantees [6] represent a niche. Most NOC research focuses on general-purpose computing platforms and efficient support for best-effort (BE) traffic (i.e., low actual/average-case latency and high bandwidth). In this bigger picture, TDM NOCs are often ruled out by the criticism that they are *not work-conserving:* time-slots are statically assigned to individual traffic flows, and consequently, bandwidth (i.e., time-slots) that is not used cannot be used by other traffic flows. This results in poor bandwidth utilization and increased latency. Another criticism is that *bandwidth and latency are inversely proportional:* in order to provide low latency, it is necessary to reserve many time-slots and thereby a large bandwidth—bandwidth that, for the most part, is not used.

Both criticisms are intuitively correct, but they ignore hardware cost. A TDM NOC avoids hardware resources for arbitration/allocation, virtual channel (VC) buffers, and flow control, so for the same hardware cost, a TDM NOC can provide *substantially higher* bandwidth, which may compensate for the poorer bandwidth utilization. If bandwidth is cheap and plentiful, it is less important to maximize its use. In addition, a TDM router is just a pipelined switch, and the number of pipeline stages is generally smaller than that of a standard BE virtual channel router, resulting in a smaller network traversal latency. These factors may weaken or perhaps even invalidate the above criticism of TDM NOCs.

The observations mentioned in the previous paragraph have been briefly touched upon in the literature, including [7,8]. Yet, to the best of our knowledge, a more thorough and quantitative analysis of this performance vs. hardware cost perspective on TDM NOCs, as well as the use of TDM NOCs for BE traffic, has not been attempted before.

The objective of this paper is to compare the performance of a typical TDM NOC against that of a typical textbook-style BE NOC [9–11] and to gain insight into how to optimize the TDM NOC for minimum latency. The TDM NOC we use in this study is the Argo NOC [5], but the results apply to most TDM NOCs including [1–4]. The performance measure we use is the average end-to-end latency of packets as a function of the packet injection rate.

For the BE NOCs, the results are obtained using the BookSim simulator [10]. For the TDM NOCs, we do not know of any available

* Corresponding author.
E-mail addresses: jspa@dtu.dk (J. Sparsø), hans.damsgaard@tuni.fi (H.J. Damsgaard), Dimitris.Katsamanis@arm.com (D. Katsamanis), masca@dtu.dk (M. Schoeberl).

https://doi.org/10.1016/j.sysarc.2022.102766
Received 30 January 2022; Received in revised form 12 September 2022; Accepted 17 October 2022
Available online 22 October 2022

simulator, and as the behavior of a TDM NOC is much simpler and rigid, we developed an analytical model using queuing theory. The model provides more insight than simulation-based results, and the results (average latency versus injection rate) are cycle-accurate. This analytical model is the *first contribution* of the paper. To consolidate the results, we developed a corresponding TDM NOC simulator, which uses the same traffic generators and time-stamping mechanisms as BookSim. This simulator is a *second contribution* of the paper.

Using these simulators and models, we then conduct a number of experiments where we compare and assess the performance and hardware cost of typical TDM and BE NOC configurations: mesh and bi-torus topologies of different sizes. The performance measures include the (minimum possible) zero-load latency and the average latency at different packet injection rates, and the saturation throughput. The hardware cost is quantified using a dimensionless relative area measure and FPGA and ASIC synthesis results. These analyses and the insights gained from them are described below and are a *third contribution* of the paper.

Most of the experiments are conducted using random uniform traffic, which is known to be benign towards BE NOCs [9] because it tends to spread the traffic across the network. A TDM NOC, on the other hand, must provide private/virtual end-to-end circuits for all possible pairs of nodes, meaning that random uniform traffic is very challenging, especially as the number of nodes in the network grows. This means that the results obtained are favorable for the BE NOCs. To balance this, we also explore using bit-complement and tornado traffic, which is known to be more challenging for the BE NOCs.

Our first results show that for networks up to about 64 nodes, a typical TDM network can deliver roughly the same (saturation) throughput as a typical BE NOC, for around one-tenth of the hardware cost, but with a five to ten times higher zero-load latency. Following this, the paper explores how a TDM network can trade increased hardware cost for a corresponding reduction of latency *and* a corresponding increase of bandwidth. In these configurations, the throughput of the TDM NOC exceeds that of the BE NOC, still for a smaller hardware cost.

These characteristics make the TDM networks interesting (low cost or high throughput) points in the design space. We note that the performance of an application executing on a multi-core platform depends on both the latency and the throughput of the NOC in ways that can be very complex and application dependent. Here is an area for further research.

To summarize, the main contributions of this paper are as follows:

1. An analytical queuing theory model of a TDM NOC. The model gives cycle-accurate results for the average latency of packets.
2. A simulator that combines this model with traffic generators similar to those used in BookSim.
3. A detailed comparison of the area and performance (average packet latency versus offered load) of typical TDM and BE NOCs. One of our results shows that for a smaller area, a TDM NOC can provide similar latency and higher bandwidth than a BE NOC.
4. Insights into how to best dimension a TDM NOC for minimum latency.

The paper is organized as follows: Section 2 provides background on the relation between latency and offered load, the BookSim simulator, and the TDM-based Argo NOC. Section 3 develops an analytical queuing theory model of a TDM NOC, from which the average latency versus offered load can be calculated. Section 4 provides equations for the asymptotes (saturation throughput and zero-load latency) of the latency curves. These are subsequently used to validate the results. Section 5 provides results on latency versus offered load from our first experiments. Section 6 explores ways of improving latency and throughput. Section 7 explores the performance when using other traffic patterns (bit-complement and tornado). Finally, Section 8 concludes the paper.

2. Background and related work

This section presents background and related work on NOCs, micro-architectures of BE and TDM routers, performance expressed as latency versus offered load, and simulation tools to explore the latter.

2.1. Networks on chip

A network-on-chip (NOC) [12,13] is a packet-switched communication structure used to connect processors, accelerators, memories, and IO devices in chip multiprocessors and embedded systems-on-chips. During the last 10–20 years, NOCs have largely superseded bus-based interconnects due to the increased complexity and number of cores in today's chips. The reason is that a NOC offers solutions to both technological challenges (bandwidth and buffering of long wires, the perspective of [13]) and design methodology challenges (composability and scalability, the perspective of [12]). NOCs share many fundamental characteristics with inter-chip interconnection networks [9], but the design tradeoffs of an intra-chip network are different, and this has been extensively researched during the last 20 years.

Bus-based systems are *circuit-switched*, meaning that a physical circuit exists between an initiator node (e.g., a processor) and a target node (e.g., a memory or IO device) for the duration of a read/write transaction. NOCs are *packet-switched*, and typically packets propagate through the network when the resources needed are available—they are said to be *work-conserving* and called *best effort* (BE) networks. The latency of traversing the NOC is typically low, but when the amount of traffic increases, it may not be possible to give bounds on the latency packets may experience.

An alternative is TDM NOCs, where packets are transmitted according to a global, static, repeating schedule. In this way, traffic from one node to another always experiences the same latency regardless of other traffic in the network, effectively creating *virtual circuits* between communicating pairs of nodes. For this reason, TDM NOCs are particularly relevant in real-time systems where latency must be guaranteed. The static scheduling means that TDM NOCs avoid run-time arbitration and the associated need to buffer packets. When a packet is injected into the network, it propagates unimpeded through the NOC in a pipelined fashion. Hence, the hardware cost is small (just pipeline registers and multiplexers). The downside is that TDM NOCs are not work-conserving; even when the required resources in the NOC are available, packets still wait until the scheduled departure time. The result is higher latency.

Example BE and TDM NoCs are presented in Sections 2.3 and 2.6.

2.2. Latency vs. Offered load

The performance of an interconnection network is often described using curves showing the (average) latency that packets experience when traversing the network as a function of the offered load [9]. Fig. 1 shows a generic example.

Briefly, the curve has two asymptotes [9]: (1) a horizontal one called the zero-load latency, denoted T_0, corresponding to the minimum latency experienced for very low traffic loads, and (2) a vertical one at λ_S corresponding to an amount of injected traffic where the network saturates and the latency grows without limit. The minimum latency, T_0, depends on the topology (mesh, bi-torus, etc.) and the number of pipeline stages in routers and links along the path from source to destination. The saturation point, λ_S, depends on many partially interdependent factors, including topology (mesh, bi-torus, etc.), routing function (X first then Y, dynamic, etc.), depth of the buffers in the routers, number of VC buffers in router ports, and traffic pattern (random uniform, bit reverse, tornado, etc.).

In this paper, we aim to derive similar curves for TDM NOCs through a combination of thorough development and analysis of a queuing theory model and simulation. In a TDM NOC, both T_0 and

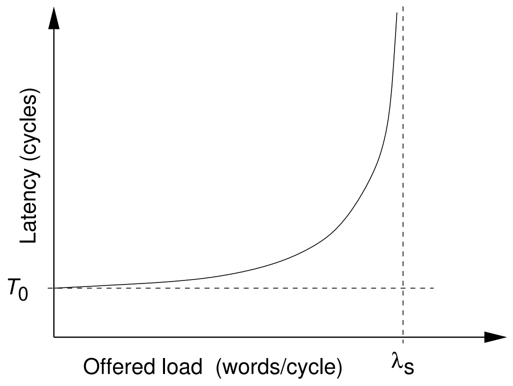


Fig. 1. Example latency versus offered load curve.

λ_S also depend on the period of the static TDM schedule, as explained later.

A representative related work targeting BE NOCs is [14]. It uses the same synthetic traffic patterns as we do but does not consider hardware cost.

2.3. The Reference Best-Effort NOC

The BE NOCs that we compare against use what can be considered a standard textbook router [9, Ch. 16.1] [11, Ch.6], i.e., an input-buffered credit-based virtual channel router. A typical implementation has four pipeline stages [11, Fig. 6.15(b)]: (i) buffer write, (ii) route computation and VC allocation, (iii) switch allocation, and (iv) switch traversal. Traversing a link connecting two routers may take one or more additional clock cycles. Multiple virtual channels per physical port are used to avoid or minimize head-of-line blocking and deadlocks.

In the simulations reported later, we vary the numbers of virtual channels (2, 4, 8, and 16 virtual channels per router port) and the packet length (1, 3, and 17 flits), and we explore mesh and bi-torus topologies of different sizes (8×8 and 15×15 nodes). The hardware cost of a number of these configurations is reported and discussed in Section 5.4.

Finally, we mention that a parameterized open-source implementation of a NOC as described here is presented in [15] and available at [16].

2.4. The BookSim simulator

BookSim is an open-source, highly flexible, and cycle-accurate NOC simulator [10]. BookSim models the router pipeline in detail and allows specification of all details from network topology, number of nodes, router micro-architecture, internal delays, and flow control algorithm to traffic generation. All parameters are passed to BookSim either through a configuration file or using command-line arguments, making it easy to perform a simulation sweep of the same NOC with different injection rates.

In BookSim, a traffic manager generates constant-size packets according to a specified traffic pattern (i.e., determination of a packet's sink node), and injection process (i.e., when to inject packages). The generated packets are injected into the network from their source nodes and are then tracked at flit level throughout the network until they reach their sinks. The simulated network consists of routers and links. The routers are configurable in terms of their port width, the number of pipeline stages from input to output, the number and size of their VC buffers, their routing function, and their allocation and arbitration algorithms. All links are bidirectional.

2.5. Low latency BE routers

On-chip interconnection networks have been researched intensively during the last 20 years. Many works have addressed router designs that improve the (zero-load) latency below the 4–5 cycles of the textbook router described above. Representative examples are [17–21].

BookSim does not support any of these optimized routers, and due to their diversity and complexity, attempts to model them quickly become a topic in itself, diverting focus. We aim to compare the cost and performance of typical and representative TDM and BE networks. Towards this goal, we note that it is possible to calculate both λ_S and T_0 for the optimized BE routers without detailed knowledge of their micro-architecture. The zero-load latency, T_0, depends solely on the (best case) number of clock cycles per hop, i.e., the number of pipeline stages in a router and a link. And the saturation throughput, λ_S, depends primarily on the bisection bandwidth of the particular NOC topology (as discussed in Section 5.3), and it is thus largely *independent* of the detailed router implementation. Exceptions are bi-torus topologies with few virtual channels.

Based on these observations, we conclude that we can estimate graphs showing latency versus offered load for the BE routers implementing different forms of bypassing and lookahead routing by simply scaling the y-axis of graphs obtained using BookSim to fit the calculated zero-load latency.

2.6. TDM NOCs

Time-division multiplexing (TDM) has been used in computer networks for decades, and the term TDM refers to the fact that resources, e.g., communication links, are used in a time-multiplexed manner. This multiplexing can be static according to a fixed schedule or dynamic to improve bandwidth use. The latter is known as statistical TDM (STDM).

Plain TDM NOCs, e.g., [2,3,5,22], are known for their time-predictability and small hardware implementation (avoiding any dynamic arbitration, buffering, and flow control). STDM is mainly used in local and wide area computer networks [23,24], and STDM-based NOCs are rare [25]. Their dynamic operation means they compromise time-predictability and are more complex to implement. In this paper, we focus on networks using static scheduling, which can be seen as TDM in its simplest and cleanest form.

All the TDM NOCs cited above offer similar functionality: transfer of packets/messages across virtual end-to-end circuits connecting pairs of processors. To help illustrate the operation of a TDM NOC, Fig. 2(a) shows a 3×3 node platform using a mesh NOC. A key feature of a TDM NOC is that packets, once injected into the network, travel without ever competing for resources. This avoids all buffering, dynamic arbitration, and flow control. Consequently, a router is merely a crossbar, possibly pipelined, which can be implemented using just a multiplexer in each output port [4]. Like in any other NOC, links may be pipelined as well.

Let us assume that routers and links in Fig. 2(a) each have a latency of one clock cycle, and that packets consist of three flits; a header and two payload flits. A static and periodic schedule controls the transmission of packets out of a node. The schedule specifies (i) the departure time for packets destined for other nodes and (ii) the route that the packets must travel. The situation is analogous to the departure schedules posted in train stations. To travel to a given city, one has to wait for the next scheduled train to that city following a pre-planned route.

The communication requirements of an application are typically specified using a so-called *core communication graph*, or short *core graph*, where nodes represent processors and directed edges, annotated with bandwidth requirements, represent the virtual circuits. A more formal description is given in Definition 1 in [26,27].

We use a metaheuristic algorithm to produce the global schedule comprising the local departure schedules for all the nodes. Given a core

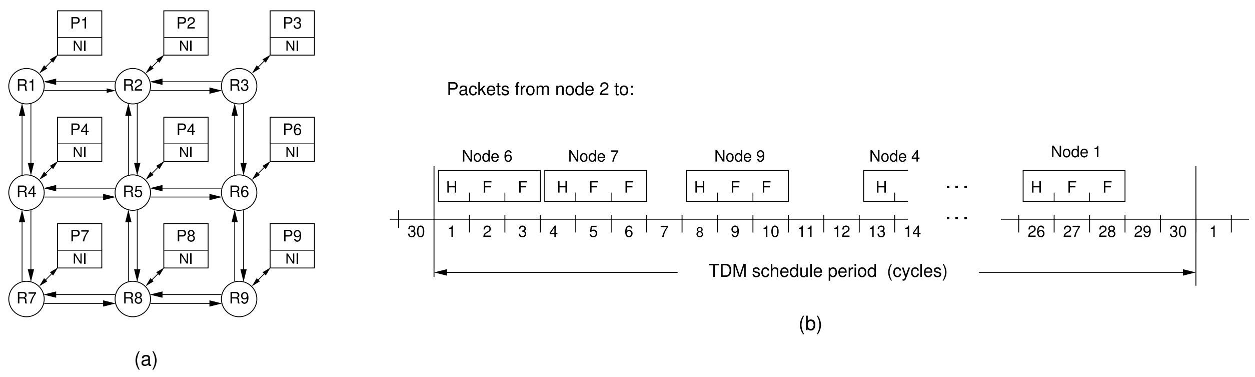


Fig. 2. (a) Example 9 core multiprocessor platform using a 3 × 3 mesh NOC. (b) TDM schedule for transmission of packets from node 2.

communication graph, the algorithm aims to find the shortest possible periodic schedule that can support all the required virtual circuits.

One of the traffic patterns we use later in our analysis is random uniform traffic. To support this traffic pattern, every processor must have a virtual circuit towards each of the other processors. In a 3 × 3 node platform, as shown as an example in Fig. 2(a), the NOC must provide a total of $8 \cdot 9 = 72$ virtual (end-to-end) circuits.

In the 3×3 mesh platform using 3-flit packets, all the 72 virtual circuits necessary to support random uniform traffic can be implemented using a schedule with a period of 30 clock cycles. Fig. 2(b) shows the schedule for node 2.

As seen, the three flits comprising a packet destined for node 7 must depart from node 2 in clock cycles 4, 5, and 6, collectively called *a slot*. In these three cycles, the packet traverses router R2. There are 3 possible shortest paths from node 2 to node 7: R2-R1-R4-R7, R2-R5-R4-R7, and R2-R5-R8-R7. Assuming the scheduler has selected the first of these, the packet traverses the link from R2 to R1 in cycles 5, 6, and 7; router R1 in cycles 6, 7, and 8; the link from R1 to R4 in cycles 7, 8, and 9, etc.

Schedule periods are typically longer than what a node needs to inject packets. In our example, it takes 24 cycles to inject the packets sent by a node, but it takes the NOC 30 cycles to transmit all the packets. This is the primary reason for the gaps in the schedule (cycles 7, 11, 12, …, 29 and 30 in Fig. 2(b)). In addition, the scheduler is typically unable to find schedules that use all network resources in every clock cycle. This adds additional gaps. Finally, it must be noted that if a node does not have data to send when the schedule allows this, then the reserved slot in the schedule is not used, and the reserved resources are consequently left idle. Alternatively, some implementations transmit void flits in these unused cycles.

For all networks (BE and TDM), it is important to correctly consider the queuing of packets at the source nodes to obtain accurate latency results. As mentioned in Section 2.4, BookSim assumes infinitely deep queues in the source nodes. For the TDM networks, it is even more important to consider the implementation and modeling of these queues because packets wait in the source nodes until their scheduled departure time, as described above.

Suppose a source node has only a single queue shared by all outgoing virtual circuits. In that case, a packet at the head of the queue scheduled for a later point in time may block a packet scheduled for an earlier point in time and for a different destination. For a TDM network, such head-of-line blocking would be extremely harmful for both latency and bandwidth utilization. To avoid this, every source node must offer separate queues for each outgoing virtual circuit, as was silently assumed in the presentation above.

2.7. The Argo TDM NOC

The Argo NOC presented in this section implements the queues and the TDM scheduling described in the previous section using a very efficient and small hardware implementation. The key to this lies in the TDM scheduling, which means the queues are accessed one at a time.

In Argo, every outgoing virtual circuit from a processor core has its own direct memory access (DMA) controller that can be set up to read a block of data (a message) from a local private scratchpad memory (SPM) and push this data, in the form of a sequence of packets, across the NOC and into the local private SPM of the receiver as illustrated in Fig. 3. The figure omits details on instruction and data memories; it only shows details of the network interface and how it is connected to the processor using a dual-ported SPM.

The Argo NOC uses source routing, and each network interface stores its local subset of the global schedule. The network interface consists of a counter (*TDM count* in Fig. 3) that counts modulo the period of the TDM schedule, a schedule table, and a bank of DMA controllers. The latter is organized as a table where each entry contains an active-bit, a read pointer, a write pointer, and a word counter. The schedule table is indexed by the TDM counter. An entry holds a valid flag (indicating if the slot is used or not), the route to be taken by a packet transmitted in the slot, and an index into the DMA table (identifying which DMA controller is to source the payload data and the destination address for the packet).

The packets in a TDM period are generally short; in the original version of Argo, one header word and two payload words. Larger messages are transferred using multiple packets as the TDM schedule repeats.

The schedules used by the Argo NOC are generated using a meta-heuristic scheduling algorithm [28]. Input to this scheduler is a core communication graph specifying the desired set of virtual circuits, their relative bandwidth, and the size/length of the packets. Depending on their bandwidth requirements, virtual circuits are assigned one or more slots for transmitting packets within the periodically repeating schedule. The meta-heuristic scheduling algorithm aims to minimize the period of the schedule, P_{TDM}. As will be clear from the following sections, P_{TDM} greatly affects the performance of the TDM networks. Intuitively, a core communication graph with few virtual circuits, equal bandwidth requirements, and a NOC using short packets results in schedules with short periods. In contrast, core communication graphs with many virtual circuits and long packets result in very long schedule periods.

3. Analytical model of a TDM network

In this section, we derive a formal queuing theory model that can be used to calculate the average latency of a TDM NOC exposed to random uniform traffic. The model is derived for the Argo NOC [5], but the same model applies to other TDM NOCs that offer separate FIFOs for each virtual end-to-end circuit, e.g., Æthereal [2] and aelite [3].

To support random uniform traffic in a TDM NOC, all processors must have virtual end-to-end circuits towards all other processors,

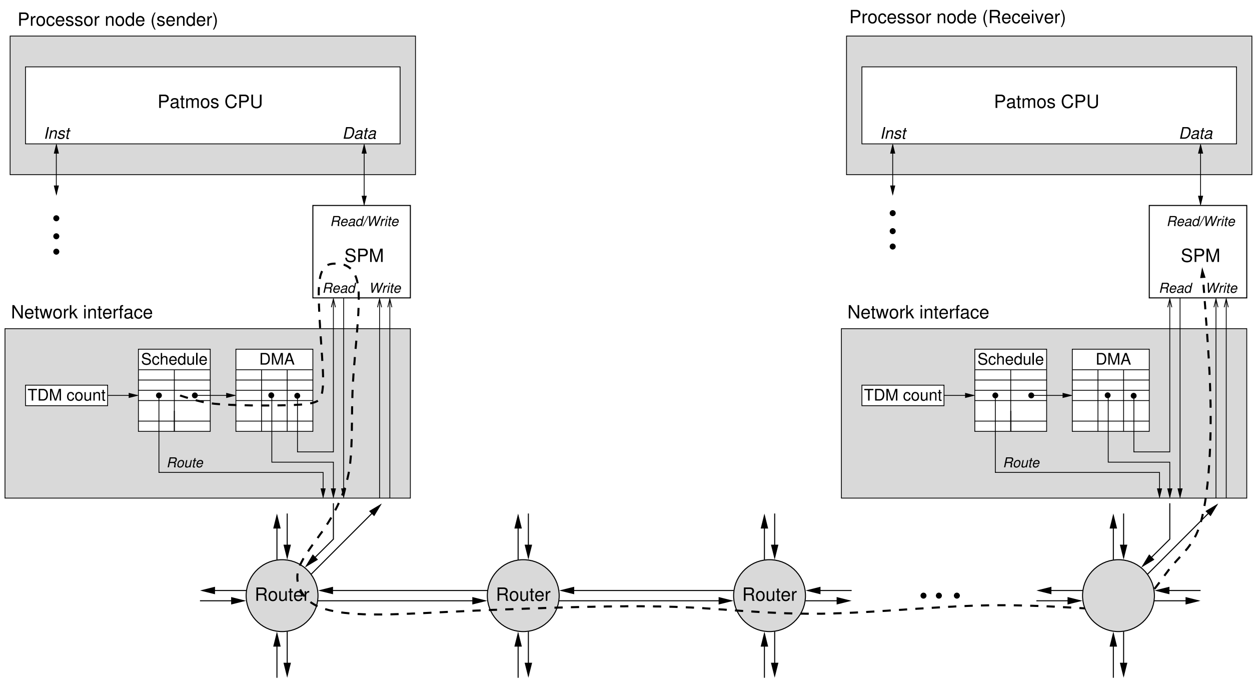


Fig. 3. Controlled by the TDM schedule, a DMA controller, one for each outgoing virtual circuit, reads data from the SPM and injects it into the network of routers and links. Incoming packets write the payload data directly into the SPM of the receiver.

and all these virtual circuits have identical bandwidth, as discussed in Section 2.6. The traffic out of one processor is generated by a random memoryless source: packets are generated at random points in time with an average rate, and the destination processor is randomly selected. Data is transferred from the SPM in a source processor and into the SPM in a sink processor.

The SPM at the source end can be represented by a first-in, first-out queue (FIFO) for each end-to-end circuit holding data to be transmitted. When packets arrive at their destination, they are immediately written into the SPM, meaning that the sink can be ignored in an analysis of the total end-to-end latency. Considering a specific source–destination node pair (i, j), or an average across all virtual circuits, the network of routers causes a constant latency: the time it takes to traverse the NOC once the packet has been injected into the NOC.

Ignoring the setup of the DMA controllers needed to initiate data transfers and remembering that packets arriving at their destination essentially write themselves into the SPM without any involvement from the network interface, we get the model shown in Fig. 4. Packets are generated in processor node P_i at a rate λ_{Node} and are destined randomly to one of the $N-1$ processors. The TDM NOC transfers data from the head of the FIFOs in the source processor to the destination processors in a fully deterministic manner according to the TDM schedule. Such NOC-traversal represents a constant latency, simply the accumulated pipeline depth in all the routers traversed from source to destination. At the sink, incoming packets immediately write their payload into the SPM, and this contributes no additional latency.

The source representing one processor in Fig. 4(a) can be replaced by a source for each FIFO injecting packets at a rate of $\lambda = \frac{\lambda_{\text{Node}}}{N-1}$. In combination with the constant average latency of the NOC, this allows us to model a virtual end-to-end circuit connecting a pair of processors, as shown in Fig. 4(b), using a single source, a single queue, a single server, and a delay element representing the latency of traversing the NOC. The server operates in a periodic manner with a service rate of $\mu = 1/P_{\text{TDM}}$; the period of the TDM schedule. When the injection rate λ reaches this rate, the FIFOs will overflow, and the NOC will fail to operate correctly.

The source, the queue, and the server in Fig. 4(b) resemble the M/D/1 queue model [29–31], where the average waiting time in the queue is:

$$T_{queue(M/D/1)} = \frac{\lambda/\mu}{2\mu(1-\lambda/\mu)} \tag{1}$$

However, there is one important difference. In the M/D/1 model, an element arriving at an empty queue when the server is idle is serviced immediately, whereas in our TDM NOC, an element arriving at an empty queue is not serviced (i.e., injected into the network) until the next scheduled time. The situation resembles a bus stop where buses with a capacity of one passenger depart according to a periodic schedule. The service rate is the inverse of the interval between departures. In a situation where a passenger arrives at an empty bus stop at a time that is random and uniform in relation to the TDM schedule, the average waiting time is half the period between departures. Adding this to the expression in Eq. (1) we get Eq. (2).

$$T_{queue(TDM)} = \frac{1}{2\lambda} + \frac{\lambda/\mu}{2\mu(1-\lambda/\mu)} \tag{2}$$

Substituting $1/\mu$ by P_{TDM}, substituting λ by $\frac{\lambda_{\text{Node}}}{N-1}$, and rearranging the equation, we get Eq. (3). It expresses the average time that a packet waits in the queue before it is injected into the network of routers as a function of the injection rate and the period of the TDM schedule.

$$T_{\text{queue(TDM)}} = \frac{P_{\text{TDM}}}{2(1-\frac{\lambda_{\text{Node}}}{N-1}\cdot P_{\text{TDM}})} \tag{3}$$

The time it takes to cross the network, the NOC-latency in Fig. 4(b), is the time from the packet header enters the source node until the tail of the packet is received by the destination node. This NOC latency depends on: (i) the relative distance between the sender and the receiver, often called the hop count, (ii) the number of pipeline stages in the routers and links, and (iii) the length of a packet. The number of hops is the number of links traversed from source to destination. Most of the (pipeline) registers in a path from source to destination are in the routers, and we note that a path that contains H links contains $H+1$ routers. The average NOC-latency can be calculated from the average hop count, H_{Avg}, the pipeline depth of a router, R, the pipeline depth of the links, L, and the length of a packet S, as follows:

$$T_{\text{NOC}} = [\,(H_{\text{Avg}}+1)\cdot R + H_{\text{Avg}}\cdot L\,] + [\,S-1\,] \tag{4}$$

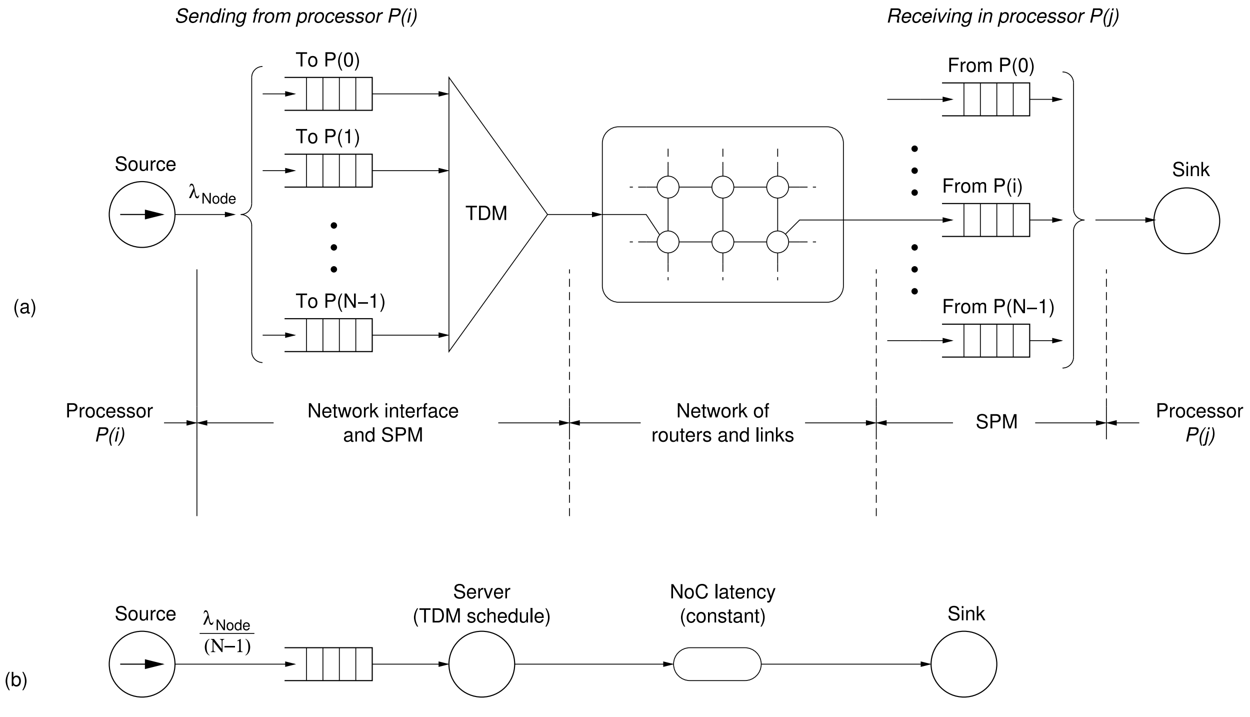


Fig. 4. A model of a TDM NOC suitable for statistical analysis of latency as a function of offered load. Typically, a node does not send to itself ($i \neq j$), hence the injection rate $\frac{\lambda_{\text{Node}}}{(N-1)}$ in Fig. 4(b). The NOC latency is constant for a specific source-to-destination node pair (i,j). Alternatively, in a coarser model, the NOC latency is the average across all the virtual source-to-destination connections.

The first part in square brackets is the number of cycles it takes a first flit in a packet to traverse and exit the network. The second part is the number of cycles it takes the remaining flits in a packet to exit the network.

Adding Eqs. (3) and (4), we obtain the average latency for random uniform traffic across a TDM NOC (expressed in clock cycles):

$$T_{\text{TDM,NOC}} = T_{\text{queue(TDM)}} + T_{\text{NOC}} \tag{5}$$

Note that the periodic operation of the server means that a packet that arrives in an empty FIFO is not serviced immediately, even if the server is idle; it is serviced at the next scheduled point in time. At low packet injection rates, a packet waits on average half of the schedule period, $\frac{1}{2}P_{\text{TDM}}$, before it is injected into the network, and when the injection rate increases, it waits longer, as expressed in Eq. (3). Finally, we note that the latency of the NOC itself, T_{NOC}, is independent of the packet injection rate and constant for a given NOC implementation.

4. Asymptotes of the performance curves

In the next section, we present latency curves for a number of NOC topologies. However, before doing so, it is relevant to calculate the asymptotes, i.e., the *zero-load latency* T_0 and the *saturation throughput* λ_S, in order to instill confidence in the latency-curves produced from our analysis and simulations.

The zero-load latency can be obtained from Eq. (4) for the BE NOC and from Eq. (5) for the TDM NOC. The average hop count for a topology (here, a bi-torus and a mesh) can be calculated as the average of the minimum distance between all possible node pairs. For an N-node network ($N = n^2$) the average hop counts for a bi-torus and a mesh are stated in Eqs. (6) and (7) [11], which do not take the zero-length path from a node to itself into account, meaning that the average is calculated across $N-1$ nodes:

$$H_{\text{Avg, Bi-Torus}} = \begin{cases} n/2 & n \text{ even} \\ n/2 - 1/2n & n \text{ odd} \end{cases} \tag{6}$$

$$H_{\text{Avg, Mesh}} = \begin{cases} 2n/3 & n \text{ even} \\ 2(n/3 - 1/3n) & n \text{ odd} \end{cases} \tag{7}$$

For all network topologies, the bisection bandwidth, B_b, is the primary limiting factor when the number of nodes in the network is not too small [4,9], and it can be used as an upper bound when estimating λ_S. Tighter bounds are possible if the topology, the routing function, and the amount of buffering are also considered, but this requires a detailed analysis. For a bi-torus, a bisection cuts $4n$ links and hence $B_b = 4n$ (flits/s). For a mesh, a bisection cuts $2n$ links and hence $B_b = 2n$ (flits/cycle). Using these values we can estimate $\lambda_{S,\text{ BE, Bi-Torus}}$ and $\lambda_{S,\text{ BE, mesh}}$ as follows:

$$\lambda_{S,\text{ BE, Bi-Torus}} = \frac{B_b}{N/2} = \frac{4n \text{ (flits/cycle)}}{n^2/2} = \frac{8}{n} \text{ (flits/cycle)} \tag{8}$$

$$\lambda_{S,\text{ BE, Mesh}} = \frac{B_b}{N/2} = \frac{2n \text{ (flits/cycle)}}{n^2/2} = \frac{4}{n} \text{ (flits/cycle)} \tag{9}$$

For the TDM NOCs, a tighter bound on λ_S can be determined from the number of nodes, N, the packet size (measured in flits), S, and the period of the TDM schedule (measured in cycles), P_{TDM}:

$$\lambda_{S,\text{TDM}} = \frac{(N-1)\cdot S}{P_{\text{TDM}}} \tag{10}$$

5. Evaluation of typical NOC configurations

In this section, we analyze the performance of a number of NOC configurations. First, we list the parameters characterizing these configurations. Then we compute the asymptotes for the performance graphs and assess the hardware cost. Following this, we present the latency versus offered load curves for the different configurations and explore the effects of varying the packet length. Finally, based on the insights gained, we explore ways of reducing latency and increasing throughput and the effect of using other traffic patterns (tornado and bit complement).

5.1. The NOC simulator TDMSim

The BookSim NOC simulator is flexible and highly configurable. However, it does not support TDM networks. Therefore, we wrote our

Table 1
Zero-load latency, T_0, and saturation throughput, λ_S, for different NOC configurations and packet sizes, S. For the TDM NOC, $T_0 = 1/2\, P_{TDM} + T_{NOC}$, and the two components are shown in columns 4 and 5. As seen $1/2\, P_{TDM}$ contributes the most to T_0.

Topology	Network	S	TDM NOC				BE NOC	
			$1/2\, P_{TDM}$	T_{NOC}	T_0	λ_S	T_0	λ_S
Bi-Torus	8 × 8	3	126	17	143	0.75	24.0	1.0
	8 × 8	17	885	31	916	0.61	40.0	1.0
	15 × 15	3	711	27	738	0.47	38.1	1.0
	15 × 15	17	5203	41	5244	0.37	56.0	1.0
Mesh	8 × 8	3	207	26	233	0.46	32.7	0.5
	8 × 8	17	1367	40	1407	0.39	46.7	0.5
	15 × 15	3	1362	46	1408	0.25	55.8	0.5
	15 × 15	17	9110	61	9171	0.21	69.8	0.5

own simulator TDMSim. TDMSim uses the same traffic generators and time-stamping mechanisms as BookSim. Following the illustration in Fig. 4, it models packet injection into an infinitely long FIFO queue from the C++ containers library representing a single node in the NOC and subsequent packet transmission at appropriate times throughout a TDM period. The network simulation time-stamps a packet when it enters the waiting queue and measures its latency when it exits the NOC. The source code is available at https://github.com/hansemandse/tdmsim

5.2. Network configurations explored

Below we analyze the performance of the following NOC configurations: 64-node mesh and bi-torus NOCs, and 225-node mesh and bi-torus NOCs, all using either 3-flit packets, $S = 3$, (a header and a double-word payload) or 17-flit packets, $S = 17$, (a header and a 16-word payload). The number of 32-bit words in a flit is one, ($F = 1$).

For the BE NOCs, we also vary the number of VCs from 2 to 16. For each of the four situations (topology × number of nodes) with 3-flit packets, we present curves showing the latency of a packet (measured in clock cycles) as a function of the offered traffic, i.e., the injection rate per node (measured in flits/cycle/node), while results for 17-flit packet cases are presented in tabular form. The same BE router is used in all of the following experiments as well.

For the TDM NOC, we assume a router implementation similar to aelite [3] and Argo [5] with three pipeline stages in the router; one stage to traverse the link ($L = 1$) and two stages ($R = 2$) to propagate through the router (header parsing and crossbar traversal). Simpler TDM NOCs, possibly using distributed routing (to avoid header flits), may have $R = 1$ or even $R = 0$, [4]. Later we vary and experiment with different values of all the parameters S, F, R, and L.

For the BE NOCs, we consider the default configuration in BookSim [10] with minor changes to be a standard BE router, and we use this in our comparison. We assume a router with $L = 1$ and $R = 4$ as in [11, Fig. 6.15(b)], where the four pipeline stages are: (i) buffer write, (ii) route computation and VC allocation, (iii) switch allocation, and (iv) switch traversal. Focusing on principles, we assume $L = 1$ for all network topologies and types. We further assume dimension order routing (DOR), eight flits deep VC buffers, and no crossbar speedup (i.e., the crossbar can transfer at most one flit to each router port in a cycle). As mentioned above, we provide results for configurations with 2, 4, 8, and 16 VC buffers per port.

5.3. Values of the asymptotes, T_0 and λ_S

Using the parameters just stated and equations from Section 4, we get the values for T_0 and λ_S shown in Table 1. The values for T_0 and λ_S for the TDM NOCs are tight bounds, whereas λ_S for the BE NOC that is derived from the bisection bandwidth is relatively loose. The actual curves shown in the next section conform with these statements.

The zero-load latency for the BE NOC, $T_{0,\text{BE}}$, is the time it takes a packet to traverse the network of routers and links. The zero-load latency of the TDM NOC, $T_{0,\text{TDM}}$, is higher. The difference is the time a packet waits for admission to the network. On average, this is half of the TDM period, which, as seen, is the major part of the latency.

For large NOCs ($n \times n = 15 \times 15$ nodes), or when using large packets ($S = 17$), the TDM networks exhibit very high zero-load latency (T_0). The reason is the very long schedule periods, as seen in Table 1. This can be explained as follows. For a bi-torus, a bisection cuts $4n$ links and hence $B_b = 4n$ (flits/cycle). This bandwidth must support half of the traffic in the network. For an all-to-all schedule supporting random uniform traffic this is $1/2 \cdot n^2(n^2 - 1) \cdot S$ flits in a schedule period. For the 15 × 15 node bi-torus using 3-flit packets this bisection argument gives a lower bound: $P_{TDM} > 1/8 \cdot n(n^2 - 1) \cdot S = 1200$ cycles. This fits nicely with Table 1 that states $1/2\, P_{TDM} = 711$ cycles. For a torus, a bisection cuts only $2n$ links, and hence P_{TDM} and T_0 are roughly doubled compared to the bi-torus. We discuss ways of reducing P_{TDM} and thereby T_0 later.

The mesh and bi-torus networks analyzed above are the most common NOC topologies. The main takeaway from the analysis is that for a TDM NOC, the zero-load latency, T_0, is dominated by the schedule length, P_{TDM}. The same observation applies to other topologies of TDM networks, for example, ring topologies. For a ring with N nodes, a bisection cuts two links, and these two links must support the transmission of $1/2 \cdot N(N-1)\, S$ flits. For $N = 64$ nodes and $S = 3$ flits in a packet, we can estimate $P_{TDM} > 3024$ cycles. As the ring is a very simple topology, this is a relatively tight bound, and hence $T_0 \approx 1512$ cycles.

5.4. Hardware cost

This paper aims to assess the performance of TDM NOCs and BE NOCs taking hardware cost into consideration. A fair comparison requires precise and objective area measures, and to derive such measures, it is necessary to consider both the micro-architecture and gate-level design as well as the particular technology used for the implementation.

While it is rather straightforward to assess the design of a TDM NOC, due to its simple router, it is more difficult for a BE NOC. The area of a BE router depends on many architectural- and circuit-level details [32], including the number of VC buffers per port, the depth of these VC buffers, and their implementation (arrays of flip-flops, arrays of latches, or SRAM). This large design space for a BE router is an unavoidable source of uncertainty when attempting a comparison.

The next question is what technology (e.g., FPGA, standard-cell ASIC, or full-custom) and what technology node (e.g., 14 nm, 22 nm, 28 nm) to assume. Moreover, if using ASIC or full-custom technology, the design may be optimized for speed, area, or leakage, which also affects the area. All this means that numbers like 1234 slices in a Virtex II FPGA or 123,456 square microns in a 22 nm process are far less accurate than they may seem at first. It also means that results from different papers basically cannot be compared. Finally, such numbers offer little insight.

Instead, we use a more abstract yet precise and transparent measure of area. We observe that all routers are dominated by pipeline registers and VC buffers, and we observe and assume that, to a first-order approximation, the amount of logic is proportional to the amount of storage. In the TDM router, the logic is predominantly in the crossbar. In the BE router, the logic is in the crossbar and the logic administering the VC buffers. In this way, a dimensionless relative area measure is simply the total amount of storage measured in bits. A more precise area measure assumes that the pipeline registers are implemented using discrete flip-flops and the VC buffers as register-files or memory blocks, assumed to be two times denser than flip-flops. Throughout the rest of this paper, we use this more precise, dimensionless relative area measure and denote it $\square$. For all routers and NOC topologies, we assume link traversal is done in one cycle.

Table 2
Hardware resources for FPGA implementation of a single router.

Router	FFs	LUTs	Ratio
Argo	565	932	1:1.6
OpenSoC (2 VCs, 4 flits deep)	2519	4 672	1:1.9
OpenSoC (2 VCs, 8 flits deep)	4337	6 057	1:1.4
OpenSoC (4 VCs, 4 flits deep)	5158	10 935	1:2.1
OpenSoC (4 VCs, 8 flits deep)	8794	13 794	1:1:5

Table 3
Area of a router implemented in the STM 45 nm CMOS cell library.

Router	Cell area (μm^2)
Argo	4 360
OpenSoC (2 VCs, 4 flits deep)	52 023
OpenSoC (2 VCs, 8 flits deep)	79 885
OpenSoC (4 VCs, 4 flits deep)	111 302
OpenSoC (4 VCs, 8 flits deep)	166 975

The baseline TDM router considered in this paper is a five-ported router composed of a 5×5 crossbar, and one, two, or three pipeline stages. The ports are 35 bits wide supporting flits with 32 bits for data/address/route and 3 bits indicating the flit type (header, intermediate, tail). A router with one pipeline stage is discussed in [4], and routers with three pipeline stages (link traversal, extract routing information from header flit, and crossbar traversal) are used in the aelite and Argo networks [3,5]. The dimensionless relative area measure for these routers range from $5 \cdot 35 \cdot 1 = 175$ $\square$ for routers with a single pipeline stage to $5 \cdot 35 \cdot 3 = 525$ $\square$ for routers with three pipeline stages.

As a representative BE router, we consider a 5-ported input-buffered router with 4 VCs per port. We assume 35 bit wide ports, as for the TDM router. We further assume: (i) 8 flits deep VC buffers, (ii) 35 bit flits, and (iii) a router with three pipeline stages in addition to the VC buffers. The crossbar in such a router is either a 20×5 crossbar or a 4:1 multiplexer in each input port followed by a 5×5 crossbar. The relative area measure of this BE router is $3,325$ $\square$ (computed as $5 \cdot 3 \cdot 35 = 525$ $\square$ for the pipeline registers and $5 \cdot 0.5 \cdot 8 \cdot 4 \cdot 35 = 2800$ $\square$ or the VC buffers). If the depth of the VC buffers is reduced to 4 flits, the relative area reduces to 1925 $\square$.

From these area estimates, we see that the area of a typical BE router is 4–10 times larger than a typical TDM router, and the reason is primarily the VC buffers and the related logic.

Finally, we mention the router used in Intel's 48-core IA-32 cloud computer [33] uses 5-ported routers, 144 bit wide flits and links (128-bit data and 16-bit sideband information), 8 VCs per input port with 3 flit deep buffers (implemented as a single register file per port), and a pipeline depth of 4 stages (VC buffer write, switch allocation, buffer read and VC allocation, and switch traversal). For this router, the relative area is: $5 \cdot (3 \cdot 144 + 0.5 \cdot 3 \cdot 8 \cdot 144) = 10,800$ $\square$.

To consolidate and substantiate the above estimates, we have synthesized an Argo TDM router with three pipeline stages and several versions of the BE router from OpenSoc [15,16], targeting both an FPGA and an ASIC implementation. The OpenSoc is written in Chisel [34] and is first translated to Verilog via the Chisel runtime. Argo is written in VHDL and can directly be synthesized

For the FPGA implementation, we used Intel Quartus Prime version 19.1.0 and targeted a Cyclone IV FPGA. The resources offered by this FPGA are flip-flops, lookup tables (LUTs) with up to 4 inputs, and block RAM. The latter is not used by any of the router designs. Table 2 shows the results that confirm that a typical BE router is easily an order of magnitude larger than a typical TDM router and that the amount of combinational logic is proportional to the number of bits stored in flip-flops and buffers. As the logic required to make the VC buffers deeper is limited, the ratio reduces as the VC buffers are made deeper.

For the ASIC implementation, we opted for the STM 45 nm CMOS library of standard cells and we used commercial synthesis and place-and-route tools (Synopsys). Table 3 shows the results. As seen, the different versions of the OpenSoc are 12–48 times larger than the Argo TDM router. This is an even larger difference than our dimensionless estimate and FPGA synthesis results.

5.5. Latency vs. Offered load

Fig. 5 shows the latency versus offered load curves for the configurations mentioned above: 64-node and 225-node mesh and bi-torus topologies using 3-flit packets. We do not include plots for 17-flit packages due to the very high zero-load latency of the TDM NOCs mentioned in Section 5.3.

Results for the BE NOCs are obtained using BookSim [10] and for the TDM NOCs using TDMSim, introduced in Section 5.1. Results from the simulations conform with Eq. (5). The small discontinuities in the plotted lines result from the randomized injection processes in our simulations. Longer simulation runs would reduce these, not change the results. The dashed lines are the asymptotes for the curves for the TDM networks calculated in Section 4. As seen, they conform with the simulations.

We see that the 8×8 bi-torus NOC using TDM routers has a higher saturation throughput than a similar NOC using BE routers with 2 or 4 VCs, and for the 8×8 mesh NOC, we see that all networks have approximately the same saturation throughput. As the area of a TDM router is 5–10 times smaller than a BE router, this is a very positive and interesting result. However, in both 8×8 networks, we also see that the zero-load latencies of the TDM NOCs are 6–7 times higher than for the BE NOCs.

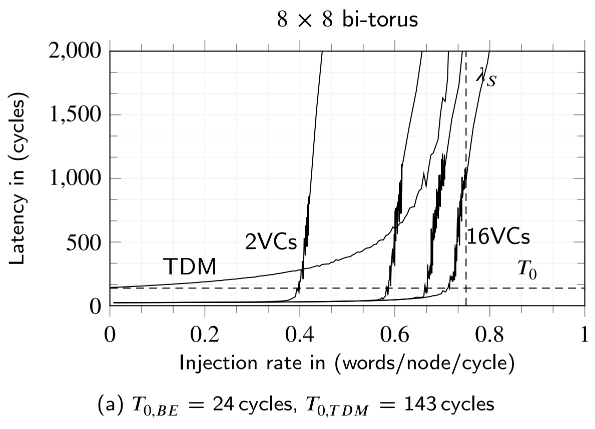
For the larger 15×15 NOCs, the versions using BE routers outperform the versions using TDM routers. As discussed in detail in Section 4, the reason is that the bisection bandwidth in an $N = n \times n$ node mesh or bi-torus grows $\mathcal{O}(n)$, and in the TDM NOCs this bandwidth is statically assigned to the $\mathcal{O}(n^4)$ virtual end-to-end circuits that the NOC implements. Hence, when increasing from $n = 8$ (64 nodes) to $n = 15$ (225 nodes), the bandwidth per virtual end-to-end circuit drops quickly, causing a proportional drop in the saturation injection rate.

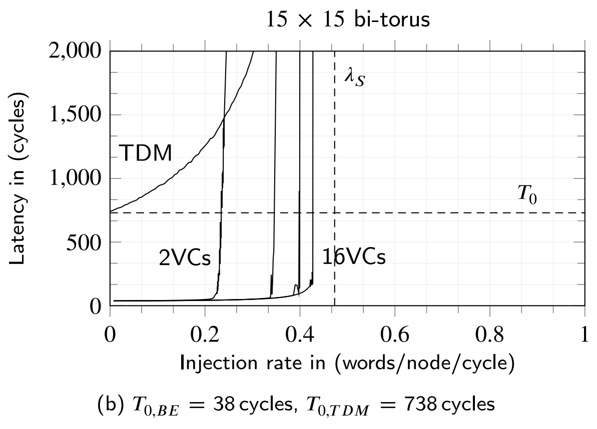
From this first evaluation of what we consider typical examples of BE and TDM NOCs, we conclude that using TDM NOCs for BE traffic is a viable option for NOCs of moderate size using short packets. Compared to an 8×8 BE NOC, an equally-sized TDM NOC can provide comparable throughput for a 5–10 times lower hardware cost at the expense of a 6–7 times higher latency. Larger TDM NOCs (15×15) do not achieve the same results owing to their long schedules. We consider this an interesting point in the solution space, and in the next section, we explore ways of trading more area for a corresponding reduction of latency in 8×8 NOCs; measures that will also increase the throughput proportionally and beyond that of the BE network.

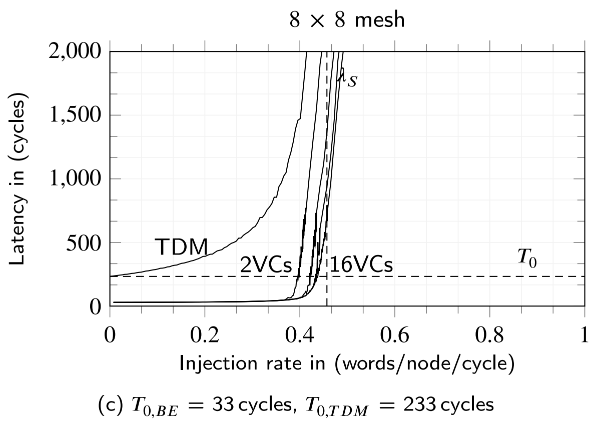
6. Improving latency and throughput

In some BE applications, for example, when supporting traffic between caches and memories, latency rather than throughput is the primary concern. It is therefore relevant to ask how a 2x, 4x, or even 8x increase of hardware resources for a TDM router could be exploited to reduce latency and increase throughput, still with an area that is lower than or similar to that of a BE NOC. In this section, we explore this perspective.

A straightforward way to increase the performance of a TDM router is to use wider links/flits, for example, four words per flit ($F = 4$). Fig. 6 shows the performance of a TDM NOC using the wider links/flits against the BE NOC from Fig. 5(a) and (c). Widening the links by a factor of four means that four times as many words can be transferred within a given time. In Fig. 6, this is seen as a four times higher saturation throughput than before. However, this also implies four times wider registers and multiplexers in the TDM router, roughly quadrupling its area. Considering the area numbers from Section 5.4







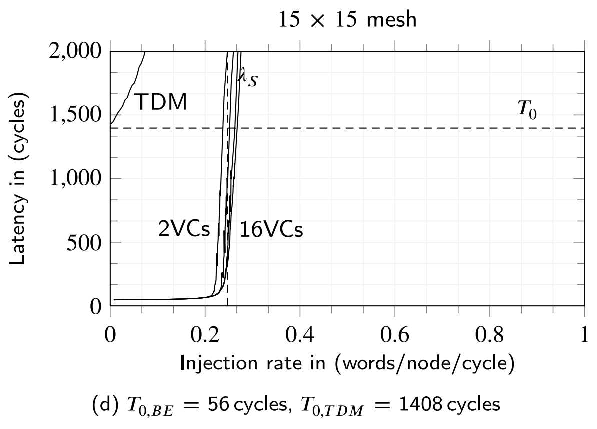
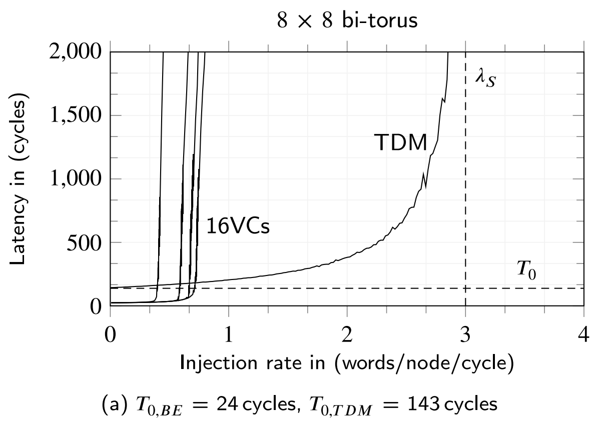


Fig. 5. Latency vs. injection rate curves for the 8 × 8 and 15 × 15 bi-torus and mesh NOCs. The BE NOC has $S = 3$, $F = 1$, $L = 1$, and $R = 4$. The TDM router has $S = 3$, $F = 1$, $L = 1$, and $R = 2$. BE curves are only partially labeled as they appear with increasing number of virtual channels (2, 4, 8, and 16) from left to right. The dashed lines are the asymptotes for the curves for the TDM networks calculated in Section 4. As seen, they conform with the simulations.



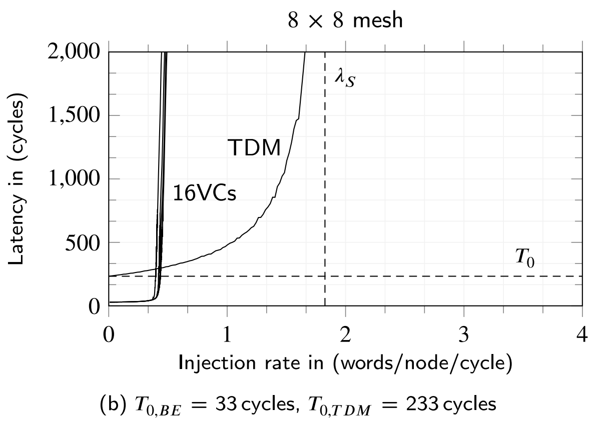


Fig. 6. Latency vs. injection rate curves for 8 × 8 bi-torus and mesh NOCs. The BE NOCs are the same as used before in Fig. 5(a) and (c). The TDM NOC is four times wider than before and has $S = 3$, $F = 4$, $L = 1$, and $R = 2$.

and Tables 2 and 3, the TDM router with wider links would still be smaller than a BE router with 2 VCs.

Exploring how more hardware resources can be used to lower the latency is more interesting and challenging. The average packet latency of a TDM NOC is expressed in Eq. (5). For the 64-node bi-torus NOC explored in the previous section ($S = 3, F = 1, R = 2, L = 1$), the zero-load latency $T_{0,TDM} = 143$ cycles, as stated in Table 1. At low injection rates (zero-load), $T_{\text{queue(TDM)}} = (1/2)\,P_{\text{TDM}} = 127$ cycles. This waiting time in the queue is clearly the dominating contribution to the latency, and it grows as the packet injection rate is increased. The second component, P_{NOC} detailed in Eq. (4), amounts to 14 cycles for a header flit to traverse the NOC and an additional 2 cycles for the remaining flits in a packet to exit the NOC. As these numbers clearly show, the most efficient way of reducing the average latency is to reduce the schedule period.

The schedules for the TDM NOCs are produced using a metaheuristic scheduler [28]. Experiments show that the schedule period, P_{TDM}, depends primarily on the topology (bi-torus or mesh) and the packet length, S. The number of pipeline stages in routers and links, L and R, respectively, play a minor role in determining the schedule.

The observation that the schedule period scales roughly linearly with packet length may not be obvious. However, it may be explained by considering the bisection bandwidth limit. When using random uniform traffic, half of the traffic in the network is carried by the links

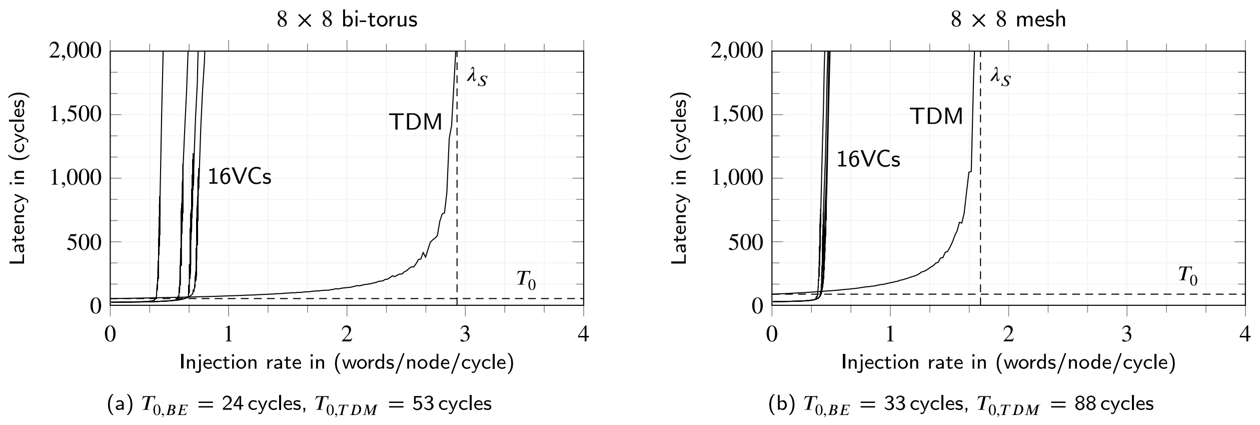


Fig. 7. Latency vs injection rate curves for 8 × 8 bi-torus and mesh NOCs using 4-word single-flit packets. The BE NOC is the same as used before in Fig. 5(a) and (c). The TDM NOC has $S = 1$, $F = 4$, $L = 1$, and $R = 1$.

that cross a bisection cut, and from this, it is trivial to realize that when these links are the bottleneck, then the time it takes to transmit a given number of packets increases linearly with packet length.

From the above discussion, we see that the most effective way to reduce latency is to reduce the packet length, possibly to a single flit ($S = 1$), because it reduces the schedule period almost proportionally. The packet size can remain unchanged if the flits and links, and thereby also the routers, are made correspondingly larger. This will reduce $\frac{1}{2} P_{\text{TDM}}$ from 126 cycles to 42 cycles for the 64-node bi-torus.

With packet length and schedule period at a minimum, the next step towards reducing latency is to reduce the number of pipeline stages in the routers and links. These pipeline stages are "used" for signals to travel a certain distance or for calculations and switching. Compared to BE routers, TDM routers perform little decision-making and only very simple switching, and consequently, most pipelining is related to signal propagation. For this reason, TDM routers generally may have fewer pipeline stages than BE routers.

In all of the above, we have assumed a TDM router using a 3-stage pipeline ($L = 1$ and $R = 2$). The reason is that this is what is used in the aelite [3] and Argo [5] NOCs. The three pipeline stages are (i) link traversal, (ii) extract routing information, and (iii) crossbar traversal. Stage (ii) is very simple, and it is possible to combine stages (ii) and (iii) and reduce the router pipelining to two stages: link traversal and router traversal. It may even be possible to reduce to a single pipeline stage for both if aggressive clocking is not needed.

Based on the insight presented in this section, we can now propose a TDM router targeting minimum latency: it uses single flit packets, wide flits/links (4 words or more), and two pipeline stages (link traversal and switch traversal), i.e., configuration ($S = 1$, $F = 4$, $L = 1$, $R = 1$). We estimate the area of this router to be still 2.5–3 times smaller than a typical BE router. Fig. 7(a) and (b) show the latency versus offered load curves for the 8 × 8 mesh and bi-torus TDM NOCs using this configuration. We note that this design achieves a comparable zero-load latency and a much higher throughput than the BE router.

Finally, we offer some ideas for further research, which can reduce the schedule period and thereby the latency below what is possible when using single flit packets. A general performance-increasing technique used in hardware design when circuit delays do not allow further pipelining is interleaving, i.e., using multiple copies of the circuit and operating them in an interleaved fashion. In a TDM NOC, this basic idea can be implemented in several alternative ways. One option is to implement a second copy of the network of routers and links and to execute their TDM schedules phase shifted by half of the TDM period. Each queue will then provide packets to both networks, effectively halving the waiting time between slots and thereby the latency. Another option is to implement two networks, each servicing traffic from half of the nodes. Both options will require some changes in the network interfaces connecting processor nodes to their corresponding routers, but the hardware cost of the network interfaces will only increase marginally.

7. Other traffic patterns

Up to now, we have assumed random uniform traffic. This is considered benign towards the BE NOCs [9] because traffic spreads across the network. To supplement the analysis, we now consider tornado traffic for the bi-torus NOC and bit-complement traffic for the mesh NOC. These traffic patterns are designed to be challenging for the respective networks [9]. In both patterns, every node transmits packets to one destination node, whose address is calculated from the address of the source node. For tornado traffic the nodes are considered in two dimensions (s_x, s_y) for $x, y \in [0, n-1]$ such that the destination node is $(d_x, d_y) = ([s_x + \lceil n/2 \rceil - 1] \mod n, [s_y + \lceil n/2 \rceil - 1] \mod n)$. For bit-complement traffic, all nodes are assigned a unique number s_i for $i \in [0, n^2 - 1]$ such that the destination node is $d_i = \neg s_i$. In both cases, we consider networks with $8 \times 8 = 64$ nodes.

The tornado and bit-complement traffic patterns can be supported by a TDM network using the same all-to-all schedule that we used for random uniform traffic. However, in this case, only 64 of the 64 × 63 virtual end-to-end circuits would be used, resulting in poor utilization of network resources and hence poor performance. For sparse communication patterns like tornado and bit-complement, it is relevant to consider specific schedules for the traffic patterns. Here we learned a lesson.

To begin, we defined core-communication graphs for bit-complement and tornado with one edge between every source–destination node pair. From this, and assuming the same router as in Section 6 ($S = 1$, $F = 4$, $L = 1$, and $R = 1$), we obtained TDM schedules of 18 cycles for the bi-torus NOC and 42 cycles for the mesh NOC. In both cases, the performance was still poor. The reason is that all packets from a source to a destination follow the same shortest path, leaving a lot of NOC resources, i.e., links, idle or partially idle.

To overcome this, we instead specified core communication graphs with, for example, 16 edges between every source–destination node pair, enabling the scheduler to use several different shortest-path routes between communicating nodes. This results in better utilization of network resources and thereby improved performance (higher throughput and lower latency) while requiring no changes to the NOC routers and only negligible changes to the network interfaces. It should be noted that the scheduler is limited to producing shortest-path routes, and hence that all packets arrive in order despite having followed different paths.

The final results for the situation where a schedule period allows nodes to send 16 packets are shown in Fig. 8(a) for a bi-torus exposed

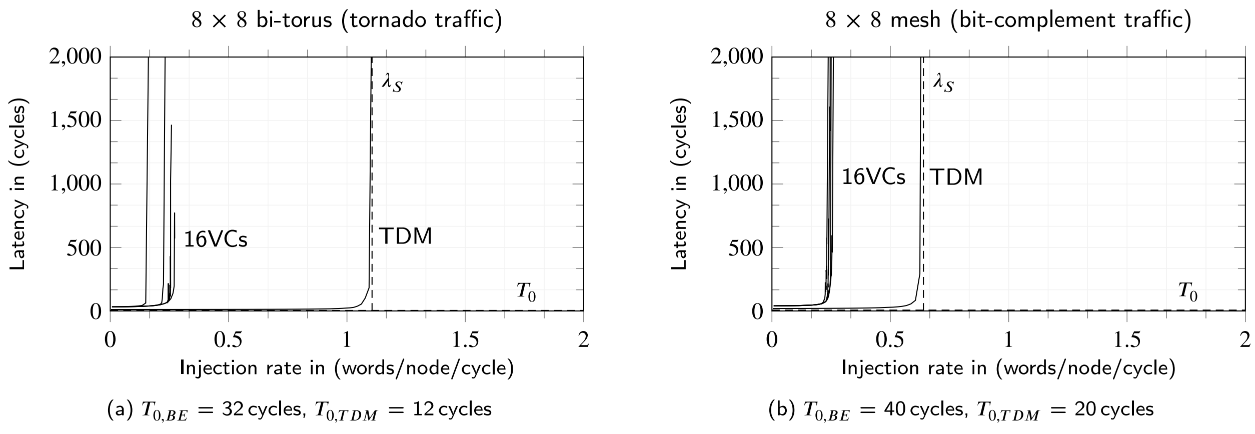


Fig. 8. Latency vs injection rate curves for 8 × 8 bi-torus and mesh NOCs with application-specific tornado and bit-complement traffic patterns. The BE NOC is the same as used before in Fig. 5(a) and (c). The TDM NOC has $S = 1$, $F = 4$, $L = 1$, and $R = 1$. Labeling follows the same pattern as in Fig. 5.

to tornado traffic and in Fig. 8(b) for mesh NOC exposed to bit-complement traffic. As seen, the TDM NOCs outperform the BE NOCs in terms of zero-load latency and saturation throughput.

As an overall conclusion to this experiment, we note that TDM NOCs provide very attractive performance when using application-specific schedules but that their schedules must be unrolled to better use the available bandwidth.

8. Conclusion

Motivated by the observation that the hardware cost of a typical time-division multiplexed (TDM) network-on-chip (NOC) [3,5] is 5–10 times lower than that of a typical best effort (BE) NOC [10,11], this paper explores and compares the performance of the two. To the best of our knowledge, such a comparison has not been attempted before.

The performance is expressed using graphs showing average packet latency versus offered load. For the BE NOCs, we use the BookSim simulator [10] to obtain these graphs. For the TDM NOCs, we derive and contribute a queuing theory model and an associated TDM NOC simulator that uses the same traffic generators and time-stamping mechanism as BookSim.

Our first experiments use random uniform traffic and show that for a *fraction of the hardware cost*, and for networks with up to 64 nodes, a TDM NOC can offer *similar saturation throughput*, albeit with significantly higher latency.

The TDM routers in [3,5] are not optimized for minimum latency, and in the second set of experiments, we explore design tradeoffs that minimize the latency of a TDM NOC. Here we find that short packets and correspondingly wider links/flits significantly reduce the TDM schedule period and thereby the latency. In this way, and for a hardware cost that is still smaller than that of a BE router, a TDM NOC can reach almost the same latency and four *times* higher bandwidth.

As a final experiment, we explored using other traffic patterns, i.e., tornado and bit-complement. Here, we learned that a TDM NOC should use longer (unrolled) schedules, where source nodes are assigned several slots for sending packets to the same destination nodes. This uses more paths through the network and results in correspondingly higher utilization of network resources.

The overall conclusion is that a TDM NOC, with its small and simple hardware, may be attractive in multi-core platforms used for BE applications.

In future work, we envision exploring real applications on BE and TDM NOCs. Furthermore, we plan to explore TDM NOC architectures with support for single-cycle multi-hop links, particularly considering the effects this has on hardware complexity and schedule length. Another direction of future work is using application-specific schedules, reconfiguration of the TDM schedule, and other forms of dynamic adaptation to actual traffic loads.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The paper includes a link to the source code for the TDM NOC simulator.

Acknowledgments

The work presented in this paper was partially funded by the Danish Council for Independent Research | Technology and Production Sciences, Denmark under the project PREDICT, contract no. 4184-00127A.

We thank William Wulff and Prof. Alberto Nanarelli for their help on producing the ASIC area figures in Table 3.

Finally, we thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions that have helped improve the paper.

References

[1] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, The Nostrum backbone - a communication protocol stack for networks on chip, in: Proc. IEEE Intl. Conference on VLSI Design, 2004, pp. 693–696.
[2] K. Goossens, J. Dielissen, A. Rădulescu, The Æthereal network on chip: Concepts, architectures, and implementations, IEEE Des. Test Comput. 22 (5) (2005) 414–421, http://dx.doi.org/10.1109/MDT.2005.99.
[3] A. Hansson, M. Subburaman, K. Goossens, Aelite: a flit-synchronous network on chip with composable and predictable services, in: Proc. Design, Automation and Test in Europe, DATE, 2009, pp. 250–255.
[4] M. Schoeberl, F. Brandner, J. Sparsø, E. Kasapaki, A statically scheduled time-division-multiplexed network-on-chip for real-time systems, in: Proc. IEEE/ACM Intl. Symposium on Networks-on-Chip (NOCS), IEEE Computer Society Press, 2012, pp. 152–160.
[5] E. Kasapaki, M. Schoeberl, R.B. Sørensen, C.T. Müller, K. Goossens, J. Sparsø, Argo: A Real-Time Network-on-Chip Architecture with an Efficient GALS Implementation, IEEE Trans. VLSI Syst. 24 (2) (2016) 479–492.
[6] S. Hesham, J. Rettkowski, D. Goehringer, M.A. El Ghany, Survey on real-time networks-on-chip, IEEE Trans. Parallel Distrib. Syst. 28 (5) (2017) 1500–1517.
[7] K. Goossens, A. Hansson, The aethereal network on chip after ten years: Goals, evolution, lessons, and future, in: Proc. ACM/IEEE Design Automation Conference, DAC, 2010, pp. 306–311.
[8] E. Kasapaki, J. Sparsø, Argo: A Time-Elastic Time-Division-Multiplexed NOC using Asynchronous Routers, in: Proc. IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC, IEEE Computer Society Press, 2014, pp. 45–52.
[9] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Elsevier Science Publishers, 2003.

[10] N. Jiang, J. Balfour, D.U. Becker, B. Towles, W.J. Dally, G. Michelogiannakis, J. Kim, A detailed and flexible cycle-accurate network-on-chip simulator, in: Proc. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), IEEE, 2013, pp. 86–96.
[11] N. Enright Jerger, T. Krishna, L.-S. Peh, On-Chip Networks (Second Edition), Morgan Claypool, 2017, pp. 1–212.
[12] L. Benini, G.D. Micheli, Networks on chips: A new SoC paradigm, Computer 35 (1) (2002) 70–78.
[13] W. Dally, Route packets, not wires: On-chip interconnection networks, in: Proc. Design Automation Conference, DAC, ACM Press, New York, 2001, pp. 684–689.
[14] R.S. Ramanujam, V. Soteriou, B. Lin, L.S. Peh, Extending the effective throughput of NoCs with distributed shared-buffer routers, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 30 (4) (2011) 548–561.
[15] F. Fatollahi-Fard, D. Donofrio, G. Michelogiannakis, J. Shalf, OpenSoC fabric: On-chip network generator, in: Proc. IEEE Intl. Symposium on Performance Analysis of Systems and Software (ISPASS), 2016, pp. 194–203.
[16] Online, OpenSoC fabric version 1.1.3, 2021, http://www.opensocfabric.org, accessed August 2021.
[17] R. Mullins, A. West, S. Moore, Low-latency virtual-channel routers for on-chip networks, in: Proc. International Symposium on Computer Architecture (ISCA), 2004, pp. 188–197.
[18] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C.R. Das, A low latency router supporting adaptivity for on-chip interconnects, in: Proc. Design Automation Conference, DAC, 2005, pp. 559–564.
[19] A. Psarras, I. Seitanidis, C. Nicopoulos, G. Dimitrakopoulos, ShortPath: A network-on-chip router with fine-grained pipeline bypassing, IEEE Trans. Comput. 65 (10) (2016) 3136–3147.
[20] A. Ejaz, V. Papaefstathiou, I. Sourdis, FreewayNoC: A DDR NoC with pipeline bypassing, in: Proc. ACM/IEEE International Symposium on Networks-on-Chip, NOCS, 2018, pp. 1–8.
[21] A. Ejaz, V. Papaefstathiou, I. Sourdis, HighwayNoC: Approaching ideal NoC performance with dual data rate routers, IEEE/ACM Trans. Netw. 29 (1) (2021) 318–331.
[22] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip, in: Proc. Design, Automation and Test in Europe, DATE, IEEE Computer Society Press, 2004, pp. 890–895.
[23] C. Gibbs, Statistical multiplexing in data-networks, Comput. Commun. 4 (6) (1981) 281–285.
[24] K. Chandra, Statistical time division multiplexing, in: Handbook of Computer Networks, Vol. 1, John Wiley and Sons, 2011, pp. 579–590.
[25] W. Jiawen, L. Li, Z. Yuang, P. Hongbing, H. Shuzhuan, Z. Rong, Statistical time division multiplexing based local system architecture for multi-cluster NoC, in: Proc. IEEE International Conference on Communication Software and Networks (ICCSN), 2011, pp. 472–476.
[26] S. Murali, G. De Micheli, Bandwidth-constrained mapping of cores onto NoC architectures, in: Proc. Design, Automation and Test in Europe, DATE, 2004, pp. 896–901.
[27] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, G.D. Micheli, NoC synthesis flow for customized domain specific multiprocessor systems-on-chip, IEEE Trans. Parallel Distrib. Syst. 16 (2) (2006) 113–129.
[28] R.B. Sørensen, J. Sparsø, M.R. Pedersen, J. Højgaard, A metaheuristic scheduler for time division multiplexed networks-on-chip, in: Proc. IEEE/IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS), 2014, pp. 309–316.
[29] D. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain, Ann. Math. Stat. 24 (3) (1953) 338–354.
[30] R. Cooper, Introduction to Queueing Theory, 2. Ed, North-Holland, 1981, p. 347.
[31] J.F. Shortle, J.M. Thompson, D. Gross, C.M. Harris, Fundamentals of Queueing Theory: Fifth Edition, wiley, 2017, pp. 1–548, http://dx.doi.org/10.1002/9781119453765.
[32] A.B. Kahng, B. Lin, S. Nath, ORION3.0: A comprehensive NoC router estimation tool, IEEE Embed. Syst. Lett. 7 (2) (2015) 41–45.
[33] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, N. Borkar, A 2 Tb/s 6 x 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS, IEEE J. Solid-State Circuits 46 (4) (2011) 757–766.
[34] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avizienis, J. Wawrzynek, K. Asanovic, Chisel: constructing hardware in a Scala embedded language, in: Proc. Design Automation Conference (DAC), ACM, San Francisco, CA, USA, 2012, pp. 1216–1225.

Jens Sparsø is professor at the Technical University of Denmark (DTU). His research interests include: design of digital circuits and systems, design of asynchronous circuits, low-power design techniques, application-specific computing structures, multi-core processors, and networks-on-chips; in short, hardware platforms for embedded and cyber–physical systems.

He has published more than 100 refereed conference and journal papers and is coauthor of the book "Principles of Asynchronous Circuit Design - A Systems Perspective" (Kluwer, 2001), which has become the standard textbook on the topic. He received the Radio-Parts Award and the Reinholdt W. Jorck Award in 1992 and 2003, in recognition of his research on integrated circuits and systems. He received the best paper award at ASYNC 2005, and one of his papers was selected as one of the 30 most influential papers of 10 years of the DATE conference.

Hans Jakob Damsgaard received the B.Sc. degree in electrical engineering in 2019 and the M.Sc. degree in computer science and engineering in 2021 as part of the Honours programme at the Technical University of Denmark, DTU. Currently, he is pursuing a PhD on approximate reconfigurable accelerators for secure edge computing as part of the APROPOS project at Tampere University. His research interests include hardware accelerators, networks-on-chip, and computer architecture.

Dimitrios Katsamanis is a FPGA engineer at Arm Sweden AB. He is currently working on FPGA images for verification and software development. Prior to that he was a graduate student at the Technical University of Denmark (DTU) where he did his master thesis on High Level Synthesis. He carried out his undergraduate studies at the Democritus University of Thrace (DUTH) in Greece, and worked on network interfaces and network on-chips for integrated circuits.

Martin Schoeberl received his PhD from the Vienna University of Technology in 2005. From 2005 to 2010 he has been assistant professor at the Institute of Computer Engineering. He is now professor at the Technical University of Denmark. His research interest is on hard real-time systems, time-predictable computer architecture, and real-time Java. Martin Schoeberl has been involved in a number of national and international research projects: JEOPARD, CJ4ES, T-CREST, RTEMP, the TACLe COST action, and PREDICT. He has been the technical lead of the EC funded project T-CREST. He has more then 100 publications in peer reviewed journals, conferences, and books.