CHAPTER 5

NETWORK AND TRANSPORT LAYERS IN NETWORKS ON CHIP*

The main goal of the network and transport layers is to support the end-to-end communication between the modules at the specified *quality of service* (QoS) using a power- and resource-efficient sharing of the interconnect resources. This seemingly simple goal forces the designer to address all classical (and some new) network layer issues in the context of an individual chip design.

In order to define the end-to-end QoS, we need to define the on-chip communication requirements in terms of the module-to-module traffic types, rates, statistical behavior, and predictability. For each such end-to-end *flow* we also need to define the service it receives such as loss and delay. We also need to define the interrelation between these flows, such as priorities, and actions that should be taken when communication resources are scarce. Finally, we need to decide on the appropriate network architecture that supports the above QoS requirements. For this the following *network on chip* (NoC) characteristics must be defined: switching technique, NoC topology, addressing and routing scheme, and end-to-end congestion and flow control schemes.

First, the appropriate data switching mechanism (e.g., circuit switching, packet switching, etc.) needs to be selected for the multiplexing of multiple flows from different sources and different requirements in a single network. Then, the appropriate network topology needs to be selected and optimized to physically connect the different modules (including basic graph topology, links and router speeds, and specific layout issues). Addressing and routing schemes need to be devised in order to allow circuit or packets traversing the network to be routed to diverse destinations including possible multicast or broadcast of information. Names meaningful to applications (such as memory and I/O addresses) need to be translated into routing-efficient labels. Since the proper delivery of certain types of traffic (or signals) on the chip is crucial for performance and is categorized in different ways, the concurrent support of multiple QoS

* This chapter was provided by Israel Cidon, of Technion, Israel and by Kees Goossens, of Philips Research, The Netherlands

requirements is essential. Since modules perform a variety of information exchanges, there is a need for end-to-end mechanisms to guarantee in order and assure delivery, end-to-end connection and flow control for rate matching and receiver buffer management and resource access control for multiple resource access resolution. In order to accommodate excessive traffic conditions and to quantify the behavior of the network under extreme conditions, NoC should also support network-level congestion control. One also needs to address reliability in the face of communication soft errors that may corrupt transmitted data.

5.1 NETWORK AND TRANSPORT LAYERS IN NoCs

The network layer deals with the QoS, switching technique, topology, and addressing and routing schemes. The transport layers address the congestion and flow control issues. However, we will not make the distinction in the remainder of this chapter.

Since similar network and transport layer problems have been extensively studied in the networking and system interconnect realm, one may be tempted to employ well-developed networking solutions in the NoC context. However, a direct adaptation of such network solutions to NoCs is impossible, due to the different communication and performance requirements, cost considerations, and architectural constraints [21].

First, the requirements are different. Unlike many off-chip networks, the NoC is at the heart of chips that support real-time operations and are embedded in critical systems from life-support gear to vehicular and aerospace equipment. Embedded systems also often deal with intrinsically real-time data, such as high-quality audio and video. Therefore, NoCs QoS requirements call for a high degree of predictability and robustness and cannot tolerate incidental glitches and anomalies [32]. As a specific example, network design must meet strict QoS requirements for certain types of traffic, such as interrupt signals or fetching real-time instructions and data from caches to *digital signal processors (DSPs)*. Such requirements may be specified in very strict terms, such as the number of clock cycles to accomplish a certain transaction.

Second, the cost considerations are different. The primary considerations in *very large-scale integration* (VLSI) are minimizing area and power dissipation. The area cost of an NoC is composed of routers and network interfaces (i.e., logic cost) and the cost of wires/links that interconnect them (area used by metal lines, spacing, shielding, repeaters, etc.). Similarly, power consumption can be separated to dynamic and static power consumed by the NoC logic (in routers and network interfaces) and links (in wires and repeaters). In both cases, both temporal power to reduce heat dissipation, and total energy consumption for saving battery power for nomadic systems [12] are to be minimized. Moreover, since an NoC

connects modules or relatively small subsystems, these area and power costs should be kept much lower compared to networks that connect large systems.

Third, there are also many architectural constraints and freedoms unique to the NoC environment. For example, on-chip network topologies are quite restricted – they are planar, often organized as (full or partial) grids, and do not need to support the dynamic addition or removal of components. A new important dimension of freedom in NoC is the ability to alter the physical layout of the network routers and links along with the chip module placements, enabling the designer to optimize the NoC geography according to traffic and layout constraints [13, 78, 80].

Furthermore, each NoC is synthesized anew for each design [43, 62], eliminating the need for standardization of network protocols. That is, protocols and architectures employed in a new chip design do not have to be compatible with those used in previous designs. Consequently, NoCs do not possess the rigid standardization constraints of traditional networks. Except for retaining module reuse compatibility, NoCs can be customized to their specific chip environment and need not assume backward, upward, and different vendor compatibility as well as regulatory constraints. In that sense the NoC environment is much more open to multiple choice and architectural innovations. Nevertheless, the use of standard network interfaces, such as *Open Core Protocol* (OCP) [86] and *Advanced extensible Interface* (AXI) [2] is important, in order to allow the reuse of modules, across chip designs.

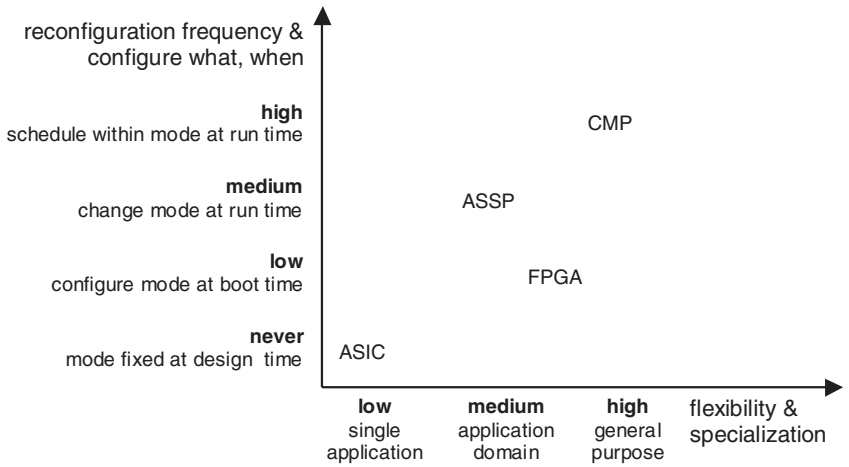
5.1.1 Classifying NoC Models

As mentioned in Chapter 1, Networks on chip can be classified into different families. We elaborate more on this concept here, and relate to a classification of chip designs based on Ref. [21] and on its impact on NoC design. *Systems on chip* (SoCs) span a vast spectrum of objectives and implementations.

On one extreme of the spectrum we identify the *application-specific* SoCs or ICs (ASICs), encompassing custom-made designs with a particular pre-known application, for example, a multichannel 3G base-station, a video camera, or a set-top box. In such chips, the network usage is known *a priori* and can be classified prior to the NoC design time.

On the opposite side of the spectrum, general-purpose *chip multiprocessors (CMPs)*, capture general-purpose chip designs supporting parallel processing, where the chip and interconnect usage is unpredictable and only determined at run time. Here we can find general multi-core processors, parallel DSP arrays, etc.

In between these two extremes we can find designs whose traffic patterns can be partially predicted, or may have several distinct usage patterns. These systems are the outgrowth of *application-specific standard*



■ **FIGURE 5.1**

An NoC classification.

products (ASSP); also called *application-specific instruction set processors* (ASIPs), which are processors whose architecture has been tuned to an application. SoCs may contain multiple processors and mix general-purpose and application-specific processors. In general, it is commonplace to refer to *platforms* as to multiprocessors dedicated to a family of applications.

The *field-programmable gate arrays (FPGAs)* define multipurpose chips that include generic hardware resources like logic arrays, flip-flops, RAM, processors, and special purpose *Internet Protocol* (IP) modules (Ethernet, USB, DSP) that can be *configured* using a programmable interconnect grid, into specific systems. The design of such systems leaves a large degree of freedom for the FPGA designer (programmer).

We illustrate the NoC design spectrum in Fig. 5.1.

ASIC

The distinguishing property of our ASIC NoC definition is the ability to predict the network usage patterns before the network is designed. Since a typical SoC has a specific functionality, this functionality can be simulated, and the inter-module communication patterns and requirements can be inferred at design time. Consequently, a specific NoC can be synthesized to satisfy these exact needs. Virtually all network parameters, including network layout, link capacity allocation, buffer sizes, packet headers, partial routing tables, and the number and requirements of service classes, can be custom synthesized to meet the particular SoCs requirements with no "open-ended" resources spent on "general-purpose" requirements. Specifically, most SoCs do not need to support all possible module-to-module

communication patterns as well as all possible QoS classes for all modules or SoC vicinities.

ASICs are optimized for the application, and hence will have the smallest area and use the least power. This comes at the cost of reduced flexibility. Related to this is the cost of designing an ASIC, which is incurred most often of all categories, namely for each application.

ASSP and platforms

A slightly more complex scenario is dealing with SoCs that are designed to support multiple applications, or that can take different chip "incarnations" at either production or power-up time. In such SoCs different modules and different software may be operating in each distinct SoC incarnation. A major design objective is to build a single SoC for a broad range of related applications [99]. In such a case the NoC is designed to support all possible applications. However, for each specific incarnation the NoC can be configured to operate at minimum power consumption. Therefore, non-required elements can be turned off and dynamic voltage and frequency scaling (DVS/DFS) techniques can be used to tune the NoC for an optimal performance versus power trade off [79].

ASSPs/platforms are tailored to an application domain (e.g., high-end set-top boxes), rather than to a particular application. They are less area and power efficient than an ASIC for each application. However, the increased flexibility is rewarded by reuse of the chip for multiple applications. As a result, the cost of designing the ASSP/platforms is paid only once per application domain.

FPGAs

An FPGAs NoC is designed in two phases: (i) the *layout phase*, which occurs when the FPGA chip is designed and (ii) the *configuration phase*, occurring when a system design is programmed into the FPGA. The design made in the former should be flexible enough so as to allow for a large variety of configurations during the latter. Unlike in the ASIC model, little is known about traffic patterns when the network is laid out, and hence, custom optimizations are impossible. In the configuration phase the traffic patterns become available, but the physical wires and network resources are already fixed in place.

The NoC infrastructure can be designed into the fixed FPGA fabric during the layout phase. But it can also be programmed, just like any other functionality, during the configuration phase. In the former case, the communication patterns of the application to be programmed in the FPGA are not yet known, and the NoC must therefore be a general one (like for CMPs, described below). But the implementation of this NoC can be highly optimized, for example using custom layout. The reverse holds for the latter, where the application and its communication patterns are known, and an NoC optimized for the application can be programmed in the FPGA. The

NoC implementation will therefore use the generic FPGA infrastructure (look-up tables, switch boxes, etc.), just like any other module functionality. As a result, the NoC implementation will be much less optimized. Of course, a mixture of these two models is also possible. For example, an NoC backbone (routers and network interface kernels) is designed in the FPGA, which is augmented with network interface shells [91] during configuration.

FPGAs are optimized for hardware-programmable flexibility. Hence, they are very area and power inefficient compared to an ASIC for each application. However, their computational efficiency is good compared to ASICs and ASSPs. Because the FPGA is designed once and then programmed, it is able to run all applications. The design cost of an FPGA is amortized over a larger number of applications than ASICs and ASSPs.

CMPs

In a general-purpose CMP, the traffic pattern is completely unpredictable until run-time and even during different phases of the same program because of very different program behaviors for different external inputs (e.g., dynamic games, simulations, etc.). In this regard, many of the challenges of CMP NoC design resemble those we normally see in traditional networks, for example, static versus dynamic routing, congestion control, connection rate fairness, etc. Unfortunately, traditional mechanisms for dealing with these issues may be prohibitively expensive to implement in silicon. On the other hand, the relatively small dimension of the complete network, combined with the fact that the entire chip is often controlled by a single operating system, may make the problems amenable to centralized solutions.

CMPs are optimized for software-programmable flexibility. Hence, they are very area and power inefficient compared to an ASIC or ASSP for each application. The CMP is designed once (in theory) and is able to run all applications. Of the four categories, the design cost of a CMP is amortized over the largest number of applications.

In the sequel we present the different aspects of the network layer architectures and their design choices: the QoS and traffic requirement characteristics, the switching paradigm, NoC topologies, the addressing and routing mechanisms, handling congestion control, and end-to-end transport layer issues.

5.2 NoC QoS

There are many important metrics for NoCs. In this chapter, we devote particular attention to QoS, because this requirement has a direct impact

on switching and routing. QoS is a common networking term that applies to the specification of network services that are required and/or provided for certain *traffic classes* (i.e., traffic offered to the NoC by modules). QoS specification can be expressed by performance metrics like loss, rate, delay, delay variation (jitter), etc. and can be categorized by absolute (worst case) bounds, average values, percentiles etc. [4]. For example, an Internet packet-based voice traffic may require a throughput of 64 Kbps, less than 1% average packet loss and a packet one way delay bound below 150 ms. QoS definitions can also apply to different service granularities. In addition to the QoS specification of voice packets, the whole voice call can require a limit on its call blocking probability (failure rate) or a specification of the call establishment times.

If certain traffic classes require an assured (guaranteed) QoS specification, there is an inherent limit to the amount of traffic that can be allocated to such a class. In other words, when there are more potential users than the network can support, some of them will be refused access to the network. Therefore, in our voice examples, the network is designed for a limited voice offered load. In case there is a risk that the user load may overpass this limit, special provisions need to be taken in order to block the excess offered load (e.g., a busy signal for telephone users).

The NoC applications (such as DSP, consumer electronics, etc.), usually stress the need for stringent bandwidth and/or latency requirements. NoC traffic can consist of urgent tasks such as code fetch following a cache miss, CMP synchronization signals or periodic refreshed data that need a guaranteed latency. Other urgent tasks may require similar latency with a high probability. Yet streaming traffic, such as audio and video, has strict bandwidth and jitter requirements, but tolerates long latencies [46]. On the other hand, certain traffic classes may not be planned for a guaranteed service (as they may produce an unspecified amount of traffic) but may require a graceful degradation of service as well as a fair allocation of the available resources as offered load increases.

The mixture of different service classes in the same network poses a special challenge to the NoC designer as each class may affect the performance of other classes. In other words, QoS also specifies how to allocate network resources at times of conflict over the network resources. Therefore, QoS mechanisms usually combine mechanisms for traffic discrimination (such as applying delay or loss priorities [18]), mechanisms for traffic separation (such as fair queueing mechanisms [48]), and mechanisms for traffic policing (enforcing limitations of traffic generation at the source modules). We return to some of these issues in Sections 5.4, Switching techniques and 5.7, Congestion control and flow control.

While the available QoS depends on the overall NoC architectural components such as topology, link and router capacities, routing algorithms, and congestion control mechanisms, it is common to separate the issue of QoS from other network issues. The rationale is the following: given

a complete network architecture we would like to be able to serve multiple traffic classes over the same network each with a different QoS specification. To achieve this, we need special mechanisms that isolate and differentiate these traffic classes within the network. These specific mechanisms are described in this section.

5.2.1 Traffic Classes and Service Classes

The definition of traffic classes is an open-ended task and NoC designers can specify and support any number of classes. However, practice has shown that the specification, management and implementation of a large number of classes are complex and ambiguous task. First, it is hard to predict ahead of time all the future possible NoC usages. Here we should differentiate between different NoCs in our design spectrum where ASIC and ASSP NoCs are much more predictable than CMP NoCs. Second, the exact interrelationships between the classes are hard to define. Finally, the mechanisms that allocate the exact performance to each class in a multiple performance metrics domain (loss, rate, delay, jitter, etc.) are complex and costly.

Traffic classes and service classes in computer networks

Consequently, most off-chip network standards have aggregated the large number of possible traffic classes (traffic offered to the NoC by the modules) into a few predefined *service classes* (offered by the NoC to the modules). Let us illustrate the way the *asynchronous transfer mode* (ATM) and IP networking standards have classified the different traffic classes.

The ATM network defines five possible standard classes [74]:

1. *Constant bit rate (CBR)*: associated with traditional time division multiplexing (TDM) services where delay and loss are fixed and small.
2. *Variable bit rate – real time (VBR-RT)*: associated with variable rate (usually compressed) real-time services that produce a predictable average rate and require low delay and jitter and can tolerate small loss.
3. *Variable bit rate – non-real time (VBR-NRT)*: associated with less interactive real-time services such as streaming or high-priority data services.
4. *Available bit rate* (*ABR*): includes most data services. The term available bit rate means that under bandwidth shortage conditions, the bandwidth allocated to this class should be allocated fairly among competing applications within this class.

5. *Best effort (BE)*: low-revenue and low-priority data that can serve as "bandwidth gap filler" such as file sharing or background synchronization traffic.

In a typical ATM implementation, the CBR and VBR service classes are accomplished via bandwidth reservation before use: a router priority mechanism among these classes and external user traffic policing guarantee no oversubscription. The ABR is accomplished using a network to user feedback control loop to assure a fair allocation of resources among all sources that share a congestion link. The BE is given the lowest priority and can take what is left. We discuss these issues for NoCs in the Section 5.4 on switching, and Section 5.7 on end-to-end control.

IP networks have two main QoS standards. IntServ [16] is a per (end-to-end) flow QoS standard and therefore require a rather complex implementation in the Internet. DiffServ [82] defines the differentiated service field with six possible service classes which are similar to the ATM ideas.

Traffic classes and service classes in NoCs

While NoC traffic classes can be mapped to ATM, IntServ, or DiffServ service classes, the direction translation of general computer networks to NoCs is not appropriate, as described in the introduction. Moreover, the specific NoC service classes should be defined based on the vast experience in supporting ASIC and CMP communications over busses and dedicated interconnection infrastructures. Another key factor that separates NoC-based designs from a general network environment is the intrinsic characterization of on-chip communication. In addition, to general message-passing type communication, modules exchange low-level signals and distributed-shared-memory transactions that are typical to SoC flows. Examples are timing and synchronization signals, control words, cache invalidations, and interrupts that require an immediate and time-controlled transfer. Memory transactions such as read/write (RD/WR), code fetch and pre-fetch, semaphore-based operation, and DMA transfer may vary in their timing priorities and urgency. Finally, I/O traffic and off-chip memory access needs to cross chip boundaries via drivers and external pin bottlenecks.

Each designer can identify service classes for a specific NoC implementation. We describe three NoC traffic and service classes in the remainder of this section:

- Goossens et al. [46] characterize different traffic classes in ASIC and ASIP SoCs. The heterogeneity of processing modules results in a variety of traffic classes, based on data rate, latency, and jitter characteristics (see Table 5.1).

  Control traffic originates from control tasks that are usually mapped on one or more processors, and which must obtain status

TABLE 5.1 ■ A classification of traffic classes [46].

Example	Data rate	Latency	Jitter
Control traffic	Low	Low	Low
Cache misses	Medium	Low	Tolerant
Cache pre-fetch	High	Tolerant	Tolerant
Hard real-time video	High	Tolerant	Low
Soft real-time video	High	Tolerant	Tolerant
Audio and MPEG2 bitstreams	Medium	Tolerant	Low
Graphics	Tolerant	Tolerant	Tolerant

information from modules and program them. It has a low data rate, but requires low latency to minimize the system response time, for example, when the application mode changes.

Multi-tasking processors, such as high-performance VLIW processors, do not have sufficient local memory to contain all instructions (code) and data of the multiple tasks. Instruction and data caches are therefore used to automatically swap in and swap out the appropriate instructions and data. This leads to medium (but instantaneously high) data rates, and requires low latency. On the other hand instructions are also speculatively pre-fetched ahead of time resulting in higher traffic that is both latency and jitter tolerant.

Dedicated video-processing modules usually operate on and generate streaming (sequential) traffic with high data rates. They are composed in deep chains without critical feedback loops, and their low-latency requirement can therefore be made less critical by using buffers to avoid underflow. The resulting traffic has a high data rate but is latency tolerant. Medium-data-rate latency-tolerant traffic is generated, for example, by audio and MPEG2-processing modules.

Jitter (latency variation) can be handled similarly, and we use the distinction between low-jitter (hard real time) and jitter-tolerant (soft real time) traffic. Modules with the latter traffic, such as the memory-based video scaler, have an average data-rate requirement but can be stopped when there is no data, and make up by processing at a higher rate later, or by averaging out data bursts. By contrast, low-jitter modules do not tolerate variations in data rates, because they cannot make up for any lost processing cycles. Examples are video-processing modules operating at actual video frequencies, where line and field blanking cannot be used as slack.

- Another rich example of a service classes set was given in Ref. [13] to illustrate a common NoC environment:
  - *Signaling* covers urgent messages and very short packets that are given the highest priority in the network to assure shortest latency. This service level is suitable for interrupts and

control signals and alleviates the need for dedicating special, single-use wires for them. Some of the signals may also take the form of a complete transaction (such as semaphore operations).

- *Real-time* service level guarantees bandwidth and latency to real-time applications, such as streamed audio and video processing or the timely refresh of an LCD screen. While these operations need a guaranteed time for completion, the time limit itself may be quite large compared to other operations.
- *RD/WR* service level provides bus semantics and is hence designed to support short memory and register accesses. This class may be subdivided according to the urgency of the operation (fetching code, capturing resources). Extremely urgent R/W transactions may utilize the signaling service class.
- *Block transfer* service level is used for transfers of long messages and large blocks of data, such as cache refill and DMA transfers.

- Several NoC studies have observed the need to classify traffic and differentiate the service according to pre-specified service classes. The Æthereal [44] and Mango [8] architectures mainly address ASIC and ASSP SoCs for consumer electronics and have separated the services into two distinct classes: guaranteed throughput (GT) and BE. The GT class (similar to the above ATM CBR) accomplishes a TDM like service by limiting the GT traffic to a limited number of periodic flows and prioritizing the GT over the BE. With the right router architecture, this bounds the delay transfer through the network [61].

It should be emphasized that like other NoC design issues, each specific NoC implementation may define its unique set of service classes and also define how to map various traffic classes to service classes. Classifying traffic class may prove to be a challenging issue that involves knowledge of the specific modules' internals. For example, it may turn out that a seemingly single traffic class (memory read operation) needs to be split into several QoS subclasses. For example, fetching code (a memory read operation) to a processor module is sometimes much more urgent (an instruction fetch was missed in a local cache) and sometimes much less urgent (a pre-fetch operation with no current miss) than fetching data (another memory read operation) at the same processor. In such a case the module that originates the memory read instructions may map these similar operations to different classes of service, yet they look identical to an external observer who is not aware of the original cause of these transactions. Connection (identifiers) can be used to indicate to the NoC to which traffic class a transaction belongs [56, 92].

The SoC type impacts the traffic classes that may be expected, and hence the NoC service classes that the NoC must offer. ASICs, ASSPs, and FPGAs implement specific, hard real time, often relatively static applications known in advance [99]. Hence, we can expect service classes that are tailored for assured (guaranteed) real-time data streaming. CMPs, on the other hand, execute a variety of dynamic soft real-time applications that are unknown in advance, and BE service may be the dominant service class.

5.3 NoC TOPOLOGY

NoC architectures and topologies have been described in Chapter 2. In this section, we summarize for convenience the main characteristics of NoC topologies, and we relate them to cost metrics such as area, performance, and power consumption.

Network topology has been intensively studied in the context of high-performance networks [33] and parallel computers architectures [23]. Here, we introduce the concepts germane to NoCs, and refer to [23, 33] for more extensive classifications.

NoCs differ from general networks because they are realized on a plane, even though new technologies, like the emerging die-stacking and three-dimensional integration techniques may change this in the future. Moreover, links between routers can travel only in X or Y direction, in a limited number of planes (the number of metal layers of the IC process). As a result, many NoCs have topologies that can be easily mapped to a plane, such as low-dimensional (1–3) meshes and tori. We list here some important topologies for NoCs, and discuss area cost and performance for each group:

- *Crossbar*: When all routers are connected to all other routers, the NoC is fully connected. The result is single crossbar [67, 68]. It does not scale up to large number of network interfaces.
- *n-dimensional k-ary mesh* (or grid): The two-dimensional 2-ary mesh is a popular NoC topology because routers can be preplaced in layout, and because all links have the same (limited) length. The number of network interfaces per router is usually one, but can also be higher. Examples include QNoC [13] and Nostrum [75]. The area of meshes (the number of routers and network interfaces) grows linearly with the number of cores. Meshes have a relatively large average distance between network interfaces, affecting power dissipation negatively. Moreover, their limited bisection bandwidth reduces performance under high load. Care has to be taken to avoid accumulation of traffic in the center of the mesh (creating a *hot spot*) [83].

- *k-ary n-cube (torus)*: The *k*-ary 1-cube (one-dimensional torus or ring) is the simplest NoC, and is used in Proteo [93]. The area and power dissipation costs (related to the number of routers and network interfaces, and the average distance) of the ring grows linearly with the number of cores. Performance decreases as the size of the NoC grows because the bisection bandwidth is very limited.

  The 3-ary 2-cube (two dimensional) torus adds wrap-around links to the two-dimensional mesh. To reduce the length of these links, the torus can be folded. Dally and Towles [30] use this topology. The area of tori is roughly the same as for meshes (some links are added), but the power dissipation and performance are better because the average distance is less than in meshes.

- *Express cube*: Meshes and tori can be extended with bypass links to increase the performance (bisection bandwidth and reduced average distance), for a higher area cost. The resulting express cubes [25] are essentially used in FPGAs.

- *d-dimensional k-ary (fat) trees*: These have *d* levels, in which each router has *k* children, network interfaces are attached only to the leaves. Examples are SPIN [50, 88]. The bisection bandwidth of tree is very low, due to concentration of all traffic at the root of the tree. To solve this problem the root can be duplicated. The resulting *fat trees* [69] (or folded butterfly) have a large bisection bandwidth (and hence performance), but associated high area cost. For larger number of nodes, the layout of the fat tree is more difficult, in comparison with meshes or tori.

- *Irregular*: NoCs are appropriate when the NoC can be optimized to a particular application domain or set of applications. Synthesizing application-specific NoCs in general at the desired cost–performance point is a challenging problem. Examples of NoCs that allow irregular topologies are Xpipes [5] and Æthereal [44]. Optimized irregular NoCs can also be obtained by removing unnecessary routers and links from a regular NoC. For example, a mesh can optimized to a partial mesh [13], which has sufficient performance for the application at hand, but at lower cost.

NoC types and topologies

There is no strict correspondence between the different types of SoC (application specific, reconfigurable, or general purpose) and their topologies. However, we can discern some general trends.

ASIC and ASSP NoCs tend to be irregular (reduced meshes or completely optimized), because much is known of the application (domain) and NoCs can be highly tailored [99]. The area and power dissipation costs can be reduced, while performance is still guaranteed. For

example, the simple irregular tree topology is already used in commercial products.

FPGAs are composed of small-grained tightly coupled computation and storage units (look-up tables, RAMs, etc.). These units communicate mostly locally, and require high bandwidth and low latency for communication. As a result, express cubes (meshes with bypasses) are mostly used in FPGAs. The NoC is configured infrequently, and has high performance in the steady state.

NoCs for CMPs tend to consist of large-grained loosely coupled (usually homogeneous) computation subsystems (called tiles) [100, 104]. Tiles usually contain a local interconnect such as a bus or switch for frequent local communication, and use the NoC for less frequent global communication. As a result, two-dimensional meshes or tori (with limited bisection bandwidth) are the preferred NoC topology. Because applications are unknown at design time, NoCs are usually not statically configured but schedule traffic at run time.

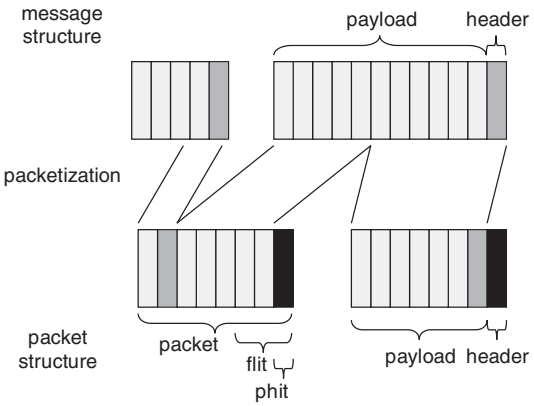
5.4 SWITCHING TECHNIQUES

Once the topology of an NoC has been decided on, the switching technique, or how data flows through the routers, must be determined. This involved defining the granularity of data transfer, and the switching technique.

Data is transferred on a link, which has a fixed width, measured in bits. The unit of data transferred in a single cycle on a link is called the *phit* (physical unit). Two routers synchronize each data transfer, to ensure that buffers do not overflow, for example. Link-level flow control is used for this, and can be based on hand shaking or the use of credits [64]. The unit of synchronization is called a *flit* (flow control unit), and it is at least as large as a phit. Finally, multiple flits constitute a *packet*, several of which may make up *messages* that modules connected to the NoC send to each other. (Note that messages can be used for different NoC programming paradigms, including message passing and distributed-shared memory, cf. Chapter 7.) Fig. 5.2 shows this structure. To increase the packetization efficiency [42] message and packet boundaries do not need to be aligned, as shown.

Different NoCs use different phit, flit, packet, and message sizes. The phit and flit sizes reflect different design choices, such as link speed versus router arbitration speed [89]. For example, Æthereal uses phits of 32 bits, flits of 3 phits (or words), and packets and messages of unbounded length. Nostrum [75] uses phits of 128 bits, and flits equal to 1 phit. SPIN [50] uses phits and flits of 36 bits, and packets can be unbounded in length.

Phits are relevant for the link layer (Chapter 4), and will not be further discussed. The switching technique determines how flits and packets



■ **FIGURE 5.2**

Phits, flits, packets, and messages.

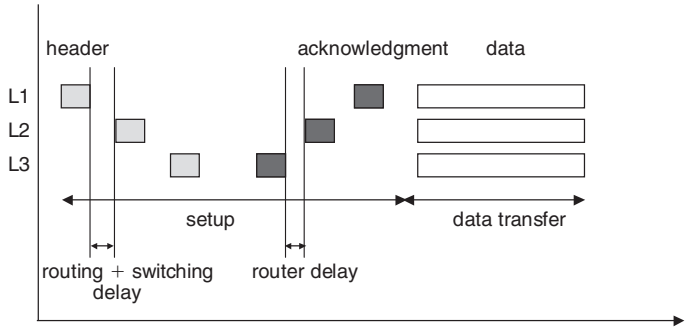
are transported and stored by the routers, as described below. Chapter 7 further discusses messages. Note that the switching technique determines how data flows through the NoC, but not along which route. This is the subject of the next Section 5.5.

There are two basic modes of transporting flits: *circuit switching* and *packet switching*. Essentially, in circuit switching a circuit with a fixed physical path is set up between sender and receiver, and all the flits of the message are sent on this circuit. In contrast, in packet switching the packets constituting a message make their way independently from sender to receiver, perhaps along different routes, and with different delays. We discuss these techniques, and several variations, in more detail below.

To determine a switching technique for an NoC, a number of issues must be balanced, such as the granularity of the data to be sent, and the frequency with which it is sent; the cost and complexity of the router; the dynamism and number of concurrent flows to be supported; and the resulting performance (bandwidth, latency) of the NoC. Different types of NoC (ASIC, ASSP, FPGA, or CMP) often use different switching techniques, as we shall see. The switching technique strongly influences QoS, as mentioned in Section 5.2 and further elaborated in Section 5.7.3. In fact, to offer different QoS levels, NoCs can use multiple switching techniques at the same time [6, 44, 76].

5.4.1 Circuit Switching

Messages from one module to another are sent in their entirety when using circuit switching. First, a physical path, that is, a series of links and routers, from sending to receiving module is determined and reserved for the circuit. Logically, the head flit of the message makes its way from



■ **FIGURE 5.3**

Circuit switching.

the sender to receiver, reserving links along the way. If it arrives at the receiver without conflicts (all links were available) an acknowledgment is sent back to the sender, who commences data transfer on its reception. If a link is reserved by another circuit, a negative acknowledgment is sent to the sender. Fig. 5.3 shows the set-up and use of a circuit over time; R1, R2, and R3 are routers along the path of the circuit. The shaded boxes show when the inter-router wires are occupied.

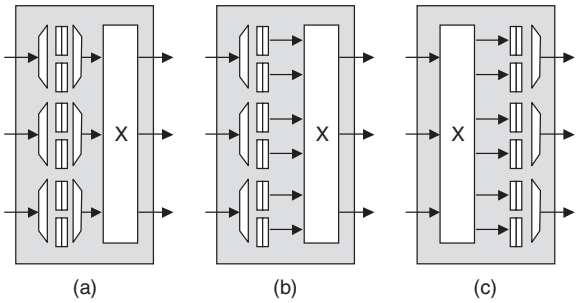
After transmission of the message, the circuit is torn down, as part of the tail flit. Pure circuit switching has not been used much in NoCs, and the principal examples are SOCBus [107, 108].

Circuit switching has a high initial latency due to the set-up phase that has to complete before data transmission starts. (Scouting routing [33] can reduce this time.) Data transmission is very efficient, however, because the full link bandwidth is available to the circuit, and results in minimal latency. Data does not have to be buffered in the routers (only pipelined perhaps), reducing the area of routers. However, circuit switching does not scale well as the diameter of the NoC grows [31, 107] because links are occupied also when data is not being transmitted (during the set-up and tear-down phases).

Circuit switching is appropriate when data is sent very often, or when the communication pattern between senders and receivers is relatively static. The circuit can be left in place in these cases. When the amount of data to be transmitted is large (making the set-up phase less relevant) circuit switching also works well. ASICs and ASSPs have relatively static communication patterns, and FPGAs also send data (bits or words) every cycle on the circuit. FPGAs (currently) exclusively use circuit switching, and ASIC and ASSP NoCs often do [44, 99, 108].

Virtual channels and virtual circuits

Circuit switching reserves physical links between routers. Multiple virtual links (more commonly called *virtual channels* [29]) can be multiplexed on a



■ **FIGURE 5.4**

Virtual-circuit switching with multiple (2) virtual channels.

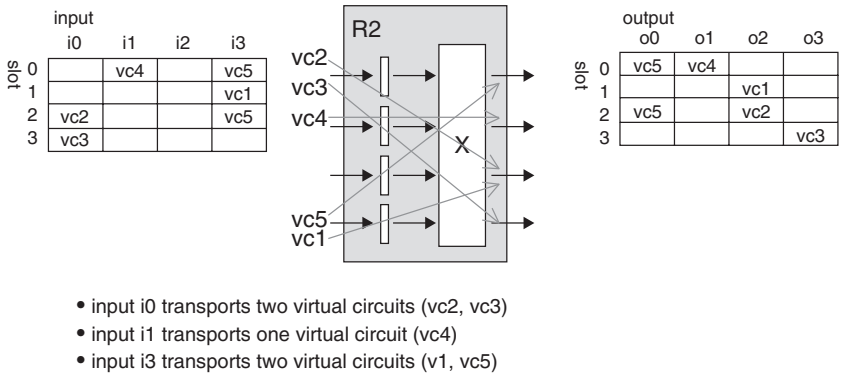
single physical link (channel). Virtual channels can be used to make circuit switching more flexible (see below), to increase performance by reducing blocking of links (described in Section 5.4.2), and to avoid deadlock (see Section 5.5).

Circuit switching reserves physical links between routers. *Virtual circuits* can be created by multiplexing circuits on links. The number of virtual channels that can be supported by a link depends on the number of buffers on the link. Two basic schemes have been used: a buffer per virtual circuit, or one buffer per link. Intermediate strategies are possible, but are more complex. We describe each scheme in turn, after which we discuss alternatives for circuit set-up and tear-down.

Virtual circuits with multiple buffers per link

A virtual circuit requires a buffer in each router it passes through. The spatial distribution of virtual circuits in the NoC therefore determines how many buffers are required in each router. This requirement can be used to determine the number of buffers for each router (virtual-circuit buffering). The number of buffers can then vary per router. Alternatively, the number of buffers per router can be given as constant and taken equal to the number of virtual channels (virtual-channel buffering). In this case, the number of virtual circuits using a given link is limited by the number of virtual channels on that link (and the routing algorithm, to avoid links of which all buffers are occupied). In both cases, virtual circuits are created by using virtual links, and it is only the determination of the number of the buffers in the routers that makes the difference. Fig. 5.4(a) shows the outline of a router with multiple buffers at the inputs of the switch. Fig. 5.4(b) shows how increasing the size of the switch from $N \times N$ (where N is the router degree) to $(V \times N) \times N$ (where V is the number of virtual channels) reduce the contention on the switch [44, 63]. Fig. 5.4(c) shows virtual-channel buffering with output queueing [8].

In any case, many systems contain hot spots, where many virtual-circuits converge, such as the external memory interface [46]. There will



■ **FIGURE 5.5**

Virtual circuit switching with a single buffer per link and TDM.

not be enough virtual channels for accommodate all flows. Virtual-circuit buffered router implementations become problematic either due to multiplexing the large number of small buffers, or due to the expensive (large) shared buffer implementations, such as SRAMs. Routers that implement virtual circuits with a single buffer per link obviously do not suffer from this problem, as described below.

The multiplexing of the individual virtual channels on a single link requires scheduling at each link/router, and results in an end-to-end schedule of the virtual circuits. Virtual circuits with input queuing are used by Refs [63, 108]. The scheduling of access to links and access to the (blocking) crossbar in the router interfere, making bandwidth and latency guarantees difficult to achieve. For this reason, the Mango NoC uses virtual-channel buffering implemented by output queuing with a non-blocking crossbar [8] (see Fig. 5.4(c)). This, together with an appropriate link scheduling scheme [9] enables it to guarantee bandwidth and latency on its virtual circuits.

Virtual circuits with a single buffer per link

Circuits can also be time multiplexed with one buffer per link, that is, without requiring buffering per virtual circuit. Essentially this is achieved by statically scheduling (using TDM) the usage of all links in the NoC by all virtual circuits. Fig. 5.5 shows a 4×4 router with four TDM slots. Each input has a one-flit buffer even though it can be used by up to four virtual circuits. The TDM tables show how virtual circuits are mapped from inputs to output in time. The example is described in more detail in Section 5.7.1 and is consistent with Fig. 5.20.

At the edge of the NoC, flits are injected by the network interfaces such that they never use the same link at the same time (see Fig. 5.20). As a result, link-level flow control and scheduling can be omitted, and only

one-flit buffer per link is necessary. This scheme assumes the propagation speed of individual flits through the NoC is fixed and known in advance. Flits wait in the network interfaces until they are injected in the NoC according to the TDM schedule. Note that network interfaces require some kind of end-to-end flow control for lossless operation, probably with virtual-circuit buffering (Section 5.7.2).

The TDM is used to globally schedule all flows, and hence end-to-end bandwidth and latency guarantees are easily given ([37]; Section 5.7.3).

The concept of globally scheduling all virtual circuits is used by NuMesh [96] (for parallel processing), Nostrum's looped containers [75], adaptive SoCs (aSoCs) global scheduling [65, 66], and Æthereal's contention-free routing [44, 90].

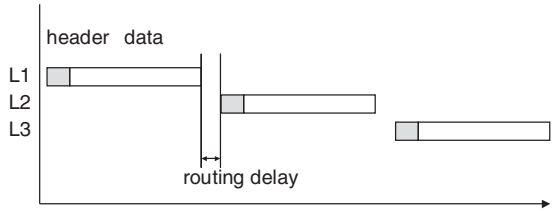
Circuit management

Set up and tear down of virtual circuits can take place as described at the start of this section. Options include static or dynamic reservations, backtracking, and multicast. Care must be taken that deadlock does not occur during set up. This can be achieved by retracting a circuit [44, 107], or by dropping data [31]. Alternatively, a programming can be centralized in a single entity such as a programmable processor [44, 63, 108].

Set-up, (negative) acknowledgment, and tear-down can also be encoded as messages (usually just of one flit), like in the ATM [3] and pipelined circuit switching [33]. Set-up and tear-down messages program routers, using a message-based or memory-mapped protocol. Æthereal [44] and Mango [6] are examples of this approach. Control messages can use a different switching scheme (e.g., wormhole (WH) packet switching) and different QoS class (e.g., BE). Using messages allows circuit management to use the NoC itself, eliminating a separate control interconnect to configure the NoC [49, 108].

5.4.2 Packet Switching

In (virtual) circuit switching a complete (shared) path is reserved before data is sent. In contrast, in packet switching, no link reservations are made, and the packets constituting a message make their way independently from sender to receiver, perhaps along different routes, and with different delays. Omitting the set-up phase removes the start-up time (set-up until acknowledgment), but without link reservations, packets of different flows may attempt to use a link at the same time. This is called *contention*, and requires that all but one packet must wait until the link becomes available again. The start-up waiting time of circuit switching (followed by a fixed minimal latency in the routers) is replaced by a zero start-up time and a variable delay due to contention in the each router along the packet's path. For this reason, QoS guarantees are harder to deliver in packet-switched NoCs than in circuit-switched NoCs.



■ **FIGURE 5.6**

Store-and-forward switching.

The lack of resource reservations means that there is no limit to the number of flows (or connections) that a link or the NoC as a whole can support, in contrast to (virtual) circuit switching.

In packet switching, packets of different flows are automatically distributed over different links through the network (if dynamic routing is used), and interleaved on links. As a result, links can be dimensioned for (at least) the average amount of traffic. Virtual-channel or virtual-circuit buffering also interleave packets of different virtual circuits on a link, but all packets of a virtual circuit always follow the same static path, which must be dimensioned for (at least) the sum of the average traffic on the virtual circuits. Instead, pure circuit switching requires that each link supports the worst-case requirements of each circuit using it.

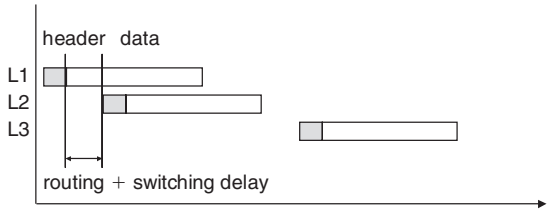
There are three basic packet-switching schemes: store and forward (SAF), virtual cut through (VCT), and WH switching. They are discussed next.

SAF switching

SAF switching is the simplest form of packet switching. A packet is sent from one router to the next only when the receiving router has buffer space for the entire packet (Fig. 5.6). Hence, packet transmission cannot stall and the notion of flit is not required (flits are equal to packets). Routers forward a packet only when it has been received in its entirety. As a result, the latency per router and the buffer size are at least equal to the size of the packet. Given that minimizing buffer size is critical in NoCs [89], few NoCs have used this basic technique. In particular, Nostrum [75] uses hot-potato routing [35] where the notions of phit, flit, and packet are conflated (to their link width of 128 bits).

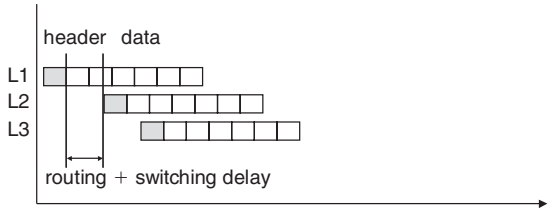
VCT switching

VCT switching reduces the per router latency by forwarding the first flit of a packet as soon as space for the entire packet is available in the next router (middle transmission in Fig. 5.7). The other flits then follow without delay. If no space is available, the whole packet is buffered (last transmission in



■ **FIGURE 5.7**

VCT switching.



■ **FIGURE 5.8**

Wormhole switching.

Fig. 5.7). Buffering requirements of SAF switching and VCT switching are the same, and no NoC has used this basic technique, therefore.

WH switching

WH switching improves on VCT switching by reducing the buffer requirements to one flit. This is achieved by forwarding each flit of a packet when there is space for that flit in the receiving router (Fig. 5.8). If no space is available in the next router for the entire packet (as required by VCT switching but not by WH switching), the packet is left strung out over two or more routers. This blocks the link, which results in higher congestion than with SAF and VCT switching. Link blocking can be alleviated by multiplexing virtual links (or virtual channels) on one physical link (cf. Section 5.4.1) [26]. WH switching is also more susceptible to deadlock than SAF and VCT switching due to the newly introduced usage dependencies between links. Virtual channels and/or routing schemes can be used to avoid deadlock (see Section 5.6.2).

Almost all NoCs use WH switching without virtual channels [13, 60] use restricted topologies (usually partial mesh, with some form of dimension-ordered routing) to avoid deadlock. SPIN [1] uses a fat tree topology with deflection routing and packet reordering at the receiving network interface, which also avoids routing deadlock. Æthereal's [89] BE service class uses WH switching class and avoids deadlock for any topology through a

combination of turn-prohibition routing and end-to-end flow control [55]. Other approaches [8, 63], use WH switching with virtual channels.

5.4.3 Combinations of Different Switching Techniques

Circuit switching and packet switching have different characteristics that may be useful to combine in a single NoC. With circuit switching it is relatively easy to guarantee bandwidth and latency guarantees for a given set of (virtual) circuits (see Section 5.4.1). Packet switching, on the other hand, has no notion of (virtual) circuits and can therefore support any number of concurrent flows. However, it is hard to give hard bandwidth and latency guarantees.

Virtual circuit switching and packet switching can be combined by allocating (at least one) virtual channel to each class. Guaranteed services are mapped to virtual circuits and BE services to packets. Æthereal [89], Nostrum [75], and Mango [8] use this approach. The first two use TDM and Mango uses "asynchronous latency guarantee" scheduling. Note that although WH switching with very long (infinite) packet lengths also creates virtual circuits [49, 63], but to offer end-to-end performance guarantees the right scheduling discipline is essential.

Basic packet switching is appropriate for very dynamic applications where flows are short lived, change frequently, or very variable in their demands, due to the absence of flow reservations, and set-up and tear-down phases. Relatively short messages (synchronization traffic, cache lines, etc.) to many different modules (e.g., distributed memories and caches) makes connection-oriented QoS inappropriate. Connection-less packet switching with BE service class is therefore natural for CMP NoCs. However, to offer assured (guaranteed) service classes, connection-oriented packet switching, based on (TDM) virtual circuits, has been used successfully for ASICs and ASSPs [99].

5.5 NoC ADDRESSING AND ROUTING

While the *switching technique*, as discussed in Section 5.4, controls how data is buffered and transported between routers, the routing layer determines the paths over which data follows through the network. Before we do this we briefly describe how the start and end points of the route are indicated through various addressing schemes.

5.6 NoC ADDRESSING

In order to route information within the chip unique identifications or addresses must be assigned to each reachable destination. It is important

to notice that such destinations may have a hierarchical relationships or layers. For example, a certain modules (e.g., subsystems or tiles) in the chip may have multiple submodules, such as processors and local shared memories. Each processor may execute multiple programs and each program may be composed of multiple tasks and threads. Consequently such an identity can be described as thread 3 of program 7 of processor 1 at module 6.

To hide implementation details logical addresses are often used instead of physical addresses. A single physical address, such as a memory (location) may be known as different logical addresses or vice versa. The former is useful when processors with different memory maps (e.g., 24 or 32 bit addresses, or with different layout) share a single memory to communicate. The latter can be used, for example, for security or virtualization purposes, or to allow different processors to boot from the same fixed address, but with different boot code. The mapping (or translation or resolution) of logical to physical address may be done in software or hardware, distributed or centralized, fixed at design time or configurable at run time [91].

The physical address space in each NoC implementation may be assigned according to the number of different modules (e.g., we use 4 bits for an NoC that has less than 16 modules), the relative location of the module in the NoC (e.g., XY coordinate in a grid or a node name in a fat tree). Logical addresses may relate to functional names (e.g., the external memory interface, a coprocessor unit) or to a global address space (e.g., 16, 24, or 32 bits).

Finally, different flows may belong to different service classes, introduced in the previous Section 5.2. Identifiers may be needed to distinguish flows for QoS purposes, therefore. (Because different flows between the same addresses may below to different flows, separating control and data, for example.) These include identifiers for individual transactions (for reordering), for communication threads (as used e.g., in AXI [2]), and flows/connections [92].

Addressing is discussed in more detail in Chapter 6 on network interfaces (e.g., address translations) and Chapter 7 on NoC programming models (e.g., message passing versus shared memory).

5.6.1 NoC Routing

In the following we focus on routing data in NoCs and particularly emphasize the planar mesh topology, which is popular for NoCs. The NoC routing mechanism is responsible for correct and efficient routing of packets (or circuits) that are traversing the network from sources to destinations. The routing protocol deals with resolution of the routing decision made at every router. Unlike traditional communication or interconnection networks, NoCs need not follow rigid networking standards, therefore, a

multiple routing schemes can be evaluated and compared for each NoC implementation. There are several potentially conflicting metrics that need to be balanced:

- *Power*: minimize the power required to route packets. This means that packets may follow the minimal power path likely to be identical to the traditional shortest distance routing [13, 87]. In certain cases, for example, when DVS is applied in a non-uniform way, each router and link may offer a different packet switching power consumption [95].
- *Area and VLSI resources*: the routing mechanism itself consumes hardware resources like finite state machines, and addresses tables. It potentially also uses NoC bandwidth if routers exchange information.
- *Performance*: reduce the delay and maximize the traffic utilization of the network.
- *Robustness to traffic changes*: certain routing schemes (e.g., static routing schemes) may perform very well to an expected traffic pattern but poorly to changing traffic patterns. Other schemes (e.g., dynamic routing) may behave better to a larger spectrum of traffic conditions.

Routing schemes can be classified into several different categories. In particular routing can be static or dynamic, distributed or performed at the source, and minimal or non-minimal. We describe each in turn.

Static and dynamic routing

The routing decision at every router can be *static* (also called oblivious or deterministic) or *dynamic* (or adaptive).

In a static routing scheme permanent paths from a given source to a given destination are defined and are used regardless of the current state of the network. This routing scheme does not take into account the current load of network links and routers when making routing decisions. Note that static routing may use single path or split the traffic in a predefined way among multiple paths between a source and a destination.

In a dynamic routing scheme, routing decisions are made according to the current state of the network (load, available links). Consequently, the traffic between a source destination changes its route with time.

Static routing is simpler to implement in terms of router logic and interaction between routers. A major advantage of a single path static routing is that all packets with the same source and destination are routed over the same path and can be kept in order. In this case, there is no need to number and reorder the packets at the network interface. Static routing is clearly more appropriate when traffic requirements are steady and known

ahead of time, and therefore is preferable for NoCs for ASICs or ASSPs. Dynamic or adaptive routing may utilize alternative paths when certain directions become congested and therefore have the potential of supporting more traffic using the same network topology. Consequently it may be preferable in irregular and unpredictable traffic conditions, which are more common to the CMP NoCs.

Distributed and source routing

Routing techniques (both static and dynamic) can be further classified according to where the routing information is held and where routing decisions are made.

In *distributed* routing, each packet carries the destination address, for example the XY coordinates of the destination router or network interface, or a module number. The routing decision is implemented in each router either by looking up the destination addresses in a routing table or by executing a hardware routing function [14]. Using this method, each network router contains a predefined routing function whose input is the destination address of the packet and its output is the routing decision. When the packet arrives at the input port of the router, its output port is looked up in the table or calculated by the routing logic according to the destination address carried by the packet. Note that in order to reduce routing table space a specific distributed routing technique may restrict its supported network topology, the type of routes that can be defined and the naming convention of network destination. A very common example in NoCs is the XY routing (also termed dimension routing [24]) for mesh networks where destinations are named after their geographical (XY) coordinates and the intermediate routing function is limited to the comparison between the router address and the destination address. Interval routing [11] suggests a similar reduced table space methodology for general topology networks.

In *source routing* the pre-computed routing tables are stored in the network interface at the system modules. When a source router (network interface) transmits a packet, it looks up the source routing information according to the destination address at the source routing table and includes it in the header of the packet. Each packet carries in its header the routing choices for each hop along its path (typically the output link identity). When the packet arrives at a network router, its next routing output port is extracted from the header routing field. The routing field is then shifted in order to expose the relevant routing choice for the next router on its path. In comparison to distributed routing, source routing does not need any intermediate routing tables or functions, it may also eliminate the destination address field required in distributed. On the other hand, it requires a source route header in the packet header (whose size increases with the path length) and requires additional routing tables with specific entries for each source (these can be located in the source network interface or in a centralized pool). Æthereal uses source routing [44].

While distributed routing and source routing describe the way packets and router interact to perform intermediate routing decisions, both schemes leave a full degree of freedom in deciding on the selection of the (static) packet routes. Most distributed routing designs make a conscious limitation on the selected routes. In order to reduce the amount of logic required, routes are based only on the destination address rather than on the source and destination address pair.

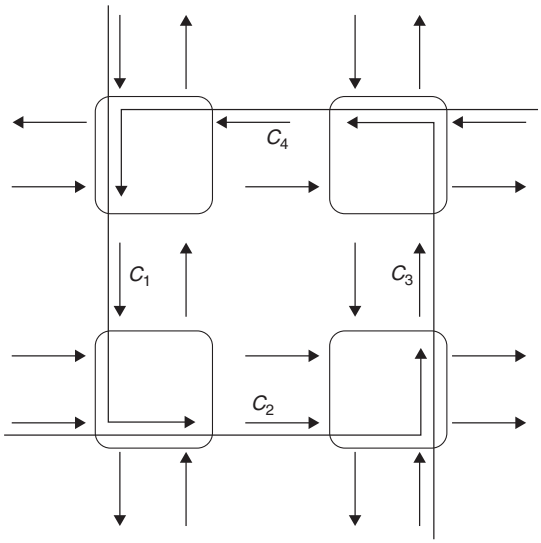
Minimal and non-minimal routing

A final classification criterion distinguishes between *minimal* and *non-minimal* distance routing. In contrast to traditional interconnection networks, the additional power consumption introduced by non-minimal routes may be prohibitively expensive in an NoC [57]. Note that in heterogeneous NoCs where links between routers are of different speed and length, minimal power routing is not equivalent to a minimal link hop routing [12].

5.6.2 Deadlock

The major constraint for any routing algorithm is assuring the freedom from *deadlock*. In packet switching networks whenever a packet or flit is transferred between neighboring routers, it releases a buffer at the transmitter and occupies a previously free buffer at the receiver. Consequently, such a transfer requires the availability of a free buffer at the receiver, or the flit is held (in a lossless network), due to link-level flow control, until such a buffer is freed at the receiver. Deadlock occurs when one or more packets in the network become blocked and stay blocked for an indefinite time, waiting for an event that cannot happen. A typical example, as depicted in Fig. 5.9 [81], is a situation where four packets are routed in a circular manner between the routers in a square mesh. The packet occupying channel c_1 is waiting for c_2 and that channel is allocated to a packet that wants to use c_3. That channel in turn is held by a packet that is requesting c_4 and the packet on this channel completes the circle by waiting for c_1. No packet can advance since the required resource, in this case the channel, is already held by another packet and will never be released.

While WH switching is the prevalent switching technique for NoCs (due to its relatively small buffering requirements), it is prone to suffer from deadlocks. Since only the header flit carries routing information all flits belonging to the same packet must be contiguous and cannot be interleaved with flits belonging to other packets [29]. If a flit is blocked due to busy resources all the trailing flits of that packet will also be stopped and keep blocking the resources they occupy in terms of channels and buffers. This can result in chained blocking [26] where the resources of a blocked packet again causes other packets to block, a property that makes WH routing very susceptible to deadlock [29, 36].



■ **FIGURE 5.9**

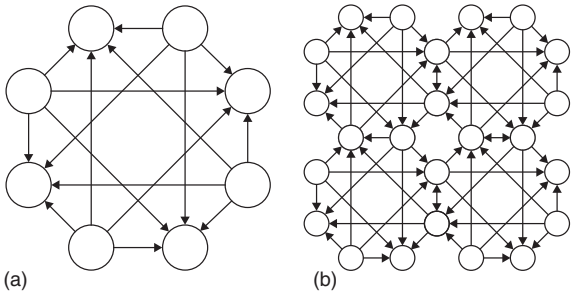
Deadlocked scenario with packets waiting for channels in a cyclic manner.

Deadlock avoidance

The prominent strategy for dealing with deadlock is avoidance and most deadlock-free routing algorithms are deduced by the strategy of [29, 36]: (1) choose a particular routing algorithm, (2) check whether this algorithm is deadlock free, and (3) if needed, add hardware resources or restrict routing to make the algorithm deadlock free.

Deadlock freedom is analyzed by building a dependency graph of the shared network resources. Whenever a packet is holding a resource while requesting another there is a *dependency* between them and if this dependency graph is cyclic then we have a circular wait. Analysis of the graph can be done statically, with all possible dependencies represented in the graph, or dynamically, with the graph reflecting the current state of the system. Static analysis is unnecessarily pessimistic since dependencies that are mutually exclusive can be part of a cycle that never occurs at run time. However, the big advantage with static analysis is that it can be done beforehand and thus detach this computationally intensive problem from the actual system.[1] Thus, deadlock is usually avoided by employing a restrictive routing algorithm.

[1] Much research has been focused on distributed cycle detection mechanisms that allow for progressive deadlock recovery based on only local information while keeping recovery overhead to a minimum [19, 20, 72].



■ **FIGURE 5.10**

Channel dependency graph of a mesh in which the routers use edge buffers. (a) For one router and (b) for the entire mesh.

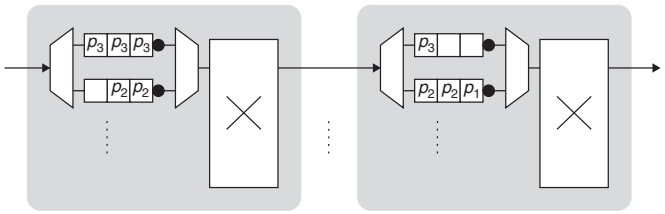
Deadlock freedom in SAF and VCT networks, which are identical from a deadlock perspective [29], is proved using a buffer-ordering technique of Ref. [51]. In a WH-switched network blocked messages remain in the network and hence continue to occupy the links until contention is resolved. Therefore, we cannot use restrictions in buffer allocation to prevent deadlock, but must instead restrict routing over the communication channels, be they physical or virtual.

The dependencies of a WH-switched network are captured in a *channel dependency graph* [29]. Fig. 5.10 illustrates the channel dependencies of a minuscule network. With a fully dynamic routing function every input channel has the possibility of forwarding a packet to any output channel and the channel dependency graph of a single router thus looks as shown in Fig. 5.10(a). The graph is obviously acyclic since all the input and output ports of the router are dangling. In Fig. 5.10(b) we see the corresponding graph for a simple two-by-two mesh and this figure contains numerous cycles. There is thus a risk of channel deadlock in the network.

Dally et al. [29] propose a necessary and sufficient condition for deadlock freedom in the case of a static routing function. This proof technique is extended in Refs [27, 70] to also cover dynamic routing.

Introducing virtual channels increases the degrees of freedom in alleviating restrictions in the choice of channels (see Section 5.4.1) [40] and can aid both in avoiding deadlock as well as increasing network throughput by reducing the effects of chained blocking [26]. By dissociating the buffers (associated with channels) from the actual physical channels, a blocked packet in one virtual channel does not preclude packets residing on other virtual channels [77] as depicted in Fig. 5.11.

The problem of deadlocks in a WH NoC can be alleviated by the use of multiple virtual channels. The idea is that more buffers can be allocated in the receiver size to different flows (e.g., a Virtual Channel per destination). However, the number of Virtual Channels that are required



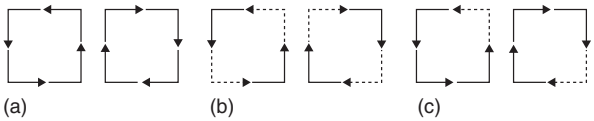
■ **FIGURE 5.11**

A packet p_2 is blocked by a previous packet p_1 and is occupying the physical channel between the routers. Packet p_3 that follows p_2 is allowed to proceed and pass p_2 by using the buffers associated with another virtual channel.

to completely solve the deadlock problem for any routing scheme, is large [28, 33] and therefore costly in terms of VLSI resources. There are general mechanisms for avoiding deadlocks by restricting the routing algorithms. Glass and Ni [40] call a specific pair of input–output links around a router as a *turn* and introduce several deadlock-free routing schemes based on *turn prohibition*. Enough turns are prohibited to break all dependency cycles while still maintaining connectivity between every pair of routers. The Æthereal BE service class uses the turn model for routing on arbitrary topologies [54]. The most simplistic approach to turn prohibition is the well-known dimension-ordered routing. It is a static and minimal distance routing algorithm where a packet is routed in one dimension at a time, finding the correct coordinate in each dimension before continuing with the next. In a two-dimensional mesh this method is known as XY routing and is very popular due to its simplicity. When a packet is sent it will be forwarded in the X dimension until it reaches the X coordinate of the destination; only then is it forwarded in the Y dimension until it reaches its goal.

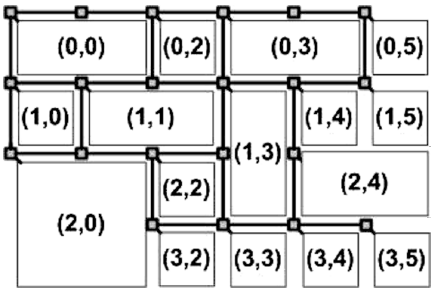
An illustration of the turn model in a two-dimensional mesh is shown in Fig. 5.12. There are eight possible turns that form two cycles if no turns are prohibited. With the static XY routing four of the turns are prohibited and it is clear that no cycles can be formed from the remaining turns. It is not necessary to prohibit this many turns to guarantee deadlock freedom. The *west-first* routing algorithm [41], illustrated in Fig. 5.12(c), prohibits only two of the eight turns and manages to break all cycles thereby constituting a deadlock-free routing algorithm.

Many contemporary NoC proposals based on a regular mesh topology have implemented the above dimension-order (XY) based routing. It is shown in Refs [13, 52] that this may cause a significant imbalance in the traffic utilization of the mesh links, even when traffic requirements are symmetric. When traffic patterns are known ahead of time (for ASICs), this can be dealt with using NoCs with asymmetric link capacities such as QNoC [13, 52], where each link capacity is planned according to its expected load. This eliminates the need for an expensive implementation



■ **FIGURE 5.12**

Illustration of turn prohibition in two-dimensional mesh. (a) Unprohibited mesh with two cycles, (b) four prohibited turns of XY routing, and (c) two prohibited turns of west-first routing.



■ **FIGURE 5.13**

SoC modules interconnected by an irregular mesh NoC.

of an dynamic load balance routing that requires the reordering of packets at the network interfaces. A simple alternative to link capacity planning is to use a combination of alternatively (or randomly) choosing between XY and YX routing as described in Refs [59, 94] that is close to optimal in terms of maximum utilization in a homogenous mesh. Therefore, the "toggle" XY–YX routing above better suits a FPGA type device where the links between the routers cannot be customized ahead of time for specific loads [39]. While other balanced static routing schemes (also termed oblivious routing [71]) were introduced for mesh-based interconnection networks (and can be implemented using source routing), they usually require too much hardware to be used in a low-budget NoC environment.

Another problem which is related to low-cost static routing is how to cope with irregular meshed NoCs that result from module size variability and the need to physically separate between the modules internals and the NoC infrastructure, as illustrated in Fig. 5.13.

Packet routing in irregular meshes resembles routing in a labyrinth, since missing links may lead to a dead end. Therefore, a simple XY or XY–YX "toggle" scheme cannot be employed and different routing techniques need to be applied. In off-chip networks, routing in irregular topologies is typically accomplished using routing tables located in routers or in sources. Routing table size and the corresponding power and area costs grow with the network size. Moreover, the time required to access

each table, which affects NoC performance, depends on its size and thus on the network size.

Reference [14] develops hardware-efficient routing techniques that reduce the VLSI cost of routing in irregular mesh topology NoCs. The techniques are based on a combination of a fixed routing function (such as "route XY" or "don't turn") and reduced routing tables for both distributed and source routing approaches. The entries in the reduced routing tables are created only for destinations whose routing decisions differ from the output of the default routing function. Random simulations of different topologies and flow scenarios are used for comparing and estimating the VLSI cost savings obtained by different algorithms. This mechanism is found to be superior (around five times lower cost) to the traditional source routing and routing table-based techniques.

5.6.3 NoC Dynamic Routing Schemes[2]

Dynamic routing is an efficient alternative to balance the traffic load over a given NoC where the NoC traffic is unpredictable or changes with time (e.g., the CMP NoC type). Note that when a source destination traffic is split over multiple paths, packets may arrive out of order and re-sequencing buffers at the destination may be required. The simplest method of dynamic routing is termed *deflection routing* or *hot-potato* routing. It was suggested for metropolitan networks [73], optical burst networks [15], and interconnection networks [35]. In this scheme, when a packet enters a router it will be sent toward a preferred output port according to a routing table or a routing function as described above. However, if the preferred port is busy (blocked by a backpressure from a neighboring router, or captured by another packet) an alternative port will be selected. Here the router has no additional buffers in which to store the packets before they are moved, and each packet is constantly transferred until it reaches its final destination. The packet is bounced around like a "hot potato," sometimes moving further away from its destination because it has to keep moving through the network. This is in contrast to SAF switching where the network allows temporary storage at intermediate locations (Section 5.4.2). Deadlocks cannot happen in deflection routing when the number of input and output ports of a switch are identical and new local packets are not allowed in when all inputs are busy. It is guaranteed that any packet in a router will be transferred in the next cycle to any of the output ports and therefore the router can receive a new packet over all its inputs from neighboring routers (so no backpressure is sent among routers). *Livelocks* may happen in deflection routing and needs to be resolved. A livelock situation happens when a packet is sent

[2] We acknowledge contributions from Andreas Hansson [53, 54] for this section.

over and over and never reaches its final destination. Simple priority rules can resolve it [17].

While there is a broad literature on the use of deflection routing in various networks, one may question the impact of this scheme on the NoC power consumption due to the long routes packets may take in the network. However, the benefits of an dynamic routing scheme for the CMP model may overcome the disadvantages. First, deflection routing automatically spreads traffic to alternate routes when primary routes are in demand. Second, routes can be selected according to profitability [109] where routes that move packets closer to the destination are favorable over paths that lead packets away from it. Finally, in the NoC environment, the backpressure from neighboring routers may be more effective than in off-chip networks due to the relative proximity of routers to each other. Nilsson et al. [83] suggest avoiding excessive oscillations by exchanging "stress values" among neighboring routers. Packets are routed away from "stressed areas." Ye et al. [109] leverage the previous idea and suggests a contention-look-ahead routing based on flits in a WH network.

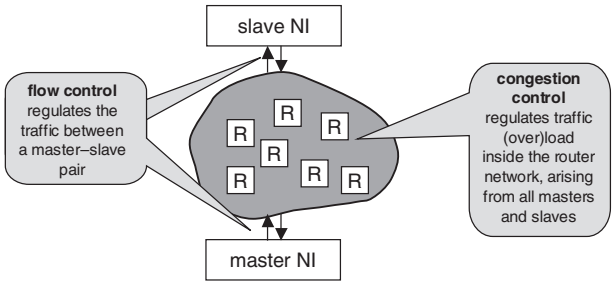
Several more dynamic routing technologies have been suggested to NoCs. An interesting technique that switches between the XY and the YX in an dynamic way is described in Ref. [59]. Gossip-based routing which is based on a broadcast of the information to all destinations is suggested in Refs [17, 34].

SPIN [49] uses dynamic routing in a fat tree topology. In routing from one module to another, any path to a common ancestor in the tree may be taken. A unique path then exists from that ancestor to the destination module. Although all paths between a pair of modules have the same length, delays may be different on different paths due to congestion and re-sequencing buffers are necessary at the receiver network interface.

An interesting balance between static and dynamic routing may be very suitable for the FPGA or CMP NoCs. In some NoCs paths can be configured at power up [39] or reconfigured at run time [7, 84, 91]. The latter usually happens based on a change in the application, for example, starting a new functionality [79]. It could, however, also be applied to deal with variation within a single application. The paths that are newly loaded can either be computed at run time or precomputed at design time. In this way, traffic can be distributed over the NoC in a way that is tailored to the application (or mode) at hand. ASICs, ASSPs, and FPGAs often have relatively static application modes (Fig. 5.1), and NoCs for these systems will benefit from this approach.

5.7 CONGESTION CONTROL AND FLOW CONTROL

In previous sections we have shown how data is transported through the network (Section 5.4), along which routes (Section 5.5), to offer the



■ **FIGURE 5.14**

Scope of congestion control and flow control.

required QoS properties (Section 5.2). In this section we describe two fundamental phenomena that have to be addressed in order to deliver the various QoS levels.

In particular, packets travelling through the router network may want to use the same resources (buffers, links) at the same time. This is called *contention*. Like an oil stain, contention can propagate, leading to *congestion* and reduced network performance. In Section 5.7.1, we illustrate the problem in more detail, and describe a number of *congestion control* techniques.

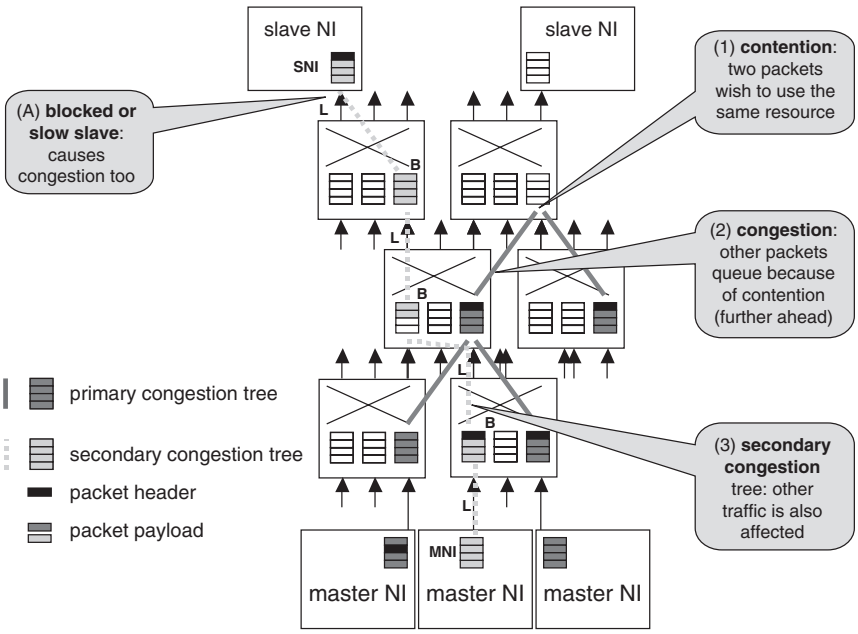
A separate but related problem arises when two communicating parties (a master and a slave) are not balanced in terms of data injection and consumption rates. A slow (or non-responding) slave can cause a backlog of packets inside the router network. In Section 5.7.2 we describe a number of *flow control* techniques that address this problem.

Fig. 5.14 shows the scope of congestion control and flow control. In a nutshell, congestion control keeps the router network free of traffic jams, while flow control ensures that no sender overwhelms any of its receivers.

In Section 5.7.3 we show that to offer desirable communication services (QoS) such as guaranteed minimum bandwidth, guaranteed maximum latency and jitter, both congestion control and flow control are required.

5.7.1 Congestion Control

Fig. 5.15 illustrates the basic problem. To send a message from a producer to a consumer, it must be accepted by a buffer in the master network interface (master NI in the figure). Then the packets constituting the message traverse links and routers (L and B). At some point in the future the message has to be accepted by a buffer in the slave network interface (Slave NI in the figure). Therefore, a series of resources (buffers, links) is used at different points in time. Managing resources spatially (which links and/or buffers) and temporally (at which points in time) can prevent overloading the resources individually (contention) as well as collectively (congestion).



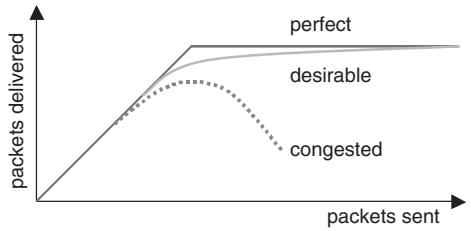
■ **FIGURE 5.15**

Contention and congestion.

Fig. 5.15 shows that congestion is the result of contention: when multiple packets wish to use the same link, the waiting packet (on the left) causes queues behind it (comment (1) in the figure). The contention is caused inside the network by two packets contending for a single output link or buffer. This occurs, for example, in the center of a mesh when dimension-ordered routing is used. A congestion tree results, which affects not only packets destined for the shared link or slave (2, solid lines, and the dark-shaded packets) but also packets that only share a link in the congestion tree (3, dashed lines, and the light-shaded packets). Congestion lowers the effective utilization of the network (because packets are waiting or deflected and take a longer route), and thus congestion should be avoided for performance reasons. In addition, congestion also makes end-to-end bandwidth and latency harder to compute (because dynamic interactions between shared resources must be modeled), and must be circumscribed if performance guarantees are to be given.

We describe several techniques to limit congestion, starting with reactive techniques that do not require explicit resource reservations. Then we detail techniques that use resource reservations that reduce or even eliminate contention entirely.

Congestion control methods are also classified as being closed loop (based on feedback) or open loop (preventive) in the computer network



■ **FIGURE 5.16**

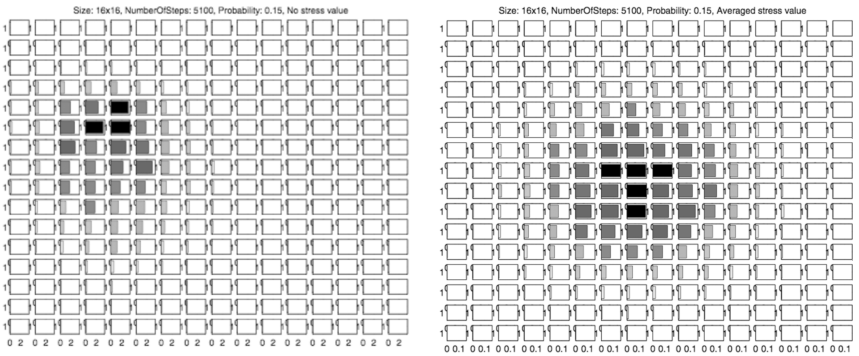
Dropping packets reduces the effective utilization of a network [101].

domain [101]. Many of the methods that exist for computer networks can be reused on chip (such as traffic shaping, token buckets, and leaky buckets). However, because NoCs have until now, not dropped packets many methods are also not applicable (such as load shedding).

Congestion control without resource reservations

- A localized solution to contention is deleting (dropping) packets. When two or more packets wish to simultaneously use the same resource, all but one are deleted. In the short term, this solves the contention and also avoids any congestion. However, on a larger time scale, dropped packets must be re-sent. Dropping therefore leads to *more* traffic in the long run, reducing the effective utilization of the NoC, as shown in Fig. 5.16. Until now, no NoC has used packet dropping for congestion control.
- Alternatively, dynamic routing schemes (cf. Section 5.6.3) can be used to send packets around the contention. SPIN [50] is an example of this approach, and uses dynamic routing with WH switching. Nostrum [75] uses another approach, namely deflection routing with SAF switching. If multiple packets should be routed to the same output port, then one is picked and the remaining packets are routed to other outputs. By always routing all incoming packets to non-conflicting outputs, only one packet needs to be stored for each input. To reduce the SAF latency, short packets are preferred, at the cost of wider inter-router links.

  In all dynamic routing schemes, packets can arrive out of order, which requires packet numbering and reordering buffers in addition to depacketization buffers. Although these can be combined with buffers to hide variations in packet arrival (jitter) [49], random-access memories must be used to store packets, instead of potentially much cheaper FIFOs.
- Local methods can be extended to take information of a router neighborhood into account. In case of dynamic routing, routers



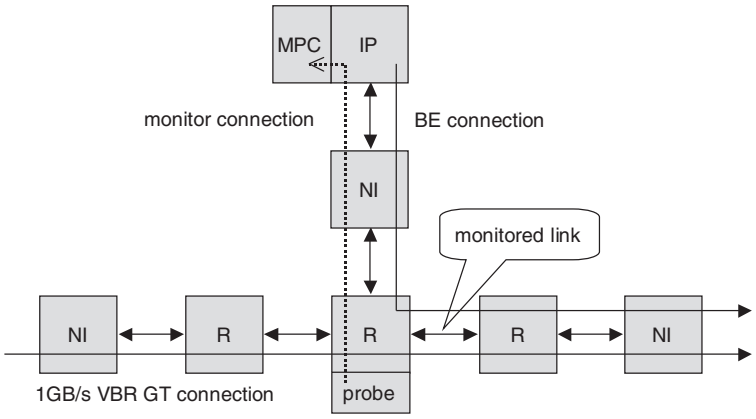
■ **FIGURE 5.17**

Dynamic routing around mesh center hot spot [83].

can exchange congestion information, and in this way route more efficiently around hot spots [83, 109]. Fig. 5.17 shows how congestion is centered on a small area in the left graph. When congestion control is added, the load spreads over a larger area and is reduced by a factor 20 (maximum load changes from 0.2 to 0.01), as shown in the right graph.

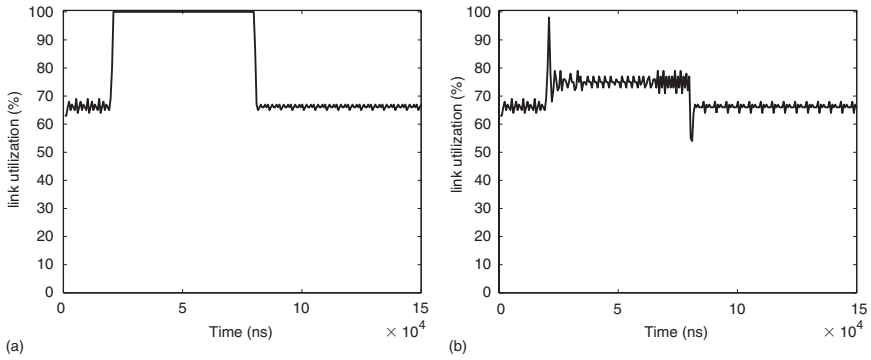
- Reducing the injection rates of packets into the network tackles congestion at a higher level. Rather than routing packets around congested areas, the number of packets in the network is reduced. Packet injection rates at network interfaces can be regulated either based on preset/precomputed values or based on measurements in the NoC (e.g., links [102] or network interfaces [85]). A (set of) rate controllers can regulate the packet injection rate, based on statistical information such as the offered load versus average latency distribution. For example, by measuring the utilization of a critical shared link, the injection rate (offered load) can be kept under limits that correspond to desired average latency. Van den Brand [102] shows how latency can be bounded by using NoC monitors [22] and multi-input multi-output *model-predictive controllers* (MPCs) (Fig. 5.18). An example is given in Fig. 5.19, where the monitor P observes the utilization of link L, and reports this information to the MPC. The controller regulates the packet injection rates of the producers to bound the link utilization (to 75% in the example). The graphs show the difference between link utilization of link L without and with use of the controller, respectively. The link utilization translates directly to latency experienced by BE traffic.

Note that all methods described above are reactive, and reduce the effects after contention or congestion has been detected. For this reason,



■ **FIGURE 5.18**

A small NoC with a monitoring probe and MPC [102].



■ **FIGURE 5.19**

Link utilization with (a) and without (b) NoC congestion control [102].

it is hard to provide hard end-to-end guarantees on bandwidth or latency. This is much easier when resources are reserved, and congestion is constrained to be within required bounds as described below.

Congestion control with resource reservations

As shown in Fig. 5.15, network resources such as buffers and links, are used in time and in space. Essentially, contention occurs when two packets use the same resource at the same time. It can therefore be avoided entirely, or bounded to a desired level by scheduling the time and location of all packets in the network. This is the aim of the category of congestion control methods described here, which reserve resources before they are used.

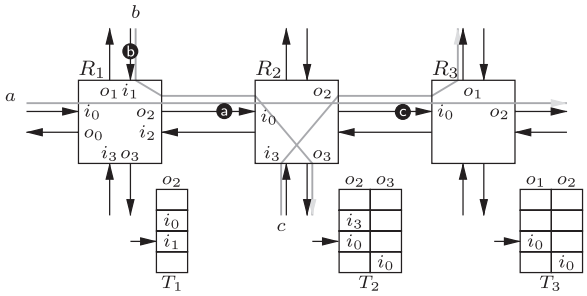
Producer and consumer network interfaces and the routers along the path(s) between them contain buffers and are interconnected by links (Fig. 5.15). Buffers and/or links are shared. Here we are concerned with resources that pertain to congestion, that is, in the router network only (Fig. 5.14). The resources in the network interfaces are managed by flow control, and will be described in Section 5.7.2.

A network user (usually a master–slave pair) must specify its requirements to the network before the network can decide what network resources are necessary for that flow. User requirements are stated using the service classes defined in Section 5.2. The network can then compute if sufficient resources are available to support the requested flow. If there are, the flow is accepted, else it is denied. This *admission control* (or set-up phase) is necessary to prevent invalidating the guaranteed QoS of flows already present in the network by overloading the network. Furthermore, the traffic that the communicating parties (e.g., master and slave) inject in the network must conform to the service class that was reserved for their flow. For example, if a CBR flow with a maximum of 50 MByte/s was requested, the master should not produce more data than promised. To guarantee the services of other flows, they should be enforced (*traffic policing*). That is, if a user does not obey the agreed traffic class, the network will enforce the contract, for example, by not accepting data. Finally, when a flow finishes, the network must be notified, after which the resources that were reserved can be freed for reuse by other flows (tear-down phase). The set-up and tear-down phases take time, and it is advisable therefore to use a flow for a longer period of time. (Although SOCBus has used flow for single transactions [107].) The aforementioned flows are often called connections [92], which are communication pipes between a master and (multiple) a slaves.

Below we describe two different methods to eliminate or reduce contention to within known bounds. They guarantee that packets that are injected in the network at the initiator network interface arrive at the target network interface within a predetermined interval. We highlight the strong relation of scheduling and buffering schemes (alluded to in Section 5.4.1):

- In the most extreme scheme, contention can be eliminated entirely by scheduling the time and location of all packets globally, using TDM. Packets are injected at the network's edge in such a way that they never collide at any link. This assumes that propagation speed of individual flits through the NoC is fixed and known in advance. This corresponds to a global notion of time (although not necessarily implemented through a synchronous clock). Flits wait in the network interfaces until they are injected in the NoC according to the TDM schedule. For lossless operation, some kind of end-to-end flow control is also necessary (described below in Section 5.7.2).

  The concept of globally scheduling all virtual circuits is used by NuMesh [96] (for parallel processing), Nostrum's looped



■ **FIGURE 5.20**

Eliminating contention through TDM [44].

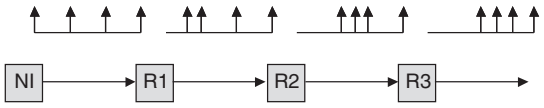
containers [75], aSoCs global scheduling [65, 66], and Æthereal's contention-free routing [44, 90].

We illustrate the concept with contention-free routing, as used in Æthereal [44], but the other methods are essentially the same. Every TDM slot table T has S time slots (rows), and N router outputs (columns). There is a logical notion of synchronicity: all routers in the network are in the same fixed-duration slot. In a slot s at most one flit can be read/written per input/output port. In the next slot, $(s+1)$ module S, the packets are written to their appropriate output ports. The entries of the slot table map outputs to inputs for every slot: $T(s,o)=i$, meaning that flits from input i (if present) are passed to output o at times $s+kS, k \in \mathbb{N}$. An entry is empty when there is no reservation for that output in that slot. There is no contention, by construction, because there is at most one input per output for each slot.

Fig. 5.20 illustrates the operation of contention-free routing. It shows a snapshot of a router network with three routers R_1, R_2, and R_3 at slot $s=2$, indicated by the arrows pointing to the third entry in the table. (Slots are numbered starting from zero.) The size of the slot tables is $S=4$, and only the relevant columns are depicted. Three flows, a, b, and c, are shown with the gray arrows; the black circles represents flits on the flow with the corresponding letter. Flit b is switched from input i_1 to output o_2 in router R_1, as indicated by the slot table $T_1(2,o_2)=i_1$. Similarly, flits a and c will be switched to outputs o_2 of R_2 and o_1 of R_3, respectively.

As discussed in Section 5.4.1, because flits never wait in the network, a single-flit buffer per input (or output) per router suffices. Network interfaces often use virtual circuit buffering, but this depends on the flow control scheme that is used (see Section 5.7.2).

The advantage of TDM for contention-free routing comes at a cost. The average latency for packets is relatively high, because although the time in the router network is minimal (no contention),



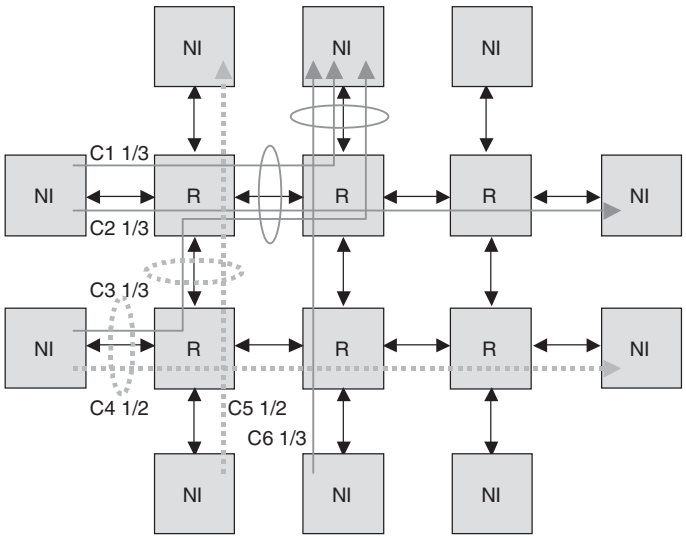
■ **FIGURE 5.21**

Characterizing congestion in rate-controlled methods [110].

they have to wait for their slots in the network interfaces, even if there are no other flits in the network that they could clash with. However, the worst-case latency is the same for TDM and rate-control described below. Note that with TDM bandwidth and latency, guarantees are (inversely) coupled, that is, a lower latency requires a larger slice of the bandwidth. Low-bandwidth high-priority flow (e.g., control traffic) may therefore be relatively expensive to offer (in terms of bandwidth overprovisioning).

- The essence of *rate-control schemes* [110] is that contention is allowed, but within known bounds. By regulating the amount of traffic that is injected at the network edge, the maximum contention that can arise at any point in the network can be computed. End-to-end latency follows from this. Each router is seen as an independent server, which means that routers must be non-blocking, that is, packets must not interfere with each other in switch or buffer usage. Each router accepts incoming traffic with a certain pattern, and produces outgoing traffic with a potentially modified pattern. Different schedulers can be used in the routers, with different results, in terms of average and worst-case latency, jitter, buffering requirements, etc. Starting with periodic traffic, two things can be observed in Fig. 5.21 [110]: (1) the incoming traffic pattern of a flow can be distorted due to network load fluctuations, (2) the distortion may make the traffic burstier and cause instantaneously higher rates. In the worst case, the distortion can be accumulated, and downstream routers potentially face burstier traffic than upstream routers. Therefore, the source network interface traffic characterization may not be applicable inside the network.

  Fig. 5.22 shows an example of how different flows can use a router network. The encircled links are the bottlenecks, determining the maximum allowed bandwidth ratios for the flows. At most three solid flows (C1, C2, C3, C6) use the same link, and they are allocated one-third of the link bandwidth each. The dashed flows (C3, C4) each share a link with at most one other flow, and can therefore use up to half of the link bandwidth. Essentially, any allocation of rates to flows is allowed, as long as the sum of rates allocated at each link does not surpass the link's capacity. At the network interfaces, the flows are constrained to not inject more than their allocated bandwidth ratio, for a fixed periodic time interval.



■ **FIGURE 5.22**

Flows and rates.

Buffers are required to deal with local bursts. The exact maximum burst size depends on the scheduling scheme used, but also on the time interval over which the flow rates are enforced (described above). This, together with the non-blocking requirement, means that virtual-circuit buffering with output queueing, or SAF switching with output queueing can be used. The latter results in large buffers, and has not been used for NoCs. Mango uses the former, with both round-robin arbitration [10] and priority-based ALG [9]. Both scheduling methods are work-conserving, which can result in lower average latencies in case the NoC is not fully utilized, unlike TDM as described before. Moreover, the ALG scheduler also decouples bandwidth and latency guarantees [9, 111]. However, ALG cannot schedule 100% of the available bandwidth for guaranteed services, unlike TDM and some other schedulers.

Reservation of resources can eliminate or bound congestion, but it comes at a price. Network users must negotiate their requirements with the network, which requires insight into their communication behavior, and which takes time (set-up and tear-down phases). Reservations may have to be for the worst-case traffic instead of the average case traffic, which can lead to overdimensioning of the NoC.[3] This is possible for ASIC,

[3] This depends on the exact service classes that are offered. But routers implementing guaranteed services may be smaller and faster than those implementing BE services [44], counteracting the impact of worst-case dimensioning.

ASSP, and FPGA NoCs. However, just like for connection-oriented circuit or packet switching, in CMP NoCs with dynamic applications where flows are short lived, change frequently, or very variable in their demands, resource reservations may be difficult to achieve.

Positive points are the possibility to offer hard guarantees regarding bandwidth, latency, jitter, etc. This has a host of advantages for system-level integration [45, 47], such as compositional system design and reduced verification effort, because modules and applications do not interfere with one another.

5.7.2 Flow Control

Using congestion control techniques described above, we can eliminate or reduce the congestion that packets encounter in the router network. However, they do not guarantee that there is space in buffers at the target network interface.[4] This happens when a target does not keep up with the incoming traffic, or even fails to respond entirely (e.g., through incorrect programming or malfunction). Its network interface buffers will fill up, and incoming packets must wait in the network. For example, the root of the dashed congestion tree in Fig. 5.15 could be caused by a slow slave, marked A. This may congest and even completely block the network, no matter what congestion method we use. Even when using congestion-free TDM, a full buffer still inhibits progress of other packets because their reserved link or buffer in the router network is, in fact, not available. Essentially, to avoid this phenomenon, packets from a master to a slave that cannot be accepted by a slave network interface must not inhibit the progress of packets on different flows (connections). Flow control, described below, ensures this.

Congestion control in the router network and flow control for network interfaces have different but related aims. Congestion control by itself is useful, for example, to increase performance, but not essential. However, flow control is often necessary even without congestion control, for example, to avoid deadlock [55]. (For example, the Æthereal BE traffic class uses no congestion control but does use flow control.)

Flow control deals with individual flows, that is, flows between a master and a slave (in its simplest form). Flow control ensures that the master does not send more data to the router network than can be accepted by the slave and its network interface. We describe a number of flow control techniques. First, three methods that do not require resource reservations:

1. The simplest form of flow control is to ensure that packets always find space in the slave network interface buffer. If the buffer is

[4] One may argue that the network interface buffers are no different than router buffers, but then the problem we will describe just shifts one "hop," into the slave. For clarity, we will omit any discussion of the behavior and internals (buffering, pipelining, etc.) of the slave, and concentrate on the network interfaces.

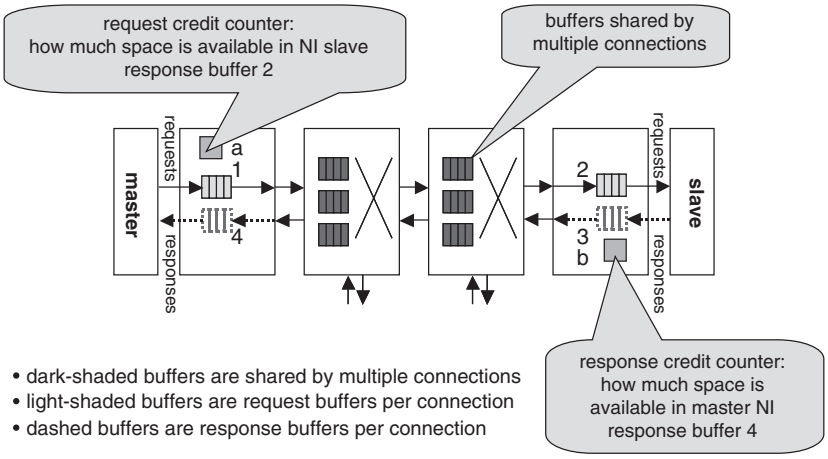
full packets are deleted (dropped), according to wine (drop new packets) or milk (drop old packets) policies. As discussed before, dropping increases congestion in the long run. For this reason, dropping packets as a flow control measure has not been used in NoCs. Note that no resource reservations are required for this method, and it fits quite well with dropping as a congestion control technique.

2. Returning packets that do not fit in buffers at the target network interface to the sender. To avoid introducing deadlock, the sender *must* accept the rejected packets. Although this technique has been used in computer networks, it is not (yet) used by any NoC [98].
3. Deflection routing for end-to-end flow control is used by SPIN [49] (and could be a natural option for Nostrum [75]). In other words, when a packet does not fit in the slave network interface buffer, it is sent back in the router network. Note that this is not the same as sending back the packet to the sender described above. There the slave does not intend accepting the packet at all, while here the slave temporarily refuses a packet with a view to accepting it later. This method combines well with using deflection routing for congestion control.

Methods that require resource reservations:

- Another method to ensure that packets always find space in the slave network interface buffer is to ensure that packets are only sent by the master network interface when there is guaranteed to be space in the slave network interface buffer. This can be achieved by *end-to-end flow control* based on credits or sliding windows [38, 97]. (Note that ACK/NACK, Go-Back-N, etc. are flow control schemes most commonly used for the link-layer and are described in Chapter 4.) These methods can be used with any buffering scheme. However, when virtual-circuit buffering (Section 5.4.1) is used a better approach exists, as described in the next point.

  Fig. 5.23 shows how credit-based end-to-end flow control operates. A master sends requests to a slave, who optionally returns responses. End-to-end flow control is required independently for both the request and the response flows. The figure shows routers that use shared buffers for all request and response traffic for all flows in the NoC. The routers have degree 3 with input buffers are shown. However, the buffer organization is not relevant here. Packets are allowed to leave the master network interface request buffer (1) only when there is space for them in the slave network interface request buffer (2). This is accomplished by tracking the free space in a credit counter (a). The counter is initialized to the capacity of the slave network interface request buffer (2), and is decremented



■ **FIGURE 5.23**

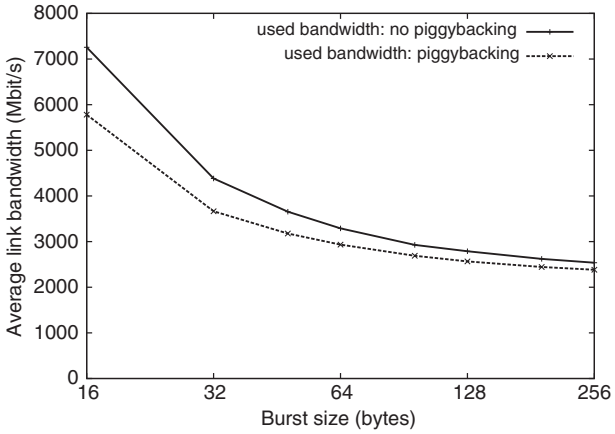
Credit-based end-to-end flow control.

whenever a packet is sent. Whenever a slave removes a packet from its request buffer (2), it sends a credit (in a packet) back to the master, who adds it to its credit counter (a). The slave network interface request buffer must be large enough to hide the delay introduced by sending back credits, otherwise the master will stall as long as the credits are in transit. Responses from slave to master are handled analogously.

Credit traffic can consume a substantial amount of the bandwidth (31% is quoted in Ref. [50]). As an optimization, the requested credit packets can be combined with response packets (and vice versa). This is called piggy backing [101], and can save substantial bandwidth (Fig. 5.24). As the burst size grows, the relative overhead introduced by credit packets decreases. With larger burst sizes, more credits are reported in a credit packet. Consequently, the number of credit packets (i.e., overhead introduced by credits) decreases. The drawback of increasing burst sizes is that larger buffers are required to accommodate the bigger bursts.

Æthereal [91] uses standard credit-based end-to-end flow control as described as above. Nostrum has been extended [42] to use the same flow control scheme.

SPIN [50] uses the same basic scheme, except that the credit counter is initialized with a value higher than the receiving buffer capacity. This can be interpreted as using the router buffers as an extension of the receiving buffer. However, this optimization introduces a dependency of flow control on congestion control. In SPIN this is solved by using deflection routing for both.



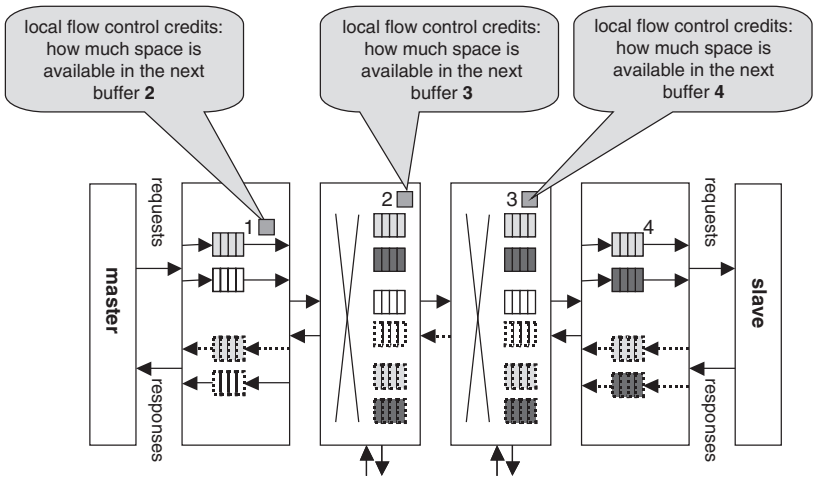
■ **FIGURE 5.24**

Piggy backing can save up to 20% on the average link bandwidth [91].

QNoC [105] uses end-to-end credits for both congestion control and flow control. Credits are not handed out on a flow (single master–single slave) basis as above. Instead a slave manages the requests of all masters that communicate with it in a single queue, and hence a single set of credits. A two-phase protocol is now used: a master requests credits from the slave using a high-priority traffic class, then it can send its data using the standard priority traffic class. This method is useful for heavily used slaves (i.e., those that cause congestion) with many masters (i.e., end-to-end flow control per master–slave pair is too expensive).

- The end-to-end flow control methods ensure that when packets enter the router network then they will after some time (depending on congestion) arrive at the receiving network interface, and find space in the appropriate buffer. Queueing in the router network must be avoided because packets of other flows *that use the same router buffers* may be blocked. In other words, by giving all flows their own buffers in every router along their paths (virtual circuit buffering, Section 5.4.1), a packet that is blocked causes no harm to other flows.

  As briefly mentioned in Section 5.4, routers synchronize the transfer of flits to ensure that the sending buffer is not empty, and that the receiving buffer does not overflow. This is performed by link-level flow control, as described in Chapter 4. Note that link-level flow occurs per virtual circuit, and if one virtual circuit is blocked, another may still progress. Now, end-to-end flow control is automatically obtained by a chain of link-level flow controls, as shown



■ **FIGURE 5.25**

Virtual-circuit buffering with credit-based local flow control.

in Fig. 5.25. Although the figure shows credit-based link-level flow control, other techniques are also possible (see Chapter 4).

Mango [8] and [106] use this approach. Although currently not the case, Nostrum [75] could use this approach for its containers (under the restriction that containers transport data from initiator to target only).

We have described congestion control and flow control, and why each is desirable or necessary. In the next section we see that both together are required if NoC offers QoS performance guarantees.

5.7.3 Congestion Control and Flow Control for QoS

We have described how congestion control and flow control regulate the interaction or interference between different packets and flows in the network. In either case, resources can be (explicitly) reserved or not, as summarized in Table 5.2.

As mentioned a number of times in this section, without resource reservations it is in general impossible to give hard (100%) guarantees. If either congestion control or flow control does not use resource reservations, no end-to-end guarantees such as minimum bandwidth and maximum

TABLE 5.2 ■ Overview of congestion control and flow control.

	Without reservations	With reservations
Congestion control	Cropping Dynamic routing Adaptive injection	Contention-free routing Rate control
Flow control	Dropping Refusing Deflection	End-to-end flow control Virtual-circuit flow control

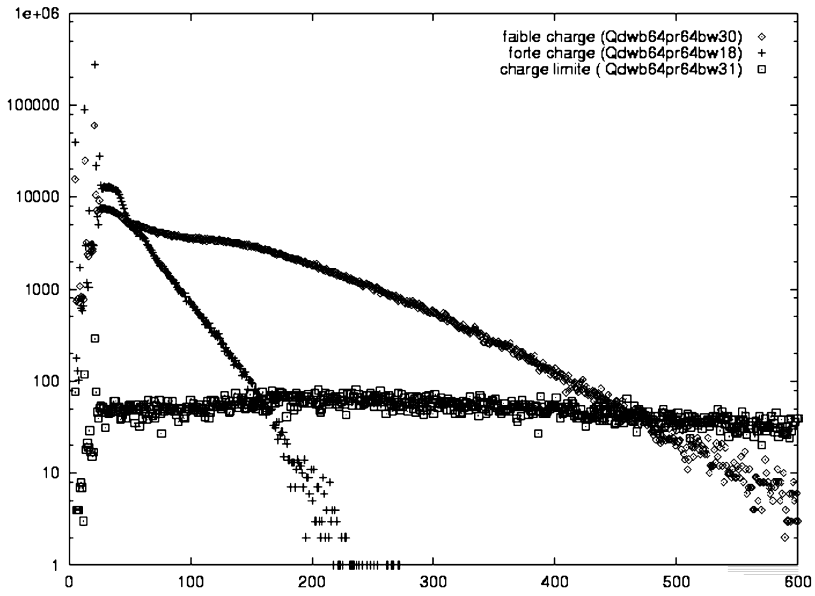
latency service classes can be offered. Congestion control is required to eliminate or characterize the interference between different flows in the router network. Flow control characterizes the interaction between the sender and the receiver. Without it a single slow or non-responding master or slave can cause loss of data (if dropping is used), or congestion or even complete blockage of the NoC. No bandwidth or latency guarantees can be given in all these cases.

When no hard performance guarantees can be given, the NoC behavior can be described statistically. Under the assumption of particular traffic patterns, distributions of for example, packet delays can be computed, as well as statistical bounds (for a given confidence level). For example, Guerrier [49] shows that in SPIN network latencies occur with an exponentially decreasing probability (Fig. 5.26). Note that the number of packets (vertical axis) is exponential. The lowest network latencies are obtained at the lowest (27%) offered load, where over 90% of packets arrive within 50 cycles. The almost horizontal curve corresponds to a saturated NoC at 47% offered load. Beyond this load the module-to-module latency rapidly becomes unbounded.

Fig. 5.26 shows that very long latencies will occur, although infrequently. However, note that "improbable" failures (e.g., 10^{-14} chance of being "late") occur often, with high NoC operating frequencies (e.g., 500 MHz), concurrent transactions (e.g., 20 active packets): every $20 \times 5 \times 10^{12} \times 10^{-14} =$ every second. In an NoC with BE service classes only, reducing the failure rate to an acceptable level may mean drastically reducing the offered load, or equivalently, the utilization of the NoC.

However, there are advantages when resources do not need to be reserved. The set-up and tear-down phases are not required, and at any point in time resources are automatically distributed over flows (although perhaps unfairly [105]). No resource re-allocation (reconfiguration) is necessary, when NoC usage changes.

Few NoCs have implemented both congestion control and flow control with a view to give hard QoS guarantees. In particular, Mango [6], Æthereal [44], and SonicsMX [106] offer complete solutions. However, a number of NoCs can be extended with techniques described above (usually end-to-end or chained link-level flow control), and they too will be



■ **FIGURE 5.26**

Network latency distributions for 27%, 45%, and 47% offered load [49].

able to offer the same level of QoS guarantees. This category contains Nostrum [75], aSoC [65], and Refs [63, 108].

NoCs differ in the granularity of resource reservations they make. There are two ends of the spectrum. The first precisely maps traffic classes to a larger number of tailored service classes, where the other more coarsely maps traffic classes to fewer, more generic service classes (Section 5.2). The former enables accurate traffic scheduling for high effective NoC utilization, at the cost of precise traffic characterizations with tailored service classes. The latter reduces the burden of precise traffic characterization, at the cost of lower effective NoC utilization. Hence, the respective approaches trade off accuracy of mapping of traffic classes to service classes versus effective NoC utilization. The QoS approaches, introduced in Section 5.2.1, of QNoC [13] (four service classes) and SPIN [49] (a single BE service class) may be interpreted as examples of these respective approaches, with Nostrum [75] and Æthereal [44] being in between (with guaranteed-bandwidth and BE service classes).

Which approach is most suitable depends on the application requirements and NoC type (ASIC, ASSP, FPGA, or CMP). For example, real-time streaming applications (such as video) tend to be long lived, and minimum bandwidth and low jitter must be guaranteed, but low latency is less important. Resource reservations to optimally schedule link and buffer utilization reduces the cost of the system, at the cost of explicit flow set-up.

This is acceptable in ASIC and ASSP NoCs, which are the natural implementation for this sort of applications for reasons of performance and cost. On the other hand, cache traffic of embedded processors of CMPs, is unpredictable in terms of (instantaneous) bandwidth usage and the average and worst-case bandwidth are very different. Average low latency is essential for good performance. Hence, precise characterization of cache traffic is difficult, and overdimensioning would be very expensive. As a result, most traffic classes may be mapped to the BE service class, which needs no or few reservations.

5.8 SUMMARY

In this chapter we described the main principles in designing the networking and transport layers of NoCs. These layers include the specification of the following NoC characteristics: switching technique, topology, addressing and routing, and end-to-end congestion and flow control schemes. We presented this complex problem as a constrained optimization process. The NoC designer should minimize the cost of the NoC which is expressed in terms of VLSI area and power consumption added to his overall chip design. The constraints are the level of services (service classes) to be provided for the traffic patterns of the SoC under consideration. Since NoCs can be designed anew for each SoC implementation, this optimization process is expected to be repeated for each new SoC design, using NoC CAD tools.

Following this definition, it is clear that the SoC traffic characteristics strongly impact the NoC characteristics. We therefore classified SoCs according to how much is known in advance about their functionality, and consequently about their expected module-to-module communication patterns (Fig. 5.1).

We reviewed topical state-of-the-art solutions for each of the important NoC characteristics: switching mechanisms, QoS implementations, topological design, routing mechanisms, congestion and flow control techniques. We showed how the NoC model impacts the choices available, for each of these, as well as the relationships and trade-offs between them.

REFERENCES

[1] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez and C.A. Zeferino, "SPIN: A Scalable, Packet Switched, On-Chip Micro-Network," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2003.

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

[3] ATM Forum, *ATM User-Network Interface Specification*, Prentice Hall, July 1994, Version 3.1.
[4] C. Aurrecoechea, A.T. Campbell and L. Hauw, "A Survey of QoS Architectures," *Multimedia Systems*, Vol. 6, No. 3, 1998, pp. 138–151.
[5] D. Bertozzi and L. Benini, "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip," *IEEE Circuits and Systems Magazine*, Vol. 4, No. 2, 2004, pp. 18–31.
[6] T. Bjerregaard, *The MANGO Clockless Network-on-Chip: Concepts and Implementation*, Ph.D. thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2006.
[7] T. Bjerregaard, S. Mahadevan, R. Grøndahl and J. Sparsø, "An OCP Compliant Adapter for GALS-Based SoC Design Using the MANGO Network-on-Chip," *Proceedings of the International Symposium on Systems on Chip (SoC)*, 2005.
[8] T. Bjerregaard and J. Sparsø, "A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2005, pp. 1226–1231.
[9] T. Bjerregaard and J. Sparsø, Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip," *Proceedings of the International Symposium on Asynchronous Circuits and Systems (ASYNC)*, March 2005, pp. 34–43.
[10] T. Bjerregaard and J. Sparsø, "Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip," *IEE Proceedings: Computers and Digital Techniques*, 2006.
[11] H. Bodlaender, R. Tan, D. Thilikos and J. van Leeuwen, *On Interval Routing Schemes and Treewidth*, 1995.
[12] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Cost considerations in Network on Chip," *Integration, the VLSI Journal*, Vol. 38, No. 1, October 2004, 19–42.
[13] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QNoC: QoS Architecture and Design Process for Network on Chip," *Journal of Systems Architecture*, Vol. 50, No. 2–3, February 2004, pp. 105–128. Special issue on Networks on Chip.
[14] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Efficient Routing in Irregular Topology Nocs," *Technion, CCIT Report*, Vol. 554, No. 5, September 2005.
[15] F. Borgonovo, L. Fratta and J.A. Bannister, "On the Design of Optical Deflection-Routing Networks," *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1994, pp. 120–129.
[16] R. Braden, D. Clark and S. Shenker, RFC 1633: *Integrated Services in the Internet Architecture: an Overview*, June 1994.
[17] J. Brassil and R.L. Cruz, "Bounds on Maximum Delay in Networks with Deflection Routing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 7, 1995, pp. 724–732.
[18] I. Cidon, R. Guérin and A. Khamisy, "On Protective Buffer Policies, *IEEE/ACM Transactions on Networking*, Vol. 2, No. 3, 1994, pp. 240–246.
[19] I. Cidon, F.M. Jaffe and M. Sidi, "Local Distributed Deadlock Detection by Cycle Detection and Clustering," *IEEE Transactions on Software Engineering*, Vol. 13, No. 1, 1987, pp. 3–14.

[20] I. Cidon, J.M. Jaffe and M. Sidi, "Distributed Store-and-Forward Deadlock Detection and Resolution Algorithms, "*IEEE Transactions on Communication*, Vol. 35, No. 11, 1987, pp. 1139–1145.

[21] I. Cidon and I. Keidar, "Zooming in on Network on Chip Architectures," *Technion, CCIT Report*, Vol. 565, No. 5, December 2005.

[22] C. Ciordaş, T. Basten, A. Rădulescu, K. Goossens and J. van Meerbergen, An Event-Based Network-on-Chip Monitoring Service," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 4, October 2005, pp. 702–723. HLDVT'04 Special Issue on Validation of Large Systems.

[23] D.J. Culler, J.P. Singh and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers, 1999.

[24] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[25] W.J. Dally, "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," *IEEE Transactions on Computers*, Vol. 40, No. 9, September 1991, pp. 1016–1023.

[26] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 2, March 1992, pp. 194–205.

[27] W.J. Dally and H. Aoki, Adaptive routing using virtual channels, Technical Report, Laboratory for Computer Science, MIT, Cambridge, Massachusetts, September 1990.

[28] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 4, April 1993, pp. 466–475.

[29] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, Vol. 36, No. 5, May 1987, pp. 547–553.

[30] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proceedings of the Design Automation Conference (DAC)*, 2001, pp. 684–689.

[31] A. DeHon, Robust, high-speed network design for large-scale multiprocessing, A.I. Technical report 1445, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, September 1993.

[32] J. Dielissen, A. Rădulescu, K. Goossens and E. Rijpkema, "Concepts and Implementation of the Philips Network-on-Chip, In *Workshop on IP-Based System-on-Chip Design*, November 2003.

[33] J. Duato, S. Yalamanchili and L. Ni, "*Interconnection Networks – An Engineering Approach*, Morgan Kaufmann, 2003.

[34] T. Dumitras and R. Mărculescu, "On-Chip Stochastic Communication," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2003.

[35] U. Feige and P. Raghavan, "Exact Analysis of Hot-Potato Routing," *Proceedings of the Annual Symposium on Foundations of Computer Science*, October 1992, pp. 553–562.

[36] E. Fleury and P. Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-Routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 7, July 1998, pp. 626–638.

[37] O.P. Gangwal, A. Rădulescu, K. Goossens, S. González Pestana and E. Rijpkema, "Building Predictable Systems on Chip: An Analysis of Guaranteed Communication in the Æthereal Network on Chip," in P. van der Stok

(editor), *Dynamic and Robust Streaming in and Between Connected Consumer-Electronics Devices*, Vol. 3. *Philips Research Book Series*, Springer, 2005, Chapter 1, pp. 1–36.

[38] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Transactions on Communications*, Vol. COM-28, No. 4, 553–574, April 1980.

[39] R. Gindin, I. Cidon and I. Keidar, NoC architecture for future fpgas, Department of EE, CCIT Report 579, Technion, March 2006.

[40] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *International Symposium on Computer Architecture*, May 1992, pp. 278–287.

[41] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *Journal of the ACM*, Vol. 41, No. 5, September 1994, pp. 874–902.

[42] S. González Pestana, K. Goossens, A. Rădulescu and R. Thid. Framework and performance metric definitions: A first step towards network-on-chip benchmarking, Technical Note 2006/00003, Philips Research, January 2006.

[43] K. Goossens, J. Dielissen, O.P. Gangwal, S. González Pestana, A. Rădulescu and E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2005, pp. 1182–1187.

[44] K. Goossens, J. Dielissen and A. Rădulescu, "The Æthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Design and Test of Computers*, Vol. 22, No. 5, September–October 2005, pp. 21–31.

[45] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavko, A. Rădulescu, E. Rijpkema, E. Waterlander and P. Wielage, "Guaranteeing the quality of Services in Networks on Chip," In A. Jantsch and H. Tenhunen (editors), *Networks on Chip*, Kluwer, 2003, Chapter 4, pp. 61–82.

[46] K. Goossens, O.P. Gangwal, J. Röver and A.P. Niranjan, "Interconnect and Memory Organization in SOCs for Advanced Set-Top Boxes and TV – Evolution, Analysis, and Trends," In J. Nurmi, H. Tenhunen, J. Isoaho and A. Jantsch (editors), *Interconnect-Centric Design for Advanced SoC and NoC*, Kluwer, 2004, Chapter 15, pp. 399–423.

[47] K. Goossens, J. van Meerbergen, A. Peeters and P. Wielage, "Networks on Silicon: Combining Best-Effort and Guaranteed Services, in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2002, pp. 423–425.

[48] A.G. Greenberg and N. Madras, "How Fair Is Fair Queueing," *Journal of the ACM*, 1992, pp. 568–598.

[49] P. Guerrier, *Un Réseau D'Interconnexion pour Systémes Intégrés*, Ph.D. thesis, Université Paris VI, March 2000.

[50] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2000, pp. 250–256.

[51] K.D. Günther. "Prevention of Deadlocks in Packet-Switched Data Transport System," *IEEE Transactions on Communications*, Vol. 29, April 1981, pp. 512–524.

[52] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Efficient Link Capacity and QoS Design for Wormhole Network-on-Chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2005, pp. 9–14.

[53] A. Hansson, K. Goossens and A. Rădulescu, UMARS: A unified approach to mapping and routing on a combined guaranteed service and best-effort network-on-chip architecture, Technical Report 2005/00340, Philips Research, April 2005.

[54] A. Hansson, K. Goossens and A. Rădulescu, "A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures," *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, September 2005, pp. 75–80.

[55] A. Hansson, K. Goossens and A. Rădulescu, Analysis of message-dependent deadlock in network-based systems on chip, Technical Report 2006/00230, Philips Research, March 2006.

[56] M. Harmanci, N. Escudero, Y. Leblebici and P. Ienne, "Quantitative Modelling and Comparison of Communication Schemes to Guarantee Quality-of-Service in Networks-on-Chip," *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, 2005, pp. 1782–1785.

[57] J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2003, pp. 688–693.

[58] J. Hu and R. Marculescu, "DyAD – Smart Routing for Networks on Chip," *Proceedings of the Design Automation Conference (DAC)*, June 2004.

[59] J. Hu and R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures, *IEEE Transactions on CAD of Integrated Circuits and Systems*, Vol. 24, No. 4, April 2005, pp. 551–562.

[60] P. Humblet, A. Bhargava and M.G. Hluchyj, "Ballot Theorems Applied to the Transient analysis of nD/D/1 Queues," *IEEE/ACM Transactions on Networking*, 1993, Vol. 1, No. 1, pp. 81–95.

[61] A. Jalabert, S. Murali, L. Benini and G. De Micheli, "XpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2004.

[62] N. Kavaldjiev, G.J.M. Smit and P.G. Jansen, "A Virtual Channel Router for On-Chip Networks," *Proceedings of the International SOC Conference (SoCC)*, September 2004.

[63] H.T. Kung, T. Blackwell and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation and Statistical Multiplexing." *SIGCOMM*, 1994, pp. 101–114.

[64] A. Laffely, J. Liang, P. Jain, N. Weng, W. Burleson and R. Tessier, "Adaptive Systems on a Chip (aSoC) for Low-Power Signal Processing, *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, 2001.

[65] A.J. Laffely, *An Interconnect-Centric Approach for Adapting Voltage and Frequency in Heterogeneous System-on-a-Chip*. Ph.D. thesis, University of Massachusetts Amherst, 2003.

[66] K. Lee, S.-J. Lee and H.-J. Yoo, "A Distributed Crossbar Switch Scheduler for On-Chip Networks," *Proceedings of the Custom Integrated Circuits Conference*, 2003.

[67] K. Lee, S.-J. Lee and H.-J. Yoo, "A High-Speed and Lightweight On-Chip Crossbar Switch Scheduler for On-Chip Interconnection Networks," *Proceedings of the International Conference on European Solid-State Circuits*, 2003.

[68] C. Leiserson, "Fat-Trees: Universal Networks for Hardware-efficient supercomputing," *IEEE Transactions on Computers*, October 1985, Vol. C-34, No. 10, pp. 892–901.
[69] D.H. Linder and J.C. Harden, "An adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes," *IEEE Transactions on Computers*, Vol. 40, No. 1, January 1991, pp. 2–12.
[70] A. Litman and S. Moran-Schein, "Fast, Minimal, and Oblivious Routing Algorithms on the mesh with bounded Queues," *Journal of Interconnection Networks*, 2001, Vol. 2, No. 4, pp. 445–469.
[71] P. López, J.M. Martínez and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks," *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, February 1998, pp. 57–66.
[72] N.F. Maxemchuk, "Routing in the Manhattan Street Network," *IEEE Transactions on Communication*, Vol. COM-35, No. 2–3, May 1987, pp. 503–512.
[73] D.E. McDysan and D.L. Spohn, *ATM: Theory and Application*, McGraw-Hill, Inc., New York, NY, USA, 1994.
[74] M. Millberg, E. Nilsson, R. Thid and A. Jantsch, "Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2004.
[75] M. Millberg, E. Nilsson, R. Thid, S. Kumar and A. Jantsch, "The Nostrum Backbone – a Communication Protocol Stack for Networks on Chip," *Proceedings of the International Conference on VLSI Design*, 2004, pp. 693–696.
[76] P. Mohapatra, "Wormhole Routing Techniques for Directly Connected Multicomputer Systems." *ACM Computing Surveys*, Vol. 30, No, 3, 1998, pp. 374–410.
[77] S. Murali, L. Benini and G. de Micheli, "Mapping and Physical Planning of Networks on Chip Architectures with Quality of Service Guarantees," *Proceedings of the Design Automation Conference, Asia and South Pacific (ASPDAC)*, 2005.
[78] S. Murali, M. Coenen, A. Rădulescu, K. Goossens and G. De Micheli, "A Methodology for Mapping Multiple Use-Cases on to Networks on Chip. *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2006, pp. 118–123.
[79] S. Murali and G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NOCs," *Proceedings of the Design Automation Conference (DAC)*, June 2003.
[80] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, February 1993, Vol. 26, No. 2, pp. 62–76.
[81] K. Nichols, S. Blake, F. Baker and D. Black. RFC 2474: Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers, December 1998.
[82] E. Nilsson, M. Millberg, J. Öberg and A. Jantsch, "Load Distribution with the Proximity Congestion Awareness in a Network on Chip," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2003.
[83] V. Nollet, T. Marescaux, P. Avasare, D. Verkest and J.-Y. Mignolet, Centralized Run-Time Resource Management in a Network-on-Chip Containing

Reconfigurable Hardware Tiles," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2005, pp. 234–239.

[84] V. Nollet, T. Marescaux and D. Verkest, "Operating-System Controlled Network on Chip. *Proceedings of the Design Automation Conference (DAC)*, June 2005, pp. 256–259.

[85] OCP International Partnership, *Open Core Protocol Specification*, 2001.

[86] U.Y. Ogras, J. Hu and R. Marculescu, "Key Research Problems in NoC Design: A Holistic Perspective," *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, September 2005.

[87] P. Pande, C. Grecu, Ivanov and R. Saleh, "Design of a Switch for Network on Chip Applications," *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, 2003.

[88] E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage and E. Waterlander, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip," *IEE Proceedings: Computers and Digital Technique*, September 2003, Vol. 150, No. 5, pp. 294–302.

[89] E. Rijpkema, K. Goossens and P. Wielage, "A Router Architecture for Networks on Silicon," *Proceedings of Progress 2001, 2nd Workshop on Embedded Systems*, Veldhoven, the Netherlands, October 2001.

[90] A. Rădulescu, J. Dielissen, S. González Pestana, O.P. Gangwal, E. Rijpkema, P. Wielage and K. Goossens, "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming," *IEEE Transactions on CAD of Integrated Circuits and Systems*, January 2005, Vol. 24, No. 1, pp. 4–17.

[91] A. Rădulescu and K. Goossens, "Communication Services for Networks on Chip," In S.S. Bhattacharyya, E.F. Deprettere and J. Teich (editors), *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, Marcel Dekker, 2004, pp. 193–213.

[92] I. Saastamoinen, M. Alho, J. Pirttimäki and J. Nurmi, "Proteo Interconnect IPs for Networks-on-Chip," In *IP Based Design 2002*, 2002.

[93] D. Seo, A. Ali, W.-T. Lim, N. Rafique and M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks, *International Symposium on Computer Architecture*, 2005, pp. 432–443.

[94] L. Shang, L.-S. Peh and N.K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," IEEE Computer Society, *HPCA '03: Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, Washington, DC, USA, 2003, pp. 91.

[95] D. Shoemaker, *An Optimized Hardware Architecture and Communication Protocol for Scheduled Communication*, Ph.D. thesis, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, May 1997.

[96] V. Shurbanov, D. Avresky, P. Mehra and W. Watson, "Flow Control in Servernet Clusters," *The Journal of Supercomputing*, June 2002, Vol. 22, No. 2, pp. 161–173.

[97] 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*, 2003, Vol. 14, pp. 259–275.

[98] F. Steenhof, H. Duque, B. Nilsson, K. Goossens and R. Peset Llopis, "Networks on Chips for High-End Consumer-Electronics TV System Architectures," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2006, pp. 148–153.

[99] P. Stravers and J. Hoogerbrugge, "Homogeneous Multiprocessing and the Future of Silicon Design Paradigms," In *VLSI-TSA*, 2001.

[100] A.S. Tanenbaum, *Computer Networks*, Prentice-Hall, 1996.

[101] J.W. van den Brand, C. Ciordaş and T. Basten, Runtime Networks-on-Chip Performance Monitoring, Technical Report 2006/00218, Philips Research, March 2006.

[102] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe and A. Agarwal, "Baring It All to Software: Raw Machines," *IEEE Computer*, September 1997, Vol. 30, No. 9. pp. 86–93.

[103] I.Z. Walter, Quality of Service in Network-on-Chip, Master's thesis, Technion, Israel Institute of Technology, August 2005.

[104] W.-D. Weber, J. Chou, I. Swarbrick and D. Wingard, "A Quality-of-Service Mechanism for Interconnection Networks in System-on-Chips. *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2005.

[105] D. Wiklund, *Development and Performance Evaluation of Networks on Chip*, Ph.D. thesis, Department of Electrical Engineering, Linköping University, 2005.

[106] P.T. Wolkotte, G.J. Smit, G.K. Rauwerda and L.T. Smit, "An Energy-Efficient Reconfigurable Circuit Switched Network-on-Chip," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, April 2005.

[107] T.T. Ye, L. Benini and G. De Micheli, "Packetization and Routing Analysis of On-Chip Multiprocessor Networks," *Journal of Systems Architecture*, February 2004, Vol. 50, No. 2–3, pp. 81–104. Special issue on Networks on Chip.

[108] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, October 1995, Vol. 83, No. 10, pp. 1374–1396.

[109] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing. *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1993, pp. 227–236.