Network-on-Chip Design 15

Haseeb Bokhari and Sri Parameswaran

Abstract

Continuous transistor scaling has enabled computer architecture to integrate increasing numbers of cores on a chip. As the number of cores on a chip and application complexity has increased, the on-chip communication bandwidth requirement increased as well. Packet-switched network on chip (NoC) is envisioned as a scalable and cost-effective communication fabric for multi-core architectures with tens and hundreds of cores. In this chapter we focus on on-chip communication architecture design and introduce the reader to some essential concepts of NoC architecture. This is followed by a discussion on the commonly used power-saving techniques used for NoCs and the drawbacks and limitations of these techniques. We then concentrate on performance optimization through intelligent mapping of applications on multi-core architectures. We conclude the chapter with a discussion of various application-specific on-chip interconnect design methods.

Acronyms

CMOS	Complementary Metal-Oxide-Semiconductor
CMP	Chip Multi-Processor
DOR	Dimension Ordered Routing
DRAM	Dynamic Random-Access Memory
DVFS	Dynamic Voltage and Frequency Scaling
IP	Intellectual Property
MPSoC	Multi-Processor System-on-Chip
NI	Network Interface
NoC	Network-on-Chip
SoC	System-on-Chip
TDMA	Time-Division Multiple Access

H. Bokhari (✉) • S. Parameswaran
University of New South Wales (UNSW), Sydney, NSW, Australia
e-mail: hbokhari@cse.unsw.edu.au; sridevan@cse.unsw.edu.au

S. Ha, J. Teich (eds.), *Handbook of Hardware/Software Codesign*,
DOI 10.1007/978-94-017-7267-9_16

Contents

15.1 On-Chip Interconnect Architecture

Every multi-core chip has two major on-chip components: processing elements (*core*) and other non-processing elements such as communication and memory architecture (*uncore*) [27]. Although high transistor density enables computer architects to integrate tens to hundreds of cores in a chip, the main challenge is to enable efficient communication between such a large number of on-chip components. The on-chip communication architecture is responsible for all memory transactions and I/O traffic and provides a dependable medium for inter-processor data sharing. The performance of on-chip communication plays a pivotal role in the overall performance of the multi-core architecture. The advantage of having multiple high-performance on-chip processors can easily be overturned by an underperforming on-chip communication medium. Hence, providing a scalable and high-performance on-chip communication is a key research area for multi-core architecture designers [17]. The main challenges faced by interconnect designers are:

- *Scalable communication for tens of cores*: It is fair to state that the performance of processing elements in multi-core chips can be communication constrained [17]. Due to ever-increasing improvement in processing capabilities, it is quite possible to have a wide gap between the data communication rate and the data consumption rate. With tens of on-chip components, it is not possible to have single-cycle communication latency between components placed at the far ends of a chip. Furthermore, with a large number of on-chip components, the on-chip interconnect is expected to support multiple parallel communication streams.
- *Limited power budget*: In 1974, Dennard predicted that the power density of transistor will remain constant as we move into lower node sizes. This is known as *Dennard's scaling law* [37]. However, in the last decade or so, researchers

have observed that the transistor's power cannot be reduced at the same rate as the area. Therefore, we are facing a situation where we have an abundance of on-chip transistors but do not have enough power to switch all these transistor at the same time, due to power and thermal constraints. Therefore, increasing the power efficiency of all on-chip components has become the main prerequisite to continue Moore's scaling. The on-chip communication architecture can consume roughly 19% of total chip power in a modern multi-core chip [35]. Therefore, it is a challenging task to design a power-efficient on-chip interconnect that can still satisfy the latency and bandwidth requirements of current and future applications.

- *Heterogeneous applications*: A modern multi-core chip is expected to execute a large set of diverse applications. Each application can interact with computing architecture in a unique way; hence, the communication latency and bandwidth requirement can vary across different applications [32]. For example, an application with a large memory footprint is expected to regularly generate cache misses and can hence be classified as a communication-bound application. The performance of such applications is highly correlated with the efficiency of interconnect. On the other hand, an application with a smaller memory footprint is expected to be processor bound and agnostic to on-chip interconnect properties. Therefore, on-chip interconnects are often designed for worst-case scenarios (memory-bound applications in this case) and can therefore be inefficient for processor-bound applications. The situation is aggravated when both memory- and processor-bound applications are executed at the same time.
- *Selecting interconnect performance metrics*: A major shortcoming in previous research is classifying the on-chip interconnect performance in terms of application-agnostic metrics such as transaction latency and memory bandwidth, instead of application-level performance metrics such as execution time and throughout [31, 70]. Therefore a major challenge is extracting the correct metric to evaluate different possible interconnect architecture design points for a given set of applications.
- *Chip design cost*: The cost of designing a multi-core chip has been increasing alarmingly due to high NRE cost (NRE Cost: *Nonrecurring engineering cost*. The term is used to classify the cost associated with researching, prototyping, and testing a new product.) associated with small node sizes. A major portion of total chip cost is reserved for design verification and testing. Therefore, designers are expected to reuse previously designed and verified on-chip interconnects for new chip designs to reduce cost. With a multitude of interconnect architectures available, it is important to incorporate time-efficient design space exploration tools in research phase to select the most suitable interconnect for a given set of target applications.
- *Interconnect reliability*: With reducing node size, the concerns about the reliability of the digital circuits are on the rise. Any unexpected change in operating conditions such as supply voltage fluctuation, temperature spikes, or a random alpha particle collision can cause erratic behavior in the output of a circuit. A soft error in on-chip interconnect can result in erroneous application output or system deadlock if the control data is corrupted. Multi-core systems are finding

their ways into reliability-critical applications such as autonomous driving cars and medical equipment. Therefore designers are expected to integrate varying levels of reliability features in on-chip interconnect under given power and area constraints.

- *Codesign of memory hierarchy and on-chip interconnect*: In modern multi-core architectures, on-chip memory hierarchy is closely coupled with on-chip interconnect architecture. In fact, for shared memory architectures, on-chip communication is the major factor in deciding the performance of memory hierarchy (cache, Dynamic Random-Access Memory (DRAM) controllers, etc.). Therefore interconnect designers are often faced with the challenge of exploring the combined design space of memory hierarchy and on-chip interconnect.

15.1.1 Bus-Based SoC Architectures

Traditionally, system-on-chip (SoC) designs used a very simple on-chip interconnect such as ad hoc point-to-point connections or buses. The bus-based architecture is perhaps the oldest on-chip interconnect standard in the computer industry and is still used in many System-on-Chip (SoC) applications [87]. The simplicity of protocol and hence low gate cost are possibly the main reasons that bus-based architectures have dominated all other available on-chip interconnect options. For a small number of on-chip components, bus interconnect is easier to integrate due to simple protocol design and is efficient in terms of both power and silicon cost. In bus-based architectures, multiple components interact using a single data and control bus, hence providing a simple master-slave connection. Arbitration is required when multiple masters try to communicate with a single slave, giving rise to resource contention. Hence the scalability of bus-based architecture in terms of performance is questionable in large SoC-based designs [61]. Some classic design techniques for bus-based SoCs proposed in [34, 41] use worst-case bus traffic to design optimal architecture. Kumar et al. [57] have given a detailed study of the scalability and performance of shared bus-based Chip Multi-Processor (CMP) architectures. They concluded that a bus-based interconnect network can significantly affect the performance of cache-coherent CMPs.

Several improvements to traditional bus-based interconnect architectures have been proposed. ARM Ltd., AMBA Architecture [1], IBM CoreConnect Architecture [3], and Tensilica PIF Interface [5] are few examples of the widely used advanced bus-based communication medium. All of these architectures provide several advanced functionalities like burst-based data transfers, multi-master arbitration, multiple outstanding transactions, bus locking, and simultaneous asynchronous and synchronous communication. However, Rashid et al. [91] have analytically showed that even advanced bus-based architectures like AMBA are outperformed by modern Network-on-Chip (NoC)-based communication architectures in terms of performance. However, the same study shows that designers are still inclined toward AMBA-based on-chip interconnects due to the area and energy overhead of modern NoC designs. SoC designers have a keen interest in fair comparison of various

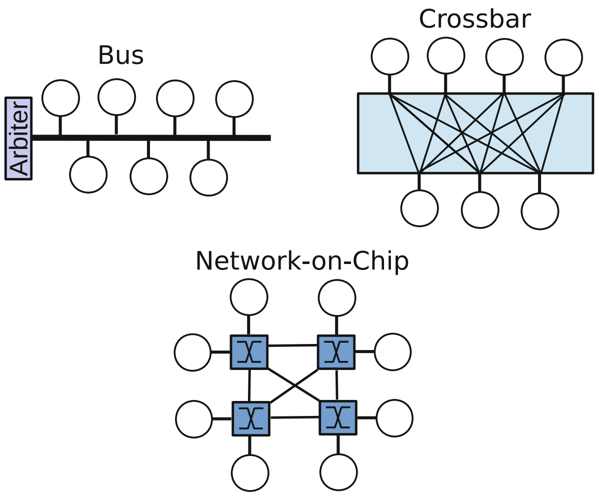

Fig. 15.1 Evolution of on-chip interconnect

commercial bus architectures; however, the performance of on-chip interconnects greatly depends on each application's traffic pattern, bus microarchitecture, and system parameters [67].

The simplicity of the bus-based architecture design, predictable access latency, and low area overhead are the key selling points. However, beyond a small number of cores, the performance of the bus interconnect degrades significantly [83].

15.1.2 Crossbar-on-Chip Interconnect

Single shared bus architecture is evidently slower in the case of multiple master-slave data transactions. The prime bottleneck is the single shared medium and latency due to arbitration among many master interfaces (Fig. 15.1). Therefore, the first approach to design a scalable on-chip interconnect was adaption of crossbar topology. A crossbar is a matrix switch fabric that connects all the inputs with all the outputs enabling multiple communication connections (Fig. 15.1). The idea has been borrowed from the telecommunication industry where such architectures have been successfully used for four decades in telephony applications [82].

The same concept of multiple communication channels was implemented in the SoC design industry by combining multiple shared buses to form an all-input-all-output connection matrix. The concept is also known as hierarchal bus or multilayer bus. STBus [53] is perhaps the best-known commercial bus architecture that inherently supports crossbar architectures. A design methodology for AMBA-based cascaded bus architecture is provided by Yoo [104]. Yoo et al. have experimented with integrating 90 IP blocks in a single crossbar-based SoC design. Similarly, authors in [72] have provided a complete design methodology for designing an application-specific crossbar architecture using the STBus protocol. They claim significant performance improvements over standard single-bus architectures. The most interesting crossbar implementation is the interconnect system for IBM Cyclops64 architecture. Each Cyclops64 crossbar connects 80 custom processors and about 160 memory banks. With single transaction latency of seven cycles and bandwidth comparable to state-of-the-art NoC architecture, Cyclops64 interconnects is perhaps the most advanced practical crossbar design for the SoC domain.

Researchers have been arguing over the scalability of crossbar-based interconnect architecture due to the nonlinear relation between the number of the ports and latency and wire cost [107]. However, recent experiments from [82] show that a 128×128 port crossbar in a 90 nm technology is feasible. They have benchmarked their crossbar design against state-of-the-art mesh-based NoC design and concluded that the crossbar design matched NoC architectures in terms of latency, bandwidth, and power consumption. However, the design complexity is prohibitively high due to complex wire layouts.

15.1.3 Network-on-Chip Interconnect

Following Moore's law of available on-chip transistor resources, we are looking beyond having thousands of cores on a single chip. It has been predicted that the performance of such kilo-core Multi-Processor System-on-Chip (MPSoC) will be communication architecture dependent [16]. Traditional bus-based architectures cannot scale beyond a few tens of IP blocks, and there is a need to provide a more scalable and protocol invariant communication architecture.

The solution to the scalability problem of bus-based architectures was found in the form of network-on-chip architectures [29, 58]. NoC inherently supports the general trend of highly integrated SoC design and provides a new de facto standard of on-chip communication design. The basic idea of NoC has been adapted from the well established structure of computer networks. In particular, the layered service architecture of computer networks has been well adapted in NoC to provide a scalable solution. In a NoC architecture, data is converted into packets, and these packets traverse number of hops (switches or routers) based on a predefined routing technique. The key advantages of using NoC as the on-chip interconnect are:

- NoCs inherently support multiple communication paths through a combination of physically distributed routers and links, which greatly increases the available on-chip data bandwidth. This enables different cores to exchange data in parallel without any central arbitration. This makes NoC an ideal candidate interconnect for supporting increasing communication needs of multi-core chips with tens and hundreds of cores. Multiple communication paths between given source and destination cores give NoC an inherent fault tolerance. In case of permanent error in the router or link on a given path, data can be rerouted through an alternate path between source and destination cores.

- NoC architectures use short electric wires that have highly predictable electric properties. Compared to bus interconnect, smaller drive transistors are required to switch short wires between routers. This helps to improve the *energy/bit* metric of interconnects. Moreover, due to shorter wire delays, NoCs can be switched at higher frequencies than buses and crossbars without any significant increase in power. Deep sub-micron semiconductor manufacturing introduces integrity issues in wires. Having shorter wires reduces the probability of manufacturing faults and hence improve the production yield. The predictable electric properties of short wires also help in reducing design verification cost.
- NoCs follow a modular design paradigm by allowing reuse of existing hardware Intellectual Property (IP) blocks. For most designs, NoCs can be easily scaled for different number of cores and applications by simply instantiating multiple copies of existing designed and verified router IPs. This reduces the overall complexity of the chip design process.
- NoCs provide a clear boundary between computation and communication. On-chip components (memory controllers, processing cores, hardware IPs, etc.) can have different communication protocols (AXI, AHB, etc.) which are converted to a standard packet format through the help of protocol convertors. Therefore, data communication between on-chip components is agnostic of communication protocol used by different components. This is a useful feature for designing heterogeneous SoCs with hardware components selected from different IP vendors.

15.2 Defining Features of Network on Chip

NoC-based MPSoC designs have attracted attention of researchers for the last decade. The defining features of NoC design are router design, routing algorithms, buffer sizing, flow control, and network topology. We will discuss them in more detail.

15.2.1 Topology

A NoC consists of routers and communication channels. NoC topology defines the physical layout of the routers and how these routers are connected with each other using the communication channels. The selection of NoC topology can have a significant effect on multi-core performance, power budget, reliability, and overall design complexity. For example, if the average number of hops between nodes is high, packets have to travel longer and hence network latency will be high. Similarly, if a topology requires very long physical links, designers have to make an effort to ensure timing closure for longer links. Moreover, topologies that allow diverse paths between nodes can potentially provide higher bandwidth and also prove to be more reliable in the case of faulty links. Therefore, when designing NoC-based multi-core systems, the first decision is to choose the NoC topology [83].

NoC topologies can be classified as *direct* and *indirect* topologies [83]. In direct topologies, each on-chip component such as core or memory, is connected with a router, and therefore each router is used to inject or eject traffic from the network. In indirect topologies, the on-chip components are connected only to *terminal* routers, whereas the other routers only forward the traffic [83]. The *degree* parameter of a router defines the number of neighboring routers and on-chip component to which the router has links. The degree parameter defines the number of input/output ports in each router. Note that the complexity of router microarchitecture increases with an increase in the degree of router.

Figure 15.2 shows three commonly used direct topologies *ring*, *mesh*, and *torus*. The ring is the simplest topology to implement in terms of silicon area and design complexity. The degree of each router in ring interconnect is 3 (two neighbor router + one local resource (core, memory, etc.)). The drawback of ring topology is performance scalability. The number of hops between two nodes in worst-case scenarios is proportional to N: the number of nodes in the topology. Furthermore, rings provide limited bandwidth and are less reliable due to poor path diversity. Therefore, rings become impractical for multi-core chips with more than 8–16 nodes [8].

Mesh and torus topologies solve the scalability problems of ring topology, albeit at a cost of higher degree routers and possibly more complex VLSI layout. Each router in a mesh topology has a degree of 5, except for the routers on the border. Torus can be classified as an enhanced form of mesh with wrap around links between border routers. These links use router ports that are not required to implement mesh topology. The wraparound links in torus reduce the average number of hops and

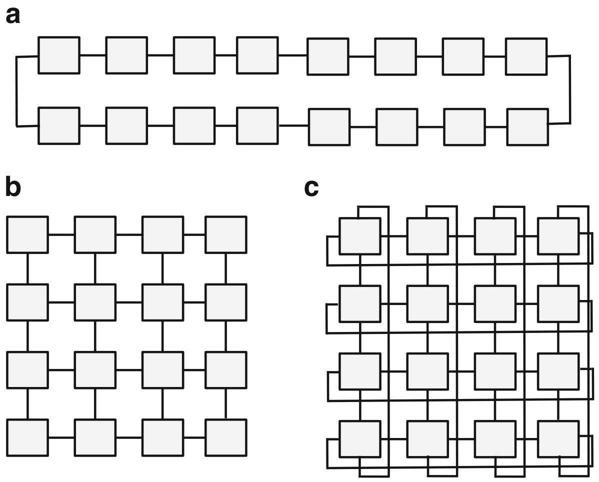

Fig. 15.2 Well known direct topologies. (**a**) Ring. (**b**) Mesh. (**c**) Torus

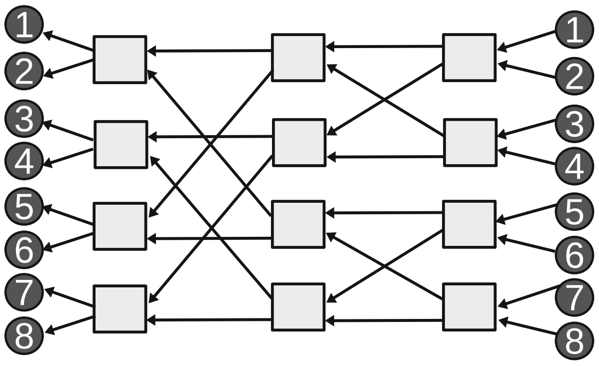

Fig. 15.3 2-ary 3-fly Butterfly topology

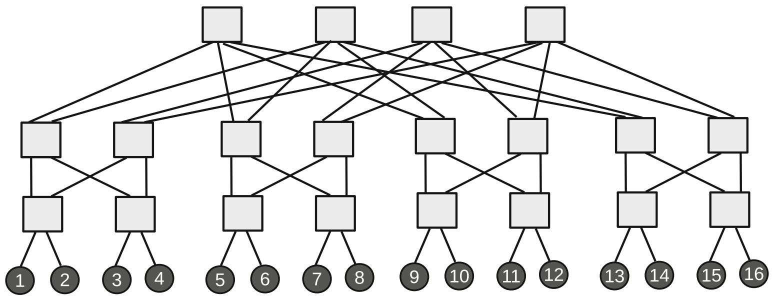

Fig. 15.4 3 Tier Fat Tree topology

provide better bisection bandwidth. The worst-case number of hops is $\sqrt{N} + 1$ for torus and $2\sqrt{N} - 1$ for mesh. In mesh, all communication links are short and equal in size. However, for torus wraparound links are considerably longer and need special attention for timing closure.

Two classical examples of indirect networks, *butterfly* and *fat tree* are shown in Figs. 15.3 and 15.4, respectively. The important feature of butterfly topology is that the hop distance between any source-destination node pair is fixed (three in the topology shown in Fig. 15.3). The router has degree 2 (two input and two output ports), resulting in low-cost routers. However, the number of routers is greater than the number of SoC components. The two main disadvantages of butterfly topologies are single communication path between a given source-destination pair resulting in low bandwidth and low link fault tolerance and more complex wire layout due to uneven link lengths. The fat tree topology provides higher bandwidth and excellent diversity of possible routing paths. However, these qualities come at a cost of silicon area (more routers) and complex wire layout.

In addition to these regular topologies, application-specific MPSoCs are often designed on top of NoCs with customized topologies [20]. The data traffic patterns are often known at design time, and therefore a communication graph can be extracted from the application specifications [12, 89]. The communication graph combined with knowledge of the physical mapping of SoC components can be used to create a topology that meets certain performance, energy, or area constraints [13, 20, 95].

15.2.2 Routing

The routing algorithm defines the sequence of routers between source and destination nodes that a data packet will traverse. The quality of the routing algorithm is determined by the average packet latency and power consumption. A good routing algorithm evenly distributes the traffic across all the routers and maximizes the saturation throughput of the network. Power can be optimized by keeping the routing circuit simple and keeping the number of hops traveled by data packets low [83].

Deterministic routing is the simplest routing scheme and is widely used in NoCs. For a given source and destination pair, data packets always travel through a predefined set of routers. Dimension Ordered Routing (DOR) for mesh is a common example of deterministic routing. In XY routing for mesh, depending on the physical location of the source and destination pair, the packet always travels first in the X (horizontal) direction and then in the Y (vertical) direction. However, deterministic routing can cause traffic hotspots in the case of an asymmetrical communication pattern between nodes [63, 83].

Oblivious routing is superior to deterministic routing in terms of path selection. For a given source-destination pair, oblivious routing can select one of many possible routes. However, this decision is taken without any knowledge of the network's current traffic condition. One example for oblivious routing is ROMM [76]. In the ROMM routing scheme for mesh, an intermediate node is selected at random on minimal paths between source and destination, and the data packet is first sent to the intermediate node and from there to the destination using a deterministic routing algorithm.

Adaptive routing is the most robust routing scheme, as it uses global knowledge about the current network traffic state to select the optimal path [63]. Adaptive routing distributes the traffic node across different network routers and hence maximally utilize the network bandwidth. However, implementing adaptive routing increases the design complexity of the routers [83]. Moreover, there is always a limitation on how much global knowledge can be forwarded to each router, hence limiting the effectiveness of the routing scheme [63].

In addition to standard routing schemes, MPSoC designers often use application-specific routing schemes for NoCs [22, 28, 79]. Application communication graphs can be analyzed to extract information about data volume and criticality. This information can be used to design routing algorithms that minimize the communication latency [79].

15.2.3 Flow Control

Flow control determines how data is transferred between different routers in a NoC. Specifically, flow control dictates the buffer and link allocation schemes. The design objective for flow control architecture is to minimize the buffer size and hence silicon area and power of routers and to keep the network latency low. In packet-switched NoCs, a data message is broken into a predefined *packet* format. A network packet size can be further broken and serialized into multiple *flits*. The size of flit normally equals physical channel width [83]. Additional information is added to each flit to indicate *header*, *body*, and tail flit. The routing and other control information can either be added only to the header flit or it can be added to each flit depending on implementation.

In *store-and-forward* flow control [30], before forwarding the packet to the next node, the router waits until the whole packet has been transmitted into its local buffer. This means that the input buffer must have enough space to store the whole packet, which can increase router area and power consumption. This scheme also increases the communication latency, as packets spend a long time at each node just waiting for buffering, although the output port might be free.

Virtual cut-through [54] improves on store-and-forward flow control by allowing a packet to be routed to the next router even before the whole packet arrives at the current router. However, the packet is only forwarded if the downstream router has enough buffer space to store the complete packet. This means that buffer size remains the same as in the case of store-and-forward flow control with improvement in per-hop latency.

Wormhole routing [90] is a more robust scheme, as it allocates buffer space at the granularity of flit, opposed to the virtual cut-through and store-and-forward scheme which allocates buffers at the granularity of packet. As soon as flit of a packet arrives at an input port, it can be forwarded even if only one flit space is available in the input port of the next router (and output channel is not allocated). The wormhole flow control scheme results in low-area routers, and it is therefore widely used in most on-chip networks [83]. The term *wormhole* implies that a single packet can span multiple routers at the same time. The main downside of this scheme is that the multiple links can be blocked at the same time in case the header flit of a multiple flit packet is blocked in one of the routers on the communication path.

15.2.4 Router Microarchitecture

The key building block of NoC is the router. The router's microarchitecture dictates the silicon area, the power, and, most importantly, the performance of the NoC. The maximum frequency at which a router can operate depends on the complexity of the logic used in the router microarchitecture, which in turn translates into higher-level performance metrics such as network latency and maximum achievable bandwidth. The complexity of the router's microarchitecture depends on the network topology (degree), flow control method, and routing algorithms. For

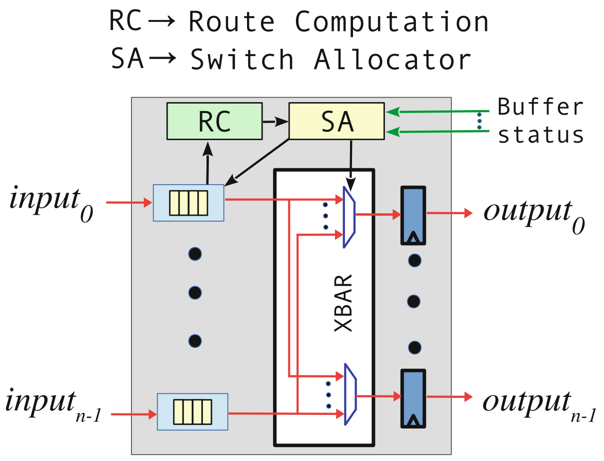

Fig. 15.5 Wormhole router architecture

example, a complex adaptive routing algorithm can be used to improve the worst-case network bandwidth, but it will result in increased area and power.

Figure 15.5 shows the overall architecture of a packet switch wormhole router [38]. The basic building blocks of a wormhole router are *input buffer*, *route computation*, *switch allocators*, and *crossbar switch*. Input buffers store flits when they arrive in the router and keep them stored until they are forwarded to the next router. The route computation unit calculates the output port for the header flit stored at the head of each input buffer, depending on the routing algorithm. The switch allocator arbitrates between different packets contending for the same output ports. The crossbar switch is logically a set of muxes that route flits from the input port to the output port. The data from the crossbar is stored in output buffers which can be as simple as flip-flops. The input buffers also propagate the buffer occupancy status to neighboring routers to implement flow control [38].

15.2.4.1 Progress of a Packet in a Wormhole Router

The incoming flit is first stored in the input buffer (Buffer Write (*BW*) stage). If the flit at the front of the input buffer is the header flit, the route computation unit calculates the output port required to send the packet to its destination and asserts the port request signal to the switch allocator (route computation (*RC*) stage). In modern routers, the switch allocator consists of multiple port arbiters, one for each output port. Each output arbiter chooses one of the multiple input requests for a given output port. When an output port is allocated to an input port, the specific output port is locked until the whole packet (tail flit) has crossed the crossbar switch. Before doing anything further, the switch allocator checks if the downstream router has enough space to store the outgoing flit. If the buffers are full at the downstream routers, the switch allocator blocks the packet transmission. However, if buffer space is available, switch allocator sets the proper select lines for the crossbar switch and also instructs the input buffer to remove the flit at the front. This whole process is called the switch allocation (*SA*) stage. On getting a valid signal and output port selection from the switch allocator, the crossbar switch transfers the flit from the input port to the output port (switch traversal *ST* stage). The flit from the output port of the router then travels over wire links to get latched in the input buffer of the downstream router (link traversal (*LT*) stage). Note that the header flit of a packet goes through all stages discussed here. The body and tail flit, however, skip the RC and SA stages, as the output had already been reserved by the header flit.

15.2.4.2 Optimization and Logic Synthesis of Routers

Executing all router stages in a single cycle can be achieved at a lower frequency because the cumulative logic delay of stages can be long. Single-cycle operation might require a higher supply voltage depending on the target frequency which can increase the power consumption. Therefore, most of the high-performance routers are often pipelined [30, 38]. However, increasing the number of pipeline stages increases the per-hop latency and hence the overall network latency. The number of pipeline stages also depends on the sophistication of the RC, SA, and ST stages.

In commonly used pipelined routers, the LT and BW are done in one cycle, and RC, SA, and ST are executed in the next cycle. However in the case of more complex router architectures, the second pipeline stage can be further divided into RC+SA and ST pipeline stages. The pipeline stages can affect the area and power consumption [11]. Using fewer pipeline stages results in more stringent latency constraints for logic synthesis, and hence the synthesis tool has to insert larger gates with lower delays. Larger logic gates have higher dynamic and leakage power. Pipeline stages on the other hand can reduce the gate sizes; however, the overall area may increase due to addition of the pipeline flip-flops [11]. Therefore, the logic synthesis of routers is a classic power-performance-area tradeoff problem [11, 38, 81].

15.2.5 Network Interface

Router is the main building block of NoC and carries the burden of routing the packets across network. *Network Interface (NI)* on the other hand acts as a *bridge* between hardware IP blocks and NoC. Figure 15.6 shows an example of NI signaling scheme. NI converts IP block's communication protocol to NoC's packet format and performs associated housekeeping operations. The communication protocol can vary across IP vendors. For example, most of the ARM IPs normally support AXI protocol [6], whereas Xtensa processors use PIF protocol [5]. Therefore, NIs actually enable the modular property of NoC by letting different IP communicate seamlessly irrespective of their communication protocol. An example MPSoC that contains IPs from different vendors is shown in Fig. 15.7. To enable maximum flexibility to system designers, commercial NoC vendors provide support for multiple communication protocols. For example, Sonics [4] supports AXI and OCP protocols, and Arteris [7] supports AMBA, PIF, BVCI, and OCP protocols.

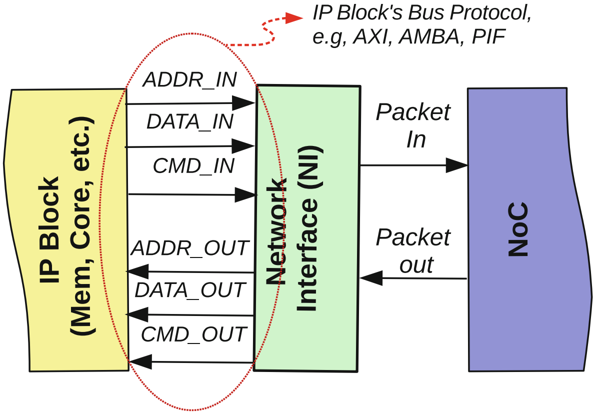

Fig. 15.6 Network interface (NI)

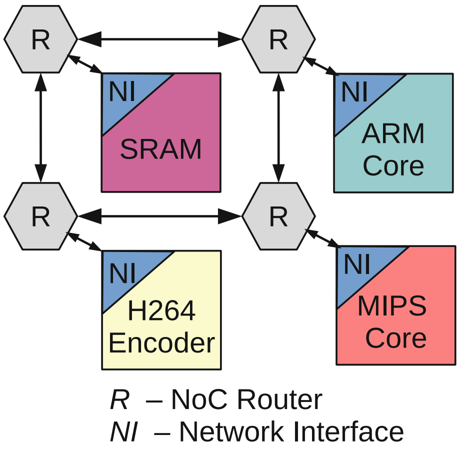

Fig. 15.7 Heterogeneous MPSoC with IPs from different vendors

Microarchitecture of the NI depends on the programming model selected for SoC [36]. For example, Tilera iMesh on-chip interconnect network [103] supports both MPI [62] and shared memory [105] programming models. iMesh enables parallel operation of these programming models by integrating separate NIs and NoCs for both of these programming models.

Researchers have proposed integrating various advance features in NI design with primary focus on quality-of-service (QoS) services. Radulescu et al. [85] proposed a NI design that integrates QoS services for shared memory MPSoCs, with support for both guaranteed and best-effort services. Mishra et al. [70] proposed keeping track of vital application information such as cache miss rate and executed instructions in NI to support fair QoS among heterogeneous applications. Similarly Chen et al. [25] monitor application cache miss rate and other processing core's information in NI logic to implement NoC power optimization.

15.2.6 Performance Metrics

As described earlier in chapter, there are number of possible customizable features in a NoC. Therefore, it is important to discuss metrics that are often used to assess the NoC's performance.

NoC performance is often evaluated using network *latency* and *throughput* [30]. An example latency versus traffic injection rate is shown in Fig. 15.8. The *latency* is defined as average time it takes for packets to travel between source-destination pairs in NoC. Network latency can calculated as sum of latency experienced at each hop (router). The *zero-load latency* metric defines the lower bound on latency when there are no other traffic in the network. However, as traffic is introduced in the NoC at a higher rate, packets travel slower in the network due to channel contention. The *saturation throughput* point is defined as injection rate at which packet latency becomes prohibitively large. Some research also define *saturation throughput* as the injection rate at which the average network latency is roughly three times the zero-load latency [50]. As a rule of thumb, designers aim to minimize the zero-load latency and maximize the saturation throughput. In the absence of real applications, *latency* and *throughput* are commonly used to evaluate different NoC architecture using different synthetic traffic patterns [64].

Although latency and throughput are easier to evaluate, researchers argue that these metrics may be misleading for assessing impact of NoC on overall system performance. Mishra et al. [70] showed that some applications are network latency sensitive, whereas some applications are bandwidth sensitive. Therefore, using application-level metrics such as execution time and average memory access latency is a better way to evaluate NoC performance. Similarly Chen et al. [25] showed that the impact of NoC latency on application's performance depends on other

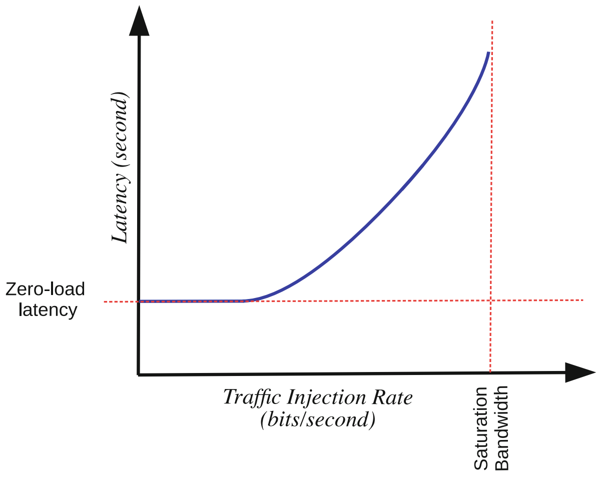

Fig. 15.8 Latency and throughput metrics for NoC

application-level metrics such as cache miss rate. This is the reason that recent works on NoC power optimization consider application-level metrics instead of network-level metrics [23, 25, 44].

15.3 Overview of Recent Academic and Commercial NoCs

Tilera iMesh on-chip interconnect network [103] is an example of NoC designed for homogeneous multi-core chips. The iMesh interconnect architecture has been used in the commercial TilePro64 multi-core chip. The latest 72-core Tilera GX chip [2] also uses the same NoC design. The proposed NoC architecture is different from other academic and commercially available on-chip architectures in terms of the number of physical NoC. iMesh provides five different physical NoC channels: two of these networks are used for memory access and management tasks, while the rest are user accessible. The motivation is that future integrated circuits will have enough silicon resources to integrate more than one NoC per chip.

The next-generation SPARC M7 processor combines three different NoC topologies for the on-chip interconnect [84], ring-based request network, point-to-point network for response messages, and mesh for data packets. The M7 processor uses a shared distributed cache model. The request ring network is used to broadcast the cache requests to all cores in the system. The point-to-point network is only used when adjacent cores share data. The mesh data network is built using ten port routers and is used primarily to stream data from memory controllers to local cache.

Anton2 is a special-purpose supercomputer for performing molecular dynamic simulations. The building block of the supercomputer is Anton ASIC which contains a number of special-purpose hardware units for numerical calculations [101]. Anton2 ASIC uses a 4×4 2D mesh for connecting on-chip components, whereas the chips are connected with each other using a 3D torus. The novel feature of the communication architecture is that the same set of routers are used for both intra- and inter-chip communication. About 10% of the total ASIC area is dedicated to on-chip communication.

SMART (single-cycle multi-hop asynchronous repeated traversal) NoC from MIT is another interesting low-latency on-chip communication architecture [56]. Authors observed that a data bit can travel 9–11 hops in a single clock cycle at 1 GHz for 45 nm silicon technology. Based on this observation, they propose a NoC architecture where data can be transferred across physically distant mesh nodes in a single cycle by bypassing multiple routers. This reduces the latency of multi-hop mesh NoC. The proposed architecture is reported to improve the performance of PARSEC and SPLASH-2 benchmarks by an average of 26% for private L2 cache and 49% for shared L2 cache architectures.

Æthereal [42] is probably one of the best-known academic NoC architecture. Even at the time of conception, Æthereal supported different IP communication protocols such as AXI and OCP. In addition to baseline best-effort NoC services, Æthereal also supports building predictable on-chip communication though time-division multiplexed circuits [43]. This enables building SoCs for system with hard real-time performance requirements such as braking system in an automobile. A more comprehensive discussion on real-time NoCs is presented in ▶ Chap. 16, "NoC-Based Multiprocessor Architecture for Mixed-Time-Criticality Applications".

Intel used a mesh-based NoC for an 80-core TeraFlop experimental chip [46]. The mesh NoC uses five stage pipelined routers designed for 5 GHz frequency. This results in 1 ns per-hop latency. According to experiments conducted on the research chip, the NoC consumed about 28% of total chip power although it consumed 17% of total chip area.

Intel has also introduced a 48-core mesh NoC-based multi-core chip called single-chip cloud computer (SCC) [47]. The target frequency for the NoC was set at 2 GHz with 1.1 V supply voltage. The router is four-stage pipelined and uses virtual cut-through flow control. To mitigate the problem with higher NoC power from the previous 80-core chip, Intel opted for a DVFS scheme for NoC. The NoC was organized as a 6×6 mesh so that two compute cores share a single router. These techniques helped to reduce the share of NoC power to 10%.

There are two well-known commercial NoC IP providers, Sonics [4] and Arteris [7]. Both Sonics and Arteris provide various predesigned NoC IPs for commonly used processor IPs such as ARM and Xtensa. Furthermore, both of them provide design tools that can be used to optimize the NoC IP for various design objectives such as power, area, performance, quality of service (QoS), and reliability. It is anticipated that hardware developers will be using third-party NoCs to reduce both design cost and development time [20].

15.4 NoC Power Optimization

As more cores are integrated on a chip, the on-chip interconnect's complexity increases and so does its power. Computer architects always want to keep the on-chip interconnect's power low so that processing elements and memory hierarchy can use a larger share of power [17]. Although NoC provides the scalable communication for multi-core chips, it can consume valuable power. For example, Daya et al. [35] report that NoC in their 36-core SCORPIO experimental chip consumes 19% of the total chip power. Similarly NoC consumes 28% of total power in Intel's TeraFlop chip [46]. With limited power budgets constraining multi-core scaling [39], employing power-saving techniques for NoCs is an active research topic.

Over the years, various voltage-scaling-based solutions have been investigated for reducing NoC dynamic power. Shang et al. [92] presented the pioneering work on Dynamic Voltage and Frequency Scaling (DVFS) for on-chip links. The scheme uses usage history of links to predict their future utilization. Based on this prediction, a certain voltage and frequency (VF) level is selected for each link. Bogdan et al. [14] proposed a DVFS scheme for spatially and temporally varying workloads. The proposed DVFS scheme worked at the granularity of individual routers. Based on fractal modeling of workloads, the scheme selected an appropriate

VF level for each router. Mishra et al. [69] proposed per-router frequency tuning in response to changes in network workload to manage power and performance. Based on the optimization goal, the frequency of the routers is increased to relieve network congestions and improve performance, or the frequency is decreased to meet certain network power constraints. Ogras et al. [78] described a framework for fine-grain VF selection for VF-island-based NoCs. The scheme first statically allocates VF level to each NoC router and then uses the run-time network information for fine-tuning the assigned VF level. All these works base their DVFS schemes on network metrics such as packet latency, injection rate, queue occupancy, and network throughput. Researchers [26, 44] argue that by neglecting higher-level performance metrics such as application execution time, these schemes can result in nonoptimal results. Therefore, work by Chen et al. [44] and Hesse and Enright [44] based their DVFS schemes on the actual execution delay faced by applications due to DVFS for NoCs. Zhan et al. [106] explored the problem of per-router DVFS schemes for streaming applications. They developed an analytical model to statically predict the effect of VF scaling on application throughput. Depending on application to core mapping, routers on the critical communication path are operated at lower frequency.

Since the ratio of leakage power to total chip power is increasing with transistor scaling, researchers have been exploring leakage power techniques. Through experiments with realistic benchmarks, Chen [23] reported that the router's static power share is 17.9% at 65 nm, 35.4% at 45 nm, and 47.7% at 32 nm. As discussed earlier in this chapter, power gating is often used to save the static power of on-chip components [48]. Soteriou and Peh [94] proposed power gating at the link level. They used the channel utilization history to select links that can be switched off with minimal impact on performance. As switching off some links results in irregular topologies, they proposed an adaptive routing algorithm to avoid the switched off links. Matsutani et al. [66] proposed adding low-frequency virtual channel buffers that can be power gated at run time to save leakage power. An ultra fine-grain power gating for NoC routers is proposed in [65]. They used special cell libraries where each Complementary Metal-Oxide-Semiconductor (CMOS) gate has a power gate signal. Therefore, the power for each router component such as input buffer, switch allocator, and crossbar can be individually turned on or off, based on the network activity. This helped to save leakage power by 78.9%. However, the main drawback of this technique is the complexity of including special power gate circuitry in each CMOS gate. The additional power circuitry also increases the router area. Kim et al. [55] proposed a fine-grain buffer management scheme for NoC routers. In the scheme, depending on network traffic load, the size of the input buffers was increased or decreased at run time. The unused buffer slots were power gated to save leakage power. However, this scheme only targeted buffer leakage power and neglected other router components. Parikh et al. [80] proposed performing power gating at the granularity of the *data-path segment* which consists of an input buffer, crossbar mux, and output link. In the router parking [88] scheme, routers for cores that are not under use are switched off, and the traffic is routed around the switched off routers through new routing paths that are calculated at run time.

The main problem with the previous proposals for power gating is the performance overhead due to wakeup latency. Chen and Pinkston [24] proposed overlaying a standard mesh NoC with a low-cost ring network. If a packet arrives at a router and the router is switched off, the packet is routed through the ring network which is always switched on. Das et al. [33] proposed to divide wider mesh NoCs into four smaller NoCs for efficiency. Additional NoCs are only switched on if the network load exceeds a certain limit and the unused NoC planes remain switched off. In this network connectivity is ensured.

As buffers in routers consume a considerable share of dynamic and leakage power, researchers have also explored using bufferless NoCs. The first effort in this direction was the BLESS router [71]. The idea is that instead of storing the flit in router, the flit is routed even in a direction that does not result in a minimal path. This means that even with some misrouting, the flit will eventually reach its destination. Therefore this scheme is applicable for routers with the number of output ports equal or higher than the number of input ports. The deadlock and livelock condition is avoided by allocating a channel to the oldest flit first. To improve some of the shortcomings of BLESS router, CHIPPER [40] was introduced. Both BLESS and CHIPPER show potential in reducing network power. However there are two concerns with bufferless NoCs. First, the deflection routing causes packets to take non-minimal routes resulting in longer packet delays. The effect of network delays is more significant for memory-intensive applications. Secondly, the bufferless routing results in out-of-order arrival of the flits which increases the complexity of the NI.

15.5 Communication-Aware Mapping

While designing an MPSoC-based system, it is not uncommon to have fixed specifications for the underlying hardware. Therefore, in such cases, it is the responsibility of the system programmer to use the available hardware resources efficiently. The situation is even more complex in the case of heterogeneous MPSoC architecture with NoC interconnect. Over the years, many techniques have been proposed for effectively mapping a given application on a target MPSoC architecture. Studying the mapping problem for NoC-based MPSoC architectures has been a key area of research. The unique data communication capabilities of NoC-based architectures demand a smart communication-aware mapping strategy [19, 28, 59, 73]. It is important to analyze both NoC's architectural properties and run-time network contention while estimating the run time of a given application. Embedded system designers are expected to tweak the application mapping to maximize the performance while meeting the hard time-to-market limits [99, 100]. There is no single technique to map applications on a given MPSoC. The mapping problem is further complicated due to the presence of many different MPSoC architectures. Therefore, the common practice is to rebuild the mapping strategy for every application-architecture combination [68, 74, 75, 97].

Application mapping was an area of interest for researchers working in parallel computing and supercomputing [15]. Their idea was to map the applications that share the data as close as possible to reduce the network latency. This naive idea is also applicable in the case of CMPs where mapping the application has to take into account the underlying on-chip interconnect architecture. Similarly, the choice of shared memory architecture and message passing interface heavily influences the mapping algorithms.

The mapping solutions proposed in the literature can be broadly divided into two categories (1) static mapping and (2) dynamic run-time mapping. Each class of mapping algorithm has its advantages and disadvantages. Static mapping is useful when applications to be executed are known at design time [18, 77, 98, 108]. System designers can formulate an optimal or near-optimal solution for such scenarios. On the other hand, run-time mapping algorithms allocate resources to application on the fly. These algorithms give a better mapping solution in cases where the application behavior is unknown at design time or system characteristics such as network congestion can change rapidly [9, 10, 45].

In the case of embedded systems, applications to be executed are known at design time. Therefore, static mapping techniques result in better system performance [86]. Unfortunately, most of the mapping problems prove to be NP-hard problems with only near-optimal solution [93]. However, heuristic-based algorithms have also been proposed, which reduce the time required to solve these NP-hard problems [93, 97]. TDMA-based NoCs are used for predictable systems. The expected traffic for a given link can be estimated using analytical modeling or simulations. Through proposed algorithms, it is then possible to map a given processor-processor communication in an allotted time slot. This not only improves the network congestion but also makes the system predictable [102, 109].

Some researchers have provided solutions where the mapping and partitioning problem of NoC-based embedded systems are dealt with as a combined problem. Although these solutions require a very strict analytical model for application, the final software and hardware are highly optimized in area footprint, power, and performance [59, 60]. It has been observed that a generalized MPSoC architecture might eventually fail to cover the design requirements of a particular real-time embedded system. Therefore, the option of highly customized hardware for a particular set of applications has also been widely explored.

15.6 Application Specific Communication Architecture for MPSoCs

Over the years, researchers have analyzed that flat communication and memory architectures are not sufficient for designing high-performance application-specific MPSoC architectures. Moreover, the quest for low-power, low-cost embedded system has forced designers to optimize the system. Therefore, it is anticipated that highly optimized hardware and software designs will include customized memory and on-chip network designs in the future [99].

Customizing the communication architectures at system level and gate level not only improves the system performance; it also results in energy-efficient design. By eliminating the logic overhead, the leakage and static power loss of the system can be significantly reduced [72]. It has been shown that some MPSoC applications actually do not need highly complex network architectures like NoC, and simple bus-based architecture can essentially meet the performance requirements. Therefore, it is important to avoid overkill by intelligently designing communication architectures that give optimal solutions without incurring area overhead [51].

The idea of application-specific communication architectures is more interesting in the case of NoC. Inherently, NoC architectures are highly customizable, and designers can tweak a variety of parameters of the architecture to meet performance, cost, and energy constraints. ×-pipes is perhaps the first project that provided a library-based approach to NoC design [96]. The selection of optimal on-chip communication architectures is nontrivial. Therefore, an effort has been made to introduce library-oriented design space exploration where the designer has several configurations to choose from. Designers can then use analytical modeling or simulations to select the best communication architectures [49]. The same idea has also been explored by Jeremy and Parameswaran [21] by developing a library of NoC components with the help of analytical power and performance modeling. Another interesting project that was targeted toward low-power NoC-based embedded MPSoC was AEthereal network on chip [42]. The significance of these projects is the flexibility provided to the designer to quickly explore the various possible optimizations for a given application. These optimizations can provide up to 12 times improvement in performance while reducing the area cost by 25 times when compared with flat communication architectures [51].

Bertozzi et al. [13] have proposed a NoC synthesis tools called *NetChip* based on ×-pipes [96] NoC library. Figure 15.9 shows an example on how application specifications are translated into NoC architecture. The first step is collect application and map it on multiple components (cores, memory, etc.). The next step is extract data bandwidth requirement from the application to core mapping. Based on communication requirement, tool automatically explores different NoC topologies and other associated architecture parameters such as routing scheme. In the final step, the tool automatically generates SystemC models for selected NoC components for simulations and synthesis.

Similarly, the option to integrate different communication architectures in a single design has also been studied extensively. Murali et al. [72] proposed a complete design method for application-specific crossbar synthesis. They use the traffic pattern of the application to design a communication architecture that is a combination of packet-oriented crossbars and bus. Bus-based architectures are simple but offer less performance. The crossbar-based architectures are more complex but provide better throughput. Therefore, Murali et al. combined the two communication architectures while keeping the gate cost low and fulfilling the performance requirements.

All the techniques referred to above improve the performance of on-chip communication in terms of aggregate on-chip bandwidth and average data latency, which are not suitable metrics for streaming applications executing on MPSoCs. In

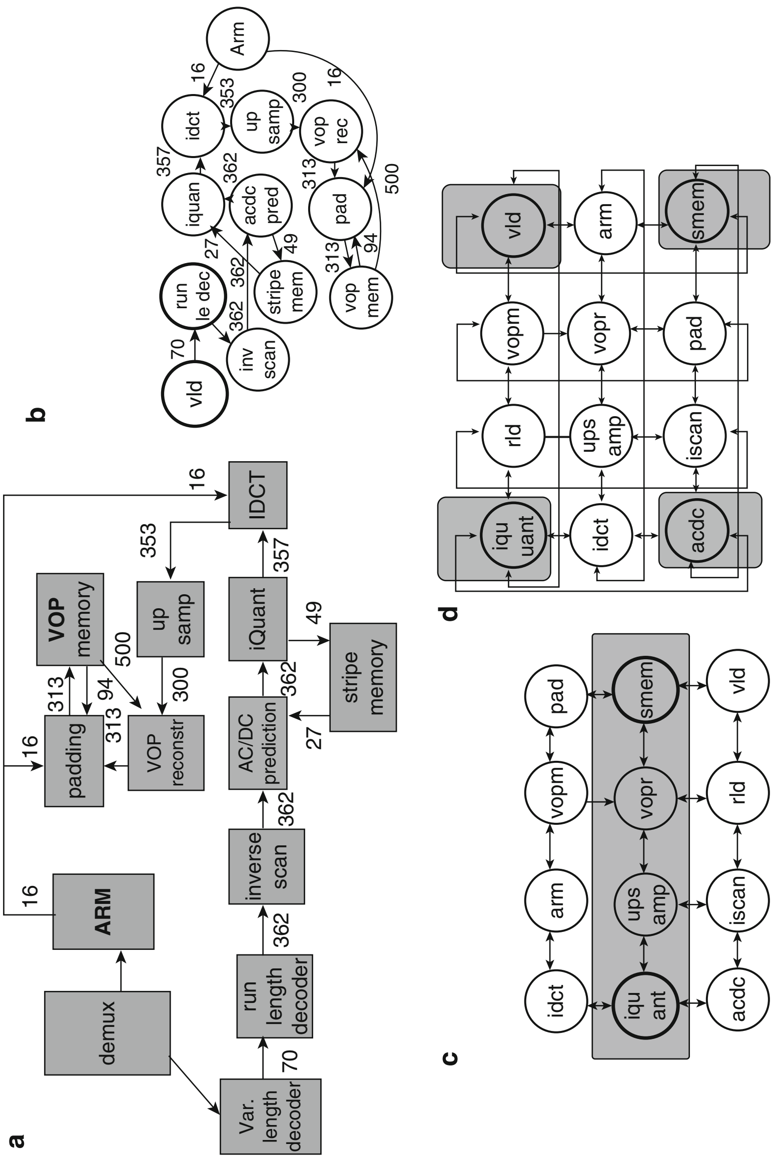

Fig. 15.9 Application-specific NoC design example from [13] for VOPD application. **(a)** VOPD block diagram, **(b)** VOPD graph with bandwidth requirements mentioned in mb/s, **(c)** mesh mapping, and **(d)** torus mapping

the case of streaming MPSoCs, application-level throughput or latency of the system is the most important performance metric [52]. Thus, it is important to incorporate system-level performance constraints in the design flow for an application-specific on-chip network [12].

15.7 Conclusion

In this chapter we introduced readers to some basic concepts of NoC architecture and motivated the use of NoC architecture for large-scale SoC designs. We presented examples of NoC designs from commercial chips and academia to show the current trends in NoC design. Given the increasing interest in reducing power consumption of SoC components, we presented various power optimization techniques proposed over the recent years. We then discussed some of the recent research work on optimizing performance using intelligent application mapping on MPSoCs. In the end, we explored how on-chip interconnect can be designed for application-specific MPSoCs.

References

1. ARM AMBA Interconnect Specification. http://www.arm.com/products/system-ip/amba-specifications.php
2. EZChip TileGX Multicore Architecture. http://www.tilera.com/products/?ezchip=585&spage=614
3. IBM CoreConnect Bus Technology. http://www.xilinx.com/products/intellectual-property/dr_pcentral_coreconnect.html
4. Sonics. http://www.sonicsinc.com/
5. Xtensa Processors. http://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable
6. AMBA AXI and ACE Protocol Specification (2013) http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022e/index.html
7. Arteris NoC (2015) http://www.arteris.com/
8. Aisopos K (2012) Fault tolerant architectures for on-chip networks. PhD Thesis, Princeton University
9. Al Faruque M, Krist R, Henkel J (2008) Adam: run-time agent-based distributed application mapping for on-chip communication. In: 45th ACM/IEEE design automation conference, DAC 2008, pp 760–765
10. Al Faruque MA, Krist R, Henkel J (2008) ADAM: run-time agent-based distributed application mapping for on-chip communication. In: Proceedings of the 45th IEEE/ACM design automation conference (DAC), pp 760–765. doi:10.1145/1391469.1391664
11. Becker DU (2012) Efficient microarchitecture for network-on-chip routers. Ph.D. thesis, Stanford University
12. Beraha R, Walter I, Cidon I, Kolodny A (2010) Leveraging application-level requirements in the design of a NoC for a 4g SoC – a case study. In: Design, automation test in Europe conference exhibition (DATE), pp 1408–1413. doi:10.1109/DATE.2010.5457033
13. Bertozzi D, Jalabert A, Murali S, Tamhankar R, Stergiou S, Benini L, De Micheli G (2005) NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. IEEE Trans Parallel Distrib Syst 16(2):113–129
14. Bogdan P, Marculescu R, Jain S, Gavila R (2012) An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly

variable workloads. In: 2012 sixth IEEE/ACM international symposium on networks on chip (NoCS), pp 35–42. doi:10.1109/NOCS.2012.32

15. Bokhari SH (1981) On the mapping problem. IEEE Trans Comput 30(3):207–214 doi:10.1109/TC.1981.1675756
16. Borkar S (2007) Thousand core chips: a technology perspective. In: Proceedings of the 44th design automation conference. ACM, pp 746–749
17. Borkar S, Chien AA (2011) The future of microprocessors. Commun ACM 54(5):67–77. doi:10.1145/1941487.1941507
18. Braun TD, Siegel HJ, Beck N, Bölöni LL, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen D et al (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib comput 61(6):810–837
19. Castrillon J, Tretter A, Leupers R, Ascheid G (2012) Communication-aware mapping of KPN applications onto heterogeneous MPSoCs. In: Proceedings of the 49th design automation conference, DAC'12. ACM, New York, pp 1266–1271. doi:10.1145/2228360.2228597
20. CHAN J (2007) Energy-aware synthesis for networks on chip architectures. Ph.D. thesis, UNSW
21. Chan J, Parameswaran S (2004) Nocgen: a template based reuse methodology for networks on chip architecture. In: 17th international conference on VLSI design, 2004. Proceedings. pp 717–720. doi:10.1109/ICVD.2004.1261011
22. Chao HL, Chen YR, Tung SY, Hsiung PA, Chen SJ (2012) Congestion-aware scheduling for NoC-based reconfigurable systems. In: Design, automation test in Europe conference exhibition (DATE), pp 1561–1566. doi:10.1109/DATE.2012.6176721
23. Chen L (2014) Design of low-power and resource-efficient on-chip networks. Ph.D. thesis, University of Southern California
24. Chen L, Pinkston TM (2012) Nord: node-router decoupling for effective power-gating of on-chip routers. In: Proceedings of the 45th annual IEEE/ACM international symposium on microarchitecture, MICRO'12. IEEE Computer Society, Washington, DC, pp 270–281. doi:10.1109/MICRO.2012.33
25. Chen X, Xu Z, Kim H, Gratz P, Hu J, Kishinevsky M, Ogras U (2012) In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches. In: 2012 sixth IEEE/ACM international symposium on networks on chip (NoCS), pp 43–50. doi:10.1109/NOCS.2012.12
26. Chen X, Xu Z, Kim H, Gratz PV, Hu J, Kishinevsky M, Ogras U, Ayoub R (2013) Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In: Proceedings of the 50th design automation conference, DAC'13. ACM, New York, pp 114:1–114:7. doi:10.1145/2463209.2488874
27. Cheng HY, Zhan J, Zhao J, Xie Y, Sampson J, Irwin MJ (2015) Core vs. uncore: the heart of darkness. In: 2015 52nd ACM/EDAC/IEEE design automation conference (DAC). IEEE, pp 1–6
28. Chou CL, Marculescu R (2008) Contention-aware application mapping for network-on-chip communication architectures. In: IEEE international conference on computer design, ICCD 2008, pp 164–169. doi:10.1109/ICCD.2008.4751856
29. Dally WJ, Towles B (2001) Route packets, not wires: on-chip interconnection networks. In: Design automation conference, 2001. Proceedings. IEEE, pp 684–689
30. Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. Elsevier
31. Das R (2010) Application-aware on-chip networks. Ph.D. thesis, The Pennsylvania State University
32. Das R, Ausavarungnirun R, Mutlu O, Kumar A, Azimi M (2013) Application-to-core mapping policies to reduce memory system interference in multi-core systems. In: 2013 IEEE 19th international symposium on high performance computer architecture (HPCA2013), pp 107–118. doi:10.1109/HPCA.2013.6522311

33. Das R, Narayanasamy S, Satpathy SK, Dreslinski RG (2013) Catnap: energy proportional multiple network-on-chip. In: Proceedings of the 40th annual international symposium on computer architecture, ISCA'13. ACM, New York, pp 320–331. doi:10.1145/2485922.2485950
34. Daveau JM, Ismail TB, Jerraya AA (1995) Synthesis of system-level communication by an allocation-based approach. In: Proceedings of the 8th international symposium on system synthesis. ACM, pp 150–155
35. Daya BK, Chen CHO, Subramanian S, Kwon WC, Park S, Krishna T, Holt J, Chandrakasan AP, Peh LS (2014) Scorpio: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering. In: 2014 ACM/IEEE 41st international symposium on computer architecture (ISCA). IEEE, pp 25–36
36. Jerraya AA, Wolf W (2005) Multiprocessor systems-on-chips. The Morgan Kaufmann series in systems on silicon. Morgan Kaufmann. ISBN:0-12385-251-X
37. Dennard RH, Gaensslen FH, Rideout VL, Bassous E, LeBlanc AR (1974) Design of Ion-implanted MOSFET's with very small physical dimensions. IEEE J Solid-State Circuits 9(5):256–268
38. Dimitrakopoulos G, Psarras A, Seitanidis I (2015) Microarchitecture of network-on-chip routers. Springer
39. Esmaeilzadeh H, Blem E, St.Amant R, Sankaralingam K, Burger D (2011) Dark silicon and the end of multicore scaling. In: 38th annual international symposium on computer architecture (ISCA), pp 365–376
40. Fallin C, Craik C, Mutlu O (2011) Chipper: a low-complexity bufferless deflection router. In: 2011 IEEE 17th international symposium on high performance computer architecture (HPCA), pp 144–155. doi:10.1109/HPCA.2011.5749724
41. Gasteier M, Glesner M (1996) Bus-based communication synthesis on system-level. In: 9th international symposium on system synthesis, 1996. Proceedings. IEEE, pp. 65–70
42. Goossens K, Hansson A (2010) The AEthereal network on chip after ten years: goals, evolution, lessons, and future. In: Proceedings of the 47th design automation conference, DAC'10. ACM, New York, pp 306–311. doi:10.1145/1837274.1837353
43. Hansson A, Wiggers M, Moonen A, Goossens K, Bekooij M (2008) Applying dataflow analysis to dimension buffers for guaranteed performance in networks on chip. In: Proceedings of international symposium on networks on chip (NOCS). IEEE Computer Society, Washington, DC, pp 211–212
44. Hesse R, Jerger NE (2015) Improving DVFS in NoCs with coherence prediction. In: 2015 9th IEEE/ACM international symposium on networks on chip (NoCS)
45. Hölzenspies PKF, Hurink JL, Kuper J, Smit GJM (2008) Run-time spatial mapping of streaming applications to a heterogeneous multi-processor system-on-chip (MPSoC). In: Proceedings of the conference on design, automation and test in Europe, DATE'08. ACM, New York, pp 212–217. doi:10.1145/1403375.1403427
46. Hoskote Y, Vangal S, Singh A, Borkar N, Borkar S (2007) A 5-GHz mesh interconnect for a TeraFlops processor. IEEE Micro 27(5):51–61
47. Howard J, Dighe S, Vangal S, Ruhl G, Borkar N, Jain S, Erraguntla V, Konow M, Riepen M, Gries M, Droege G, Lund-Larsen T, Steibl S, Borkar S, De V, Van Der Wijngaart R (2011) A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. IEEE J Solid-State Circuits 46(1):173–183. doi:10.1109/JSSC.2010.2079450
48. Hu Z, Buyuktosunoglu A, Srinivasan V, Zyuban V, Jacobson H, Bose P (2004) Microarchitectural techniques for power gating of execution units. In: Proceedings of the 2004 international symposium on low power electronics and design. ACM, pp 32–37
49. Huang PK, Hashemi M, Ghiasi S (2008) System-level performance estimation for application-specific MPSoC interconnect synthesis. In: Symposium on application specific processors, SASP 2008, pp 95–100. doi:10.1109/SASP.2008.4570792

50. Iordanou C, Soteriou V, Aisopos K (2014) Hermes: architecting a top-performing fault-tolerant routing algorithm for networks-on-chips. In: 2014 IEEE 32nd international conference on computer design (ICCD). IEEE, pp 424–431
51. Jan Y, Jóźwiak L (2012) Communication and memory architecture design of application-specific high-end multiprocessors. VLSI Des 2012:12:12–12:12. doi:10.1155/2012/794753
52. Javaid H, He X, Ignjatovic A, Parameswaran S (2010) Optimal synthesis of latency and throughput constrained pipelined MPSoCs targeting streaming applications. In: 2010 IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis (CODES+ISSS), pp 75–84
53. Jerraya A, Wolf W (2005) Multiprocessor systems-on-chips. Electronics & electrical. Morgan Kaufmann. http://books.google.com.au/books?id=7i9Z69lrYBoC
54. Kermani P, Kleinrock L (1979) Virtual cut-through: a new computer communication switching technique. Comput Netw (1976) 3(4):267–286
55. Kim G, Kim J, Yoo S (2011) Flexibuffer: reducing leakage power in on-chip network routers. In: 2011 48th ACM/EDAC/IEEE design automation conference (DAC), pp 936–941
56. Krishna T, Chen CHO, Kwon WC, Peh LS (2014) Smart: single-cycle multihop traversals over a shared network on chip. IEEE Micro 34(3):43–56
57. Kumar R, Zyuban V, Tullsen DM (2005) Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling. In: 32nd international symposium on computer architecture, ISCA'05. Proceedings. IEEE, pp 408–419
58. Kumar S, Jantsch A, Soininen JP, Forsell M, Millberg M, Oberg J, Tiensyrja K, Hemani A (2002) A network on chip architecture and design methodology. In: IEEE computer society annual symposium on VLSI, 2002. Proceedings. IEEE, pp 105–112
59. Le Beux S, Bois G, Nicolescu G, Bouchebaba Y, Langevin M, Paulin P (2010) Combining mapping and partitioning exploration for NoC-based embedded systems. J Syst Archit 56(7):223–232. doi:10.1016/j.sysarc.2010.03.005
60. Le Beux S, Nicolescu G, Bois G, Bouchebaba Y, Langevin M, Paulin P (2009) Optimizing configuration and application mapping for MPSoC architectures. In: NASA/ESA conference on adaptive hardware and systems, AHS 2009, pp 474–481. doi:10.1109/AHS.2009.35
61. Lee HG, Chang N, Ogras UY, Marculescu R (2007) On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus, and network-on-chip approaches. ACM Trans Des Autom Electron Syst (TODAES) 12(3):23
62. Ly D, Saldana M, Chow P (2009) The challenges of using an embedded MPI for hardware-based processing nodes. In: International conference on field-programmable technology, FPT 2009, pp 120–127. doi:10.1109/FPT.2009.5377688
63. Ma S, Enright Jerger N, Wang Z (2011) DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In: Proceedings of the 38th annual international symposium on computer architecture, ISCA'11. ACM, New York, pp 413–424. doi:10.1145/2000064.2000113
64. Mahadevan S, Angiolini F, Storgaard M, ndahl Olsen RG, SparsøJ (2005) A network traffic generator model for fast network-on-chip simulation. In: Proceedings of design, automation and test in Europe conference and exhibition (DATE). Mahadevan05.pdf
65. Matsutani H, Koibuchi M, Ikebuchi D, Usami K, Nakamura H, Amano H (2011) Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for CMPs. IEEE Trans Comput-Aided Des Integr Circuits Syst 30(4):520–533. doi:10.1109/TCAD.2011.2110470
66. Matsutani H, Koibuchi M, Wang D, Amano H (2008) Adding slow-silent virtual channels for low-power on-chip networks. In: Second ACM/IEEE international symposium on networks-on-chip, NoCS 2008, pp 23–32. doi:10.1109/NOCS.2008.4492722
67. Medardoni S (2009) Driving the network-on-chip revolution to remove the interconnect bottleneck in nanoscale multi-processor systems-on-chip. Ph.D. thesis, Università degli studi di Ferrara
68. Mirzoyan D, Akesson B, Goossens K (2014) Process-variation aware mapping of best-effort and real-time streaming applications to MPSoCs. ACM Trans Embed Comput Syst (TECS) 13(61):61:1–61:24

69. Mishra A, Das R, Eachempati S, Iyer R, Vijaykrishnan N, Das C (2009) A case for dynamic frequency tuning in on-chip networks. In: 42nd annual IEEE/ACM international symposium on microarchitecture, MICRO-42, pp 292–303
70. Mishra AK, Mutlu O, Das CR (2013) A heterogeneous multiple network-on-chip design: an application-aware approach. In: Proceedings of the 50th annual design automation conference, DAC'13. ACM, New York, pp 36:1–36:10. doi:10.1145/2463209.2488779
71. Moscibroda T, Mutlu O (2009) A case for bufferless routing in on-chip networks. SIGARCH Comput Archit News 37(3):196–207. doi:10.1145/1555815.1555781
72. Murali S, Benini L, De Micheli G (2007) An application-specific design methodology for on-chip crossbar generation. IEEE Trans Comput-Aided Des Integr Circuits Syst 26(7):1283–1296. doi:10.1109/TCAD.2006.888284
73. Murali S, Coenen M, Radulescu A, Goossens K, De Micheli G (2006) Mapping and configuration methods for multi-use-case networks on chips. In: Asia and South Pacific conference on design automation, 6pp. doi:10.1109/ASPDAC.2006.1594673
74. Murali S, Coenen M, Rădulescu A, Goossens K, De Micheli G (2006) A methodology for mapping multiple use-cases on to networks on chip. In: Proceedings of design, automation and test in Europe conference and exhibition (DATE). European Design and Automation Association, 3001 Leuven, pp 118–123
75. Nejad AB, Goossens K, Walters J, Kienhuis B (2009) Mapping KPN models of streaming applications on a network-on-chip platform. In: Proceedings of annual workshop on circuits, systems and signal processing (ProRisc)
76. Nesson T, Johnsson SL (1995) ROMM routing on mesh and torus networks. In: Proceedings of the seventh annual ACM symposium on parallel algorithms and architectures. ACM, pp 275–287
77. Nikitin N, Cortadella J (2012) Static task mapping for tiled chip multiprocessors with multiple voltage islands. In: Proceedings of the 25th international conference on architecture of computing systems (ARCS), pp 50–62
78. Ogras UY, Marculescu R, Marculescu D, Jung EG (2009) Design and management of voltage-frequency island partitioned networks-on-chip. IEEE Trans Very Large Scale Integr (VLSI) Syst 17(3):330–341
79. Palesi M, Holsmark R, Kumar S, Catania V (2006) A methodology for design of application specific deadlock-free routing algorithms for NoC systems. In: Proceedings of the 4th international conference on hardware/software codesign and system synthesis. ACM, pp 142–147
80. Parikh R, Das R, Bertacco V (2014) Power-aware NoCs through routing and topology reconfiguration. In: 2014 51st ACM/EDAC/IEEE design automation conference (DAC). IEEE, pp 1–6
81. Park S, Krishna T, Chen CH, Daya B, Chandrakasan A, Peh LS (2012) Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45 nm soi. In: Proceedings of the 49th annual design automation conference, DAC'12. ACM, New York, pp 398–405. doi:10.1145/2228360.2228431
82. Passas G, Katevenis M, Pnevmatikatos D (2012) Crossbar NoCs are scalable beyond 100 nodes. IEEE Trans Comput-Aided Des Integr Circuits Syst 31(4):573–585
83. Peh L, Jerger N (2009) On-chip networks (synthesis lectures on computer architecture). Morgan and Claypool, San Rafael
84. Phillips S (2014) M7: next generation sparc. In: Hot chips: a symposium on high performance chips
85. Rădulescu A, Dielissen J, Goossens K, Rijpkema E, Wielage P (2004) An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. In: Proceedings of design, automation and test in Europe conference and exhibition (DATE), vol 2. IEEE Computer Society, Washington, DC, pp 878–883
86. Salamy H, Ramanujam J (2012) An effective solution to task scheduling and memory partitioning for multiprocessor system-on-chip. IEEE Trans Comput-Aided Des Integr Circuits Syst 31(5):717–725. doi:10.1109/TCAD.2011.2181848

87. Salminen E, Lahtinen V, Kuusilinna K, Hamalainen T (2002) Overview of bus-based system-on-chip interconnections. In: IEEE international symposium on circuits and systems, ISCAS 2002, vol 2. IEEE, pp II–372
88. Samih A, Wang R, Krishna A, Maciocco C, Tai C, Solihin Y (2013) Energy-efficient interconnect via router parking. In: 2013 IEEE 19th international symposium on high performance computer architecture (HPCA2013), pp 508–519. doi:10.1109/HPCA.2013.6522345
89. Seiculescu C, Murali S, Benini L, De Micheli G (2009) NoC topology synthesis for supporting shutdown of voltage islands in SoCs. In: 46th ACM/IEEE design automation conference, DAC'09, pp 822–825
90. Seiculescu C, Murali S, Benini L, De Micheli G (2010) A method to remove deadlocks in networks-on-chips with wormhole flow control. In: Proceedings of the conference on design, automation and test in Europe. European Design and Automation Association, pp 1625–1628
91. Shafik RA, Rosinger P, Al-Hashimi BM (2008) MPEG-based performance comparison between network-on-chip and AMBA MPSoC. In: 11th IEEE workshop on design and diagnostics of electronic circuits and systems (DDECS) 2008, pp 1–6
92. Shang L, Peh LS, Jha N (2003) Dynamic voltage scaling with links for power optimization of interconnection networks. In: The ninth international symposium on high-performance computer architecture, HPCA-9 2003. Proceedings, pp 91–102. doi:10.1109/HPCA.2003.1183527
93. Singh AK, Shafique M, Kumar A, Henkel J (2013) Mapping on multi/many-core systems: survey of current and emerging trends. In: Proceedings of the 50th IEEE/ACM design automation conference (DAC), pp 1:1–1:10. doi:10.1145/2463209.2488734
94. Soteriou V, Peh LS (2007) Exploring the design space of self-regulating power-aware on/off interconnection networks. IEEE Trans Parallel Distrib Syst 18(3):393–408. doi:10.1109/TPDS.2007.43
95. Srinivasan K, Chatha KS (2006) A low complexity heuristic for design of custom network-on-chip architectures. In: Proceedings of the conference on design, automation and test in Europe: Proceedings. European Design and Automation Association, pp 130–135
96. Stergiou S, Angiolini F, Carta S, Raffo L, Bertozzi D, De Micheli G (2005) Xpipes lite: a synthesis oriented design library for networks on chips. In: Design, automation and test in Europe, 2005. Proceedings, vol 2, pp 1188–1193. doi:10.1109/DATE.2005.1
97. Stuijk S (2007) Predictable mapping of streaming applications on multiprocessors. Ph.D. thesis, Eindhoven University of Technology
98. Stuijk S, Basten T, Geilen M, Corporaal H (2007) Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs. In: 44th ACM/IEEE design automation conference, DAC'07, pp 777–782
99. Teich J (2012) Hardware/software codesign: the past, the present, and predicting the future. Proc IEEE 100(Special Centennial Issue), 1411–1430. doi:10.1109/JPROC.2011.2182009
100. Thiele L, Bacivarov I, Haid W, Huang K (2007) Mapping applications to tiled multiprocessor embedded systems. In: International conference on application of concurrency to system design, pp 29–40. doi:10.1109/ACSD.2007.53
101. Towles B, Grossman J, Greskamp B, Shaw DE (2014) Unifying on-chip and inter-node switching within the Anton2 network. In: 2014 ACM/IEEE 41st international symposium on computer architecture (ISCA). IEEE, pp 1–12
102. Weichslgartner A, Gangadharan D, Wildermann S, GlaßM, Teich J (2014) DAARM: design-time application analysis and run-time mapping for predictable execution in many-core systems. In: Proceedings of the international conference on hardware/software codesign and system synthesis (CODES+ISSS), pp 34:1–34:10. doi:10.1145/2656075.2656083
103. Wentzlaff D, Griffin P, Hoffmann H, Bao L, Edwards B, Ramey C, Mattina M, Miao CC, Brown III JF, Agarwal A (2007) On-chip interconnection architecture of the tile processor. IEEE Micro 27(5):15–31. doi:10.1109/MM.2007.89

104. Yoo J, Lee D, Yoo S, Choi K (2007) Communication architecture synthesis of cascaded bus matrix. In: Asia and South Pacific, design automation conference, ASP-DAC'07. IEEE, pp 171–177
105. Yoo S, Rha K, Cho Y, Kung J, Choi K (2002) Performance estimation of multiple-cache IP-based systems: case study of an interdependency problem and application of an extended shared memory model. In: International workshop on hardware/software codesign (CODES)
106. Zhan J, Stoimenov N, Ouyang J, Thiele L, Narayanan V, Xie Y (2013) Designing energy-efficient NoC for real-time embedded systems through slack optimization. In: Proceedings of the 50th annual design automation conference. ACM, p 37
107. Zhang YP, Jeong T, Chen F, Wu H, Nitzsche R, Gao GR (2006) A study of the on-chip interconnection network for the IBM Cyclops64 multi-core architecture. In: IEEE parallel and distributed processing symposium (IPDPS)
108. Zhu J, Sander I, Jantsch A (2010) Constrained global scheduling of streaming applications on MPSoCs. In: 2010 15th Asia and South Pacific design automation conference (ASP-DAC), pp 223–228. doi:10.1109/ASPDAC.2010.5419892
109. Zimmer C, Mueller F (2012) Low contention mapping of real-time tasks onto TilePro64 core processors. In: Proceedings of the 2012 IEEE 18th real time and embedded technology and applications symposium, RTAS'12. IEEE Computer Society, Washington, DC, pp 131–140. doi:10.1109/RTAS.2012.36