Chapter 2
NoC Basics

As the number of IP modules in Systems-on-Chip (SoCs) increases, bus-based interconnection architectures may prevent these systems to meet the performance required by many applications. For systems with intensive parallel communication requirements buses may not provide the required bandwidth, latency, and power consumption. A solution for such a communication bottleneck is the use of an embedded switching network, called Network-on-Chip (NoC), to interconnect the IP modules in SoCs. NoCs design space is considerably larger when compared to a bus-based solution, as different routing and arbitration strategies can be implemented as well as different organizations of the communication infrastructure. In addition, NoCs have an inherent redundancy that helps tolerate faults and deal with communication bottlenecks. This enables the SoC designer to find suitable solutions for different system characteristics and constraints.

This chapter first presents the main concepts involved in the design and use of networks-on-chip such as the basic building blocks, examples of structural and logic organizations of those blocks, performance parameters, and the definition of quality-of-service in the NoC environment. Then, a few examples of NoC implementations are listed, showing how different communication solutions can be defined over the same basic concepts. The reader can find a number of books that explore in detail the design and implementation issues of a NoC. For instance, one can mention Bertozzi et al. (2007), Dally and Towles (2004), De Micheli and Benini (2006), Duato et al. (2003), Flitch and Bertozzi (2010), Jantsch and Tenhunen (2010), and Peh and Jerger (2009). We refer to this material as the prime reference on the concepts resumed in this chapter.

2.1 NoC Structure and Design Space

A network-on-chip is composed of three main building blocks. The first and most important one are the links that physically connect the nodes and actually implement the communication. The second block is the router, which implements the communication protocol (the descentralized logic behind the communication protocol). One can see the

É. Cota et al., *Reliability, Availability and Serviceability of Networks-on-Chip*,
DOI 10.1007/978-1-4614-0791-1_2, © Springer Science+Business Media, LLC 2012

NoC as an evolution of the segmented busses where the router plays the role of a "much smarter buffer" (Bjerregaard and Mahadevan 2006). The router basically receives packets from the shared links and, according to the address informed in each packet, it forwards the packet to the core attached to it or to another shared link. The protocol itself consists of a set of policies defined during the design (and implemented within the router) to handle common situations during the transmission of a packet, such as, having two or more packets arriving at the same time or disputing the same channel, avoiding deadlock and livelock situations, reducing the communication latency, increasing the throughput, etc. The last building block is the network adapter (NA) or network interface (NI). This block makes the logic connection between the IP cores and the network, since each IP may have a distinct interface protocol with respect to the network.

2.1.1 Links

A communication link is composed of a set of wires and connects two routers in the network. Links may consist of one or more logical or physical channels and each channel is composed of a set of wires. In the remaining chapters, unless stated otherwise, the words net, wire, and line mean a single wire interconnecting two entities (routers and/or IP cores). The words channel and link mean a group of wires connecting two entities. Typically, a NoC link has two physical channels making a full-duplex connection between the routers (two unidirection channels in opposite directions). The number of wires per channel is uniform throughout the network and is known as the channel bitwidth.

The implementation of a link includes the definition of the synchronization protocol between source and target nodes. This protocol can be implemented by dedicated wires set during the communication or through other approaches such as FIFOs (Chelcea and Nowick 2001). Asynchronous links are also an interesting option to implement globally asynchronous locally synchronous (GALS) systems where local handshake protocols are assumed (Bjerregaard and Mahadevan 2006). The links ultimately define the raw performance (due to link delays) and power consumption in an NoC and designers are supposed to provide fast, reliable, and low-power interconnects between nodes in the network.

The concept of flits is defined at the link level. Flits (flow control units) are the atomic units that form packets and streams. In most cases, a flit corresponds to a phit (physical unit), which is the minimum amount of data that is transmitted in one link transaction. In this case, the flit width matches the width of the channel. However, when highly serialized links are used, a flit may be composed of several phits.

2.1.2 Routers

The design and implementation of a router requires the definition of a set of policies to deal with packet collision, the routing itself, and so on. A NoC router is composed of a number of input ports (connected to shared NoC channels), a number of

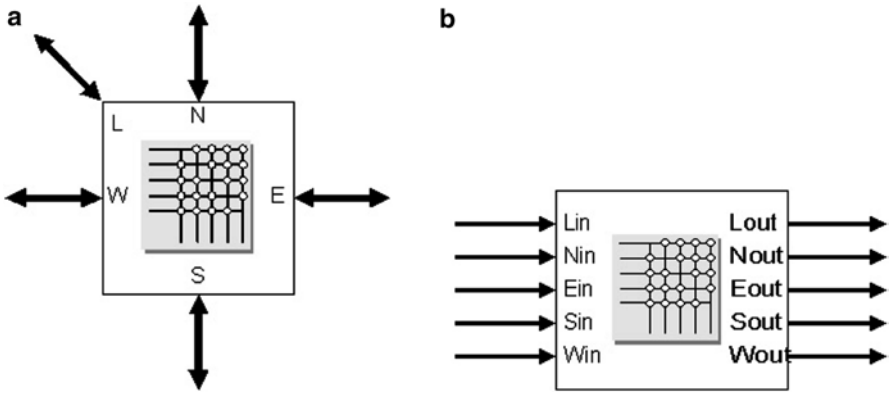

Fig. 2.1 Interface views of a typical router (**a**) functional view, (**b**) architectural view

output ports (connected to possibly other shared channels), a switching matrix connecting the input ports to the output ports, and a local port to access the IP core connected to this router. As an example, the interface of the RaSoC router (Zeferino and Susin 2003) is presented in Fig. 2.1. Herein, we use the terms router and switch as synonymous, but the term switch can also mean the internal switch matrix that actually connects the router inputs to its outputs.

In addition to this physical connection infrastructure, the router also contains a logic block that implements the flow control policies (routing, arbiter, etc.) and defines the overall strategy for moving data though the NoC.

– A **flow control** policy characterizes the packet movement along the NoC and as such it involves both global (NoC-level) and local (router-level) issues. One can ensure a deadlock-free routing, for instance, by taking specific measures in the flow control policy (by avoiding certain paths within the NoC for example). Also, the optimization of the NoC resources usage (channels, bandwidth, etc.) and the guarantees on the communication can be ensured as part of the flow control policy (for instance, by choosing a routing algorithm that minimizes the path or by implementing virtual channels to reduce congestion, etc.). Guarantees on the communication performance and quality are kown as "quality-of-service" and will be detailed later on. Control can be centralized or distributed. In *centralized control*, routing decisions are made globally and applied to all nodes, with an strategy that garantees no traffic contention. This approach avoids the need for and arbitration unit but requires that all nodes share a common sense of time. A possible implementation of this approach is the use of Time Division Multiplexing (TDM) mechanisms where each packet is associated to a time frame (Millberg et al. 2004; Goossens et al. 2005). However, NoCs typically use a *distributed control*, where each router makes decisions locally. *Virtual channels* (VCs) are an important concept related to the flow control. VCs implement the concept of multiplexing a single physical channel over several logically separate channels with individual and independent buffer queues. The main goal of a VC implementation is to improve performance by avoiding deadlocks, optimizing

wire usage and providing some traffic guarantees (Bjerregaard and Mahadevan 2006). *Deadlock* occurs when network resources are fully occupied and waiting for each other to be released to proceed with the communication, that is, when two paths are blocked in a cyclic fashion (Dally and Towles 2004). *Livelock* occurs when the status of the resources keep changing (there is no deadlock) but the communication is not completed.

– The **routing** algorithm is the logic that selects one output port to forward a packet that arrives at the router input. This port is selected according to the routing information available in the packet header. There are several possible routing algorithms that can be used in a NoC, each one leading to different trade-offs between performance and cost. For instance, in a *deterministic* routing a packet always uses the same path between two specific nodes. Common deterministic routing schemes are source routing and XY routing. In source routing, the source core specifies the route to the destination. In XY routing, the packet follows the rows first, then moves along the columns toward the destination or vice versa (Bjerregaard and Mahadevan 2006). In the *adaptive* routing, alternative paths between two nodes may be used if the original path or a local link is congested. This involves a dynamic evaluation of the link load and implies a dynamic load balancing strategy. Negative First (NF) and West First (WF) algorithms proposed by Glass and Ni (1994) are examples of adaptive routing algorithms. In the *static* routing, paths between cores are defined at compilation time (of the application), while in the *dynamic* routing the path is defined at run-time. An *unicast* routing indicates that a packet has a single target whereas in the *multicast* routing a packet can be sent to several nodes in the NoC simultaneously (similar to a bus) or several slaves of a master node. Similarly, a *broadcast* communication targets all nodes whereas a *narrowcast* communication initiated by a master is related to a single slave associated to it. A routing algorithm can also be classified as *minimal* or *non-minimal*. A *minimal* routing guarantees that the shortest path to destination is always chosen. Minimal routing algorithms are those where a bounding box is virtually present and implies that only decreasing distances from source to destination are valid. On the other hand, non minimal routing algorithms allow increasing the distance from source to destination. Routing algorithms can lead to or avoid the occurrence of deadlocks and livelocks. For instance, the *turn model* (Glass and Ni 1994) is a routing algorithm that prohibits certain turns that could lead to a cycle in the network and to a risk of deadlock. The *odd-even turn model* (Chiu 2000) restricts the locations in the network where some types of turns can be taken. Another routing algorithm worth mentioning is the *hot potato* routing. In this algorithm, the packet is immediately forwarded towards the path with the lowest delay (instead, for example, of the shortest or minimal path). This routing scheme is also called *deflective* routing because if a packet cannot be accepted by the target node, it is deflected into the network, to return at a later time. The packet is not stored in a buffer (buffer-less approach) and each packet has a set of preferred outputs that will be used whenever possible in the forwarding operation.
– While the routing algorithm selects an output port for a packet, the **arbitration logic** implemented in the router selects one input port when multiple packets

arrive at the router simultaneously requesting the same output port. Again, one has several options to implement the arbiter: it can be *distributed* (one per port) or *centralized* (one per router), it can be based on *static* (fixed) or *dynamic* (variable) priorities among ports. A centralized arbiter optimizes the use of the router switching matrix, but may lead to higher latency whereas the distributed approach optimizes the latency. Arbitration logic also defines whether the network assumes a *delay* or a *loss* communication model. In the delay model, packets can be delayed, but never dropped. In the *loss* model a packet can be dropped as a solution, for instance, to a congestion situation. In this case, retransmission logic must be implemented as well (Bjerregaard and Mahadevan 2006).

– The **switching** defines how the data is transmitted from the source node to the target one. In the *circuit switching* approach the whole path (including routers and channels) from source to node is previously established (by the header) and reserved for the transmission of the whole packet. The payload is not sent until the whole path has been reserved. This can increase latency, but once the path is defined, this approach can give some guaranteed throughput, for example. In the *packet-based* switching approach on the other hand, all flits of the packet are sent as the header establishes the connection between routers. Still in this model the designer can choose between different buffering and forward strategies that impact the overall NoC traffic (storing the whole packet in each router before establishing the connection to the next router or sending the flits in a pipeline mode for instance). In the *store-and-forward* strategy, the node stores the complete packet before forwarding it to the next node in the path. In this strategy one must ensure that the buffer size at each node is sufficient to store the whole packet or the packet can be stalled. In the *wormhole* strategy, on the other hand, the node makes the routing decision and forwards the packet as soon as the header arrives. The subsequent flits follow the header as they arrive. This reduces the latency within the router, but in case of packet stalling, many links risk to be locked at once. The *virtual-cut-through* mechanism is similar to the wormhole approach but, before forwarding the first data flit to the next node in the path, the node waits for a confirmation that the whole packet can be accepted by the next node. Thus, in case of stalling, no links are affected, only the current node.
– The **Buffering** policy is the strategy used to store information in the router when there is congestion in the network and a packet cannot be forwarded right away. The buffering strategy (number, location and size of the buffers), has an important impact on the network traffic and, therefore, on the NoC performance. In addition, the buffers are responsible for a large portion of the router area. One can have a *single buffer* in the router, shared by all input ports, or one can have one *buffer per port* (input or output). The main advantage of the first approach is the area optimization, but the control can be more complex and additional care must be taken to deal with buffer overflow. In the distributed approach, each input port has its own buffer and the most common implementation is in the form of a FIFO, although other implementations are also possible. Distributed output buffers are also possible, but they tend to be less efficient because several input ports may need to store data in a single structure.

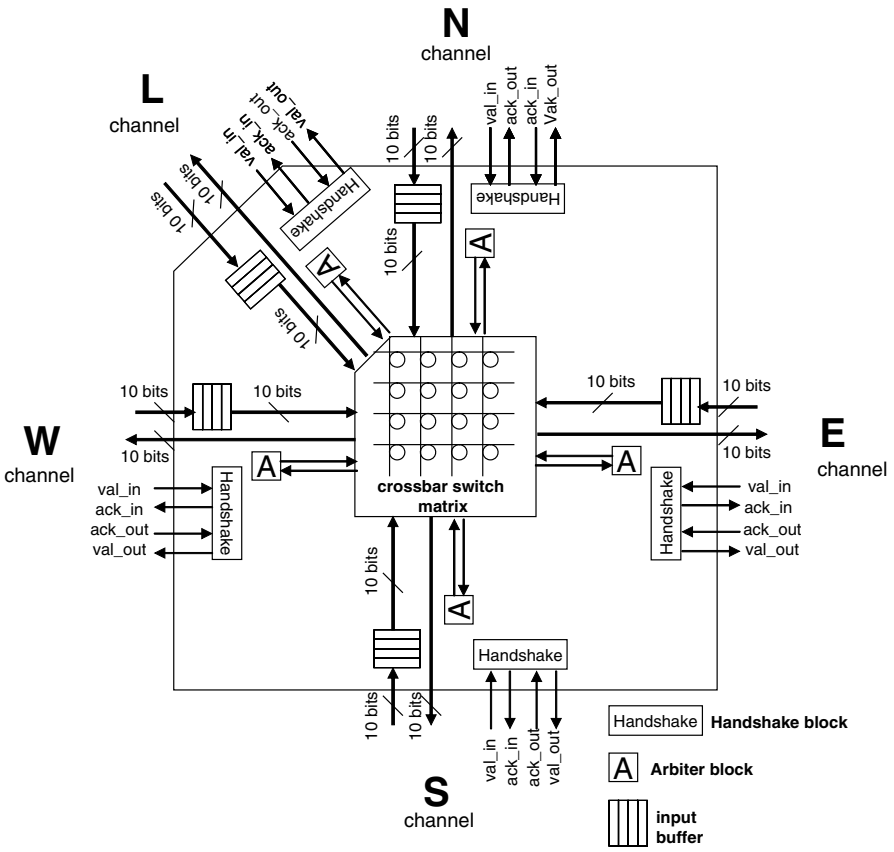

Fig. 2.2 Architecture of a typical router

Figure 2.2 depicts a typical router architecture (used in 2D NoCs) with the above mentioned elements identified. One can observe that the NoC designer has several possible strategies to implement a router (and, therefore, the network communication protocol) leading to a really large design space. This is the main advantage of the NoC approach and explains why this platform has more chances to meet the system communication requirements. Different from a bus structure, one can tail the NoC according to the specific requirements of the application and issues such as guaranteed performance can be naturally implemented as part of the network flow control.

2.1.3 Network Interface

The third NoC building block is the network adapter (NA) or network interface (NI). This block makes the logic connection between the IP cores and the network,

since each IP may have a distinct interface protocol with respect to the network. This block is important because it allows the separation between computation and communication. This allows the reuse of both, core and communication infrastructure independent of each other (Bjerregaard and Mahadevan 2006).

The adapter can be divided into two parts: a front end and a back end. The front end handles the core requests and is ideally unaware of the NoC. This part is usually implemented as a socket – OCP (OCPIP 2011), VCI (VSI Alliance 2011), AXI (ARM 2011), DTL (Philips Semiconductors 2002), etc. The back end part handles the network protocol (assembles and disassembles the packet, reorder buffers, implement synchronization protocols, helps the router in terms of storage, etc.).

2.2 NoC Performance Parameters

The performance of a network-on-chip can be evaluated by three parameters: bandwidth, throughput, and latency.

The **bandwidth** refers to the maximum rate of data propagation once a message is in the network. The unit of measure for bandwidth is bit per second (bps) and it usually considers the whole packet, including the bits of the header, payload and tail.

Throughput is defined by Duato et al. (2003) as the maximum traffic accepted by the network, that is, the maximum amount of information delivered per time unit. The throughput measure is messages per second or messages per clock cycle. One can have a normalized throughput (independently from the size of the messages and of the network) by dividing it by the size of the messages and by the size of the network. As a result, the unit of the normalized throughput is bits per node per clock cycle (or per second).

Latency is the time elapsed between the beginning of the transmission of a message (or packet) and its complete reception at the target node. Latency is measured in time units and mostly used as comparison basis among different design choices. In this case, latency can also be expressed in terms of simulator clock cycles. Normally, the latency of a single packet is not meaningful (Duato et al. 2003) and one uses the average latency to evaluate the network performance. On the other hand, when some messages present a much higher latency than the average, this may be important. Therefore the standard deviation may be an interesting measure as well.

2.3 NoC Topologies

A NoC can be characterized by the structure of the routers connections. This structure or organization is called **topology** and is represented by a graph G(N,C) where N is the set of routers and C is the set of communication channels. The routers can be connected in direct or indirect topologies.

In the *direct topologies*, each router is associated to a processor and this pair can be seen as a single element in the system (so-called a node in the network). In this topology, each node is directly connected to a fixed number of neighbor nodes and

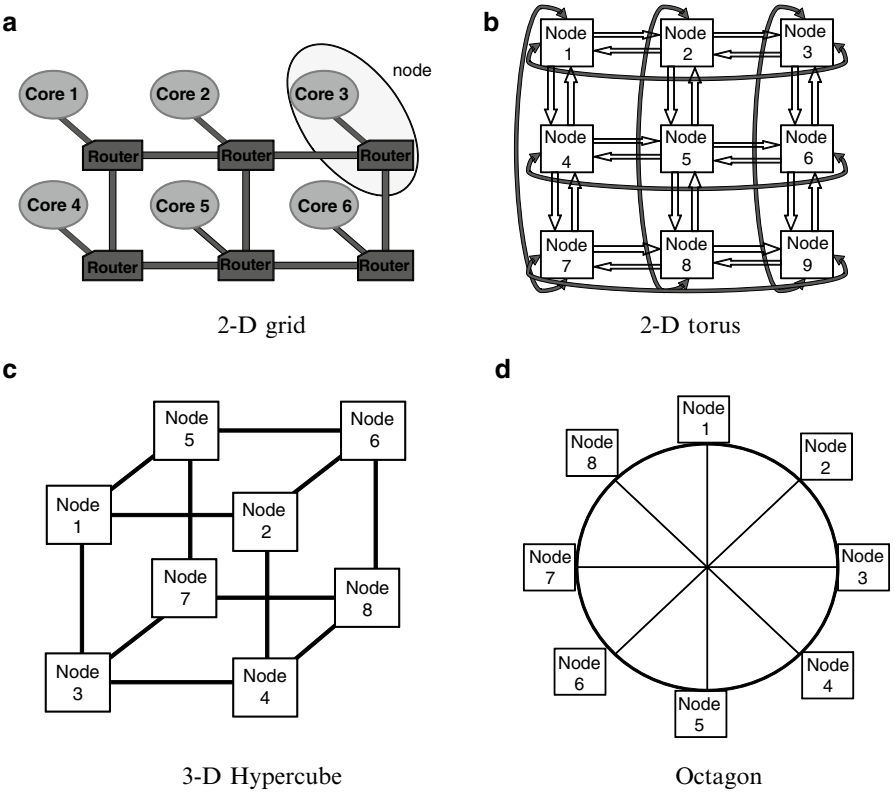

Fig. 2.3 NoCs with direct regular topologies (**a**) 2-D grid, (**b**) 2-D torus, (**c**) 3-D hypercube, (**d**) octagon

a message between two nodes goes through one or more intermediate nodes. Only the routers are involved in the communication in a direct topology and the communication is based on the routing algorithm implemented by the routers. Most NoC implementations are based on orthogonal arrangements of the routers in a direct topology. In this arrangement, nodes are distributed in a n-dimensional space and the packet moves in one dimension at a time. These arrangements are the ones that present the best tradeoff between cost and performance, and also present good scalability. The most common direct topologies are the n-dimensional grid or mesh, torus (or k-ary n-cube) and the hypercube, as shown in Fig. 2.3.

In an *indirect topology* not all routers are connected to processing units as in the direct model. Instead, some routers are used only to propagate the messages through the network, while other routers are connected to the logic and only those can be source and/or target of a message. Some topologies of indirect networks stand out: the crossbar, and the multi-stage. The multi-stage topology is a regular NoC, where routers are identical and organized in stages. Input and output stages are connected to the functional units in one side and to the internal nodes in another side. For instance, Fig. 2.4 shows two examples of indirect networks.

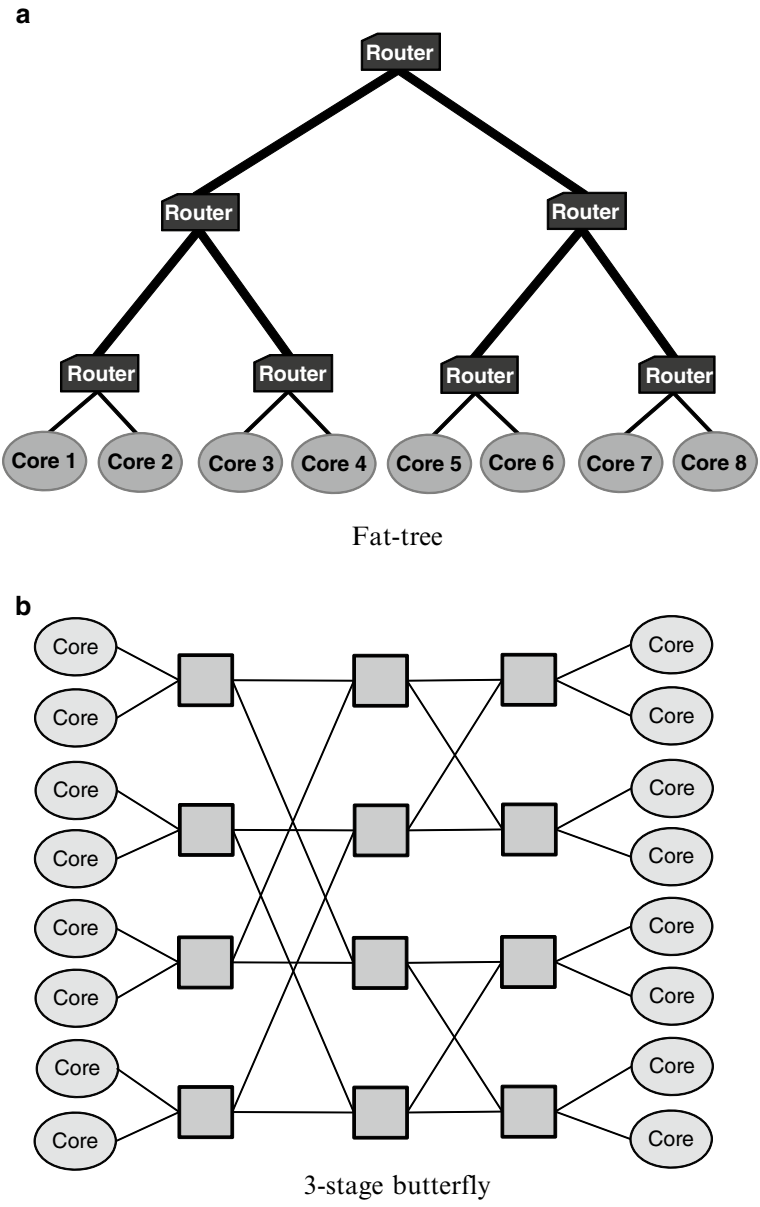

Fig. 2.4 NoCs with indirect topologies (**a**) fat-tree, (**b**) three-stage butterfly

Another possible classification for network topologies is related to the regularity of the connections between routers. In *regular networks*, all routers are identical in terms of number of ports connecting to other routers or elements in the network. For instance, in a regular grid topology presented in Fig. 2.3a all routers have five ports,

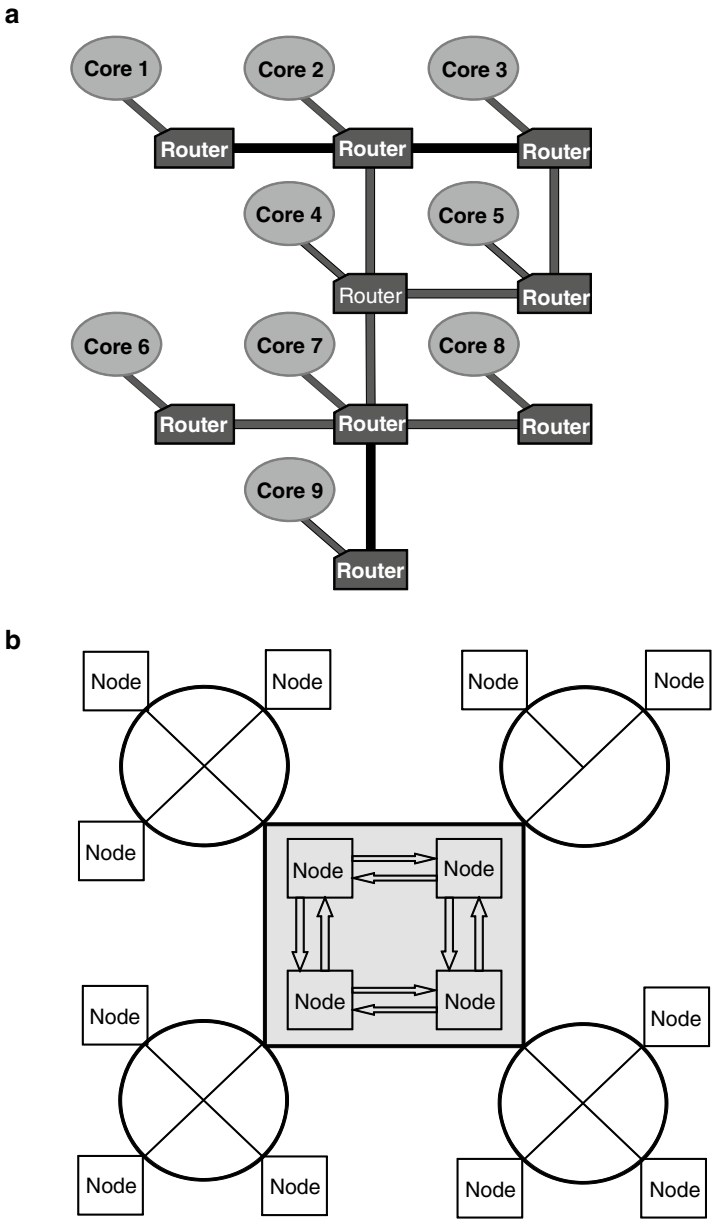

Fig. 2.5 NoCs with irregular topologies (**a**) reduced mesh, (**b**) cluster-based hybrid topology

one local port connecting to the functional unit and another four ports connecting to neighbor routers. In *irregular topologies* the routers may present different connection patterns, usually defined according to the application (Pasricha and Dutt 2008), as depicted in Fig. 2.5.

2.4 Quality of Service

According to Bjerregaard and Mahadevan (2006), quality of service (QoS) is defined "as service quantification that is provided by the network to the demanding core". QoS is also about predictability of the communication behavior (Goossens et al. 2003). Following from the previous definition, one must first define the required services and a quantification measure. Typically, services are defined based on the performance metrics of the network (low latency, high throughput, low power, etc.). Then, one must define implementation mechanisms to meet the core's demands using the network services.

Goossens et al. (2003) define two types of NoCs with respect to QoS:

- *Best-effort* (BE) NoCs – These NoCs offer no commitment. For most NoCs this means that only completion of the communication is ensured.
- *Guaranteed Services* (GS) NoCs – Those NoCs ensure that some services requirements can be accomplished. Commitment can be defined in several levels, such as correctness of the result, completion of the transaction, bounds on performance, and so on.

Due to the characteristics of the on-chip networks and the systems based on those structures, service guarantees and predictability are usually given through hardware implementations, as opposed to statistical guarantees of macro-networks. For instance, in real-time or other critical applications, one really needs guarantees more than predictability. In order to give hard guarantees, one must use specific routing schemes (virtual channels, adaptive routing, fault-tolerant router and link structures, etc.) to ensure the required levels of performance and/or reliability. Naturally, for GS NoCs, implementation costs and complexity grows with the complexity of the system and the QoS requirements. BE NoCs, on the other hand, tends to present a better utilization of the available resources.

2.5 Industrial and Academic NoCs

In order to demonstrate how different communication solutions can be more naturally implemented using a NoC, we mention a few industrial and academic NoC implementations with their basic characteristics. We will not detail these NoCs, we will only mention the most important characteristics to show how even a small difference in the design can create a different communication structure that fits better one application. The variety of NoCs available shows the flexibility of this infrastructure and explains why it is being pointed out as a good solution for future complex chips.

– **ÆTHEREAL** – This NoC was proposed by Philips (Goossens et al. 2005) and it is both a BE and a GT (guaranteed throughput) NoC implemented in a synchronous indirect topology with wormhole switching, and contention-free source routing algorithm based on TDM. It implements a number of connection types

including narrowcast, multicast, and simple connection and the network adapter can be synthesized for four standard socket interfaces (master or slave, OCP, DTL, or AXI based) (Dielissen et al. 2003).

- **STNoC** – Proposed by ST Microelectronics (Karim et al. 2002), this GS NoC presents a spidergon/ring topology with minimal path, 32-bit links, and input buffering. The main difference in this NoC is the topology, which is quite efficient from the performance point of view.
- **Nostrum** – In this guaranteed bandwidth NoC proposed by KTH (Kumar et al. 2002), the protocol includes the data encoding to reduce power consumption. It is implemented in a 2D mesh topology with hot potato routing. Links are composed of 128 bits of data plus 10 bits of control. Virtual channels and a TDM mechanism are used to ensure bandwidth.
- **SPIN** – One of the first proposed NoC architectures, SPIN network was developed at LIP6 (Guerrier and Greiner 2000). It presents a fat-tree topology with wormhole switching, deterministic and adaptive (deflective) routing, input buffering and two shared output buffering. It uses 36-bit links (32 data bits plus 4 control bits). It also implements the VCI socket in the network interface.
- **XPIPES** – This NoC presents an arbitrary topology, tailored to the application to improve performance and reduce costs (Osso et al. 2003). It requires a better support from CAD tools to be synthesized (Jalabert et al. 2004; Murali and De Micheli 2004) and was proposed in a cooperation between University of Bologna and Stanford University. XPipes consists of soft macros of switches and links that can be instantiated during synthesis. The standard OCP protocol is used in the network interface.
- **SoCin** – SoCin NoC has been proposed by Universidade Federal do Rio Grande do Sul (UFRGS) (Zeferino and Susin 2003). SoCin is a simple and parameterizable BE NoC that can be implemented in 2D mesh or torus topologies, with narrowcasting routing, input buffering, and parameterizable channel width. Due to its simplicity and availability, it has been used as case study for the first papers on NoC-based testing and other test strategies that are discussed later in this book.
- **QNoC** – Developed at Technion in Israel (Bolotin et al. 2004), this direct NoC is implemented in an irregular mesh topology with wormhole switching and XY minimal routing scheme. Four different classes of traffic are defined to improve QoS, although hard guarantees are not given.
- **HERMES** – This NoC was proposed by Pontifícia Universidade Católica do Rio Grande do Sul (Moraes et al. 2004) and implements a direct 2-D mesh topology with wormhole switching and minimal XY routing. Hermes is a best-effort NoC with parameterizable input queuing and presents a whole set of support development tools.
- **MANGO** – Developed at Technical University of Denmark this NoC implements a message-passing asynchronous (clockless) protocol with GS services over OCP interfaces. Mango also provides BE services using credit-based and source routing (Bjerregaard et al. 2005). BE connections are source routed and virtual channels are used in GS communications.

The list above is by no means exhaustive and several other implementations can be listed: CHAIN (Bainbridge and Furber 2002), Proteo (Siguenza-Tortosa et al. 2004), SoCBus(Sathe et al. 2003), ANoC (Beigne et al. 2005), etc. Each NoC has a distinct aspect that leads to a distinct tradeoff among performance, QoS, cost, fault tolerance, etc. This only shows that indeed the design space for this communication infrastructure is quite larger when compared to a bus-based solution.

Despite the variability in the design decisions, one can map some common design trends among available NoCs: most implementations use packet switching for communications for its efficiency; most NoCs use 2D mesh topologies because of the good tradeoff between cost and performance; XY routing is very common, for mesh topologies, although not standard, due to its property of being deadlock-free; most NoCs use input buffering only, again because of the tradeoff between cost and performance gain.

References

VSI Alliance (2011) Virtual component interface standard version 2. VSI alliance. www.vsi.org. Accessed 26 May 2011

ARM (2011) AMBA advanced extensible interface (AXI) protocol specification, version 2.0. http://www.arm.com. Accessed 26 May 2011

Bainbridge J, Furber S (2002) CHAIN: a delay-insensitive chip area interconnect. IEEE Micro 22(5):16–23

Beigne E, Clermidy F, Vivet P, Clouard A, Renaudin M (2005) An asynchronous NOC architecture providing low latency service and its multi-level design framework. In: Proceedings of the 11th international symposium on asynchronous circuits and systems (ASYNC), Pasadena, pp 54–63

Bertozzi D, Kumar S, Palesi M (2007) Networks-on-chip. Hindawi Publishing Corporation, New York, NY

Bjerregaard T, Mahadevan S (2006) A survey of research and practices of network-on-chip. ACM Comput Surv 38:1–51

Bjerregaard T, Mahadevan S, Olsen RG, Sparsø J (2005) An OCP compliant network adapter for gals-based soc design using the MANGO network-on-chip. In: Proceedings of international symposium on system-on-chip (ISSoC), Tampere, Finland, pp 171–174

Bolotin E, Cidon I, Ginosar R, Kolodny A (2004) QNoC: QoS architecture and design process for network on chip. Elsevier J Syst Architect EUROMICRO J 50:2–3, 105–128

Chelcea T, Nowick SM (2001) Robust interfaces for mixed-timing systems with application to latency-insensitive protocols. In: Proceedings of the 38th design automation conference (DAC), Las Vegas, pp 21–26

Chiu G-M (2000) The odd-even turn model for adaptive routing. IEEE Trans Parallel Distrib Syst 11(7):729–738

Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. The Morgan Kaufmann series in computer architecture and design. Morgan Kaufmann, Burlington

De Micheli G, Benini L (2006) Networks on chips: technology and tools (systems on silicon). Morgan Kaufmann, Burlington

Dielissen J, Radulescu A, Goossens K, Rijpkema E (2003). Concepts and implementation of the Phillips network-on-chip. In: Proceedings of the IP based SOC (IPSOC), Grenoble, France

Duato J, Yalamanchili S, Ni LM (2003) Interconnection networks: an engineering approach. Morgan Kaufmann, Burlington

Flich J, Bertozzi D (2010) Designing network on-chip architectures in the nanoscale era. Chapman & Hall/CRC Computational Science, Boca Raton

Glass C, Ni L (1994) The turn model for adaptive routing. J Assoc Comput Mach 41(5):874–902

Goossens K, Dielissen J, van Meerbergen J, Poplavko P, Radulescu A, Rijpkema E, Waterlander E, Wielage P (2003) Guaranteeing the quality of services in networks on chip. In: Jantsch A, Tenhunen H (eds) Networks-on-chip. Kluwer, Boston

Goossens K, Dielissen J, Radulescu A (2005) Æthereal network on chip: concepts, architectures and implementations. IEEE Design Test Comput 22(5):414–421

Guerrier P, Greiner A (2000) A generic architecture for on-chip packet-switched interconnections. In: Proceedings of the design automation and test in Europe conference (DATE), Paris, pp 250–256

Jalabert A, Murali S, Benini L, De Micheli G (2004). XpipesCompiler: a tool for instantiating application specific networks-on-chip. In: Proceedings of design, automation and testing in Europe conference (DATE), Dresden, pp 884–889

Jantsch A, Tenhunen H (2010) Networks on chip. Kluwer, Boston

Karim F, Nguyen A, Dey S (2002) An interconnect architecture for networking systems on chips. IEEE Micro 22(5):36–45

Kumar S, Jantsch A, Soininen J-P, Forsell M, Millberg M, Oberg J, Tiensyrj A K, Hemani A (2002). A network-on-chip architecture and design methodology. In: Proceedings of the computer society annual symposium on VLSI (ISVLSI), Pittsburgh, pp 117–124

Millberg M Nilsson E, Thid R, Jantsch A (2004). Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network-on-chip. In: Proceedings of design, automation and testing in Europe conference (DATE), Paris, pp 890–895

Moraes F, Calazans N, Mello A, Möller L, Ost L (2004) HERMES: an infrastructure for low area overhead packet-switching networks on chip. VLSI Integr 38:69–93

Murali S and De Micheli G (2004) SUNMAP: A tool for automatic topology selection and generation for NoCs. In: Proceedings of the 41st design automation conference (DAC), San Diego, pp 914–919

OCPIP (2011) http://www.ocpip.org/. Accessed 25 May 2011

Osso M D, Biccari G, Giovannini L, Bertozzi D, Benini L (2003) Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In: Proceedings of 21st international conference on computer design (ICCD), San Jose, pp 536–539

Pasricha S, Dutt N (2008) On-chip communication architectures: system on chip interconnect (systems on silicon). Morgan Kaufmann, Burlington

Peh L-S, Jerger NE (2009) On-chip networks (synthesis lectures on computer architecture). Morgan and Claypool, San Rafael

Philips Semiconductors (2002) Device Transaction Level (DTL) protocol specification, version 2.2

Sathe S, Wiklund D, Liu D (2003). Design of a switching node (router) for on-chip networks. In: Proceedings of the 5th International Conference on ASIC, Beijing, pp 75–78

Siguenza-Tortosa D, Ahonen T, Nurmi J (2004) Issues in the development of a practical NoC: the Proteo concept. Integr VLSI J 38(1):95–105

Zeferino CA, Susin AA (2003) SoCIN: a parametric and scalable network-on-chip. In: Proceedings of the 16th symposium on integrated circuits and systems design (SBCCI), Sao Paulo, pp 169–174

http://www.springer.com/978-1-4614-0790-4

Reliability, Availability and Serviceability of Networks-on-Chip

Cota, É.; Morais Amory, A. de; Soares Lubaszewski, M.

2012, XIII, 209 p. 103 illus., Hardcover
ISBN: 978-1-4614-0790-4