APPENDIX

Networks-on-Chip

Sheng Ma
State Key Laboratory of High Performance Computing
P. R. China

F.1 Introduction

Network on chip or network on a chip (NoC or NOC) is a communication subsystem on an integrated circuit (commonly called a "chip"), typically between IP cores in a system on a chip (SoC). NoCs can span synchronous and asynchronous clock domains or use unclocked asynchronous logic. NoC technology applies networking theory and methods to on-chip communication and brings notable improvements over conventional bus and crossbar interconnections. NoC improves the scalability of SoCs, and the power efficiency of complex SoCs compared to other designs.

F.2 Communication Centric Design

The steadily shrinking of semiconductor device size driven by Moore's law provides computer architects a rapid increase transistor budget to design and discover innovative techniques to scale processor performance. Until the beginning of this century, the traditional methodology for performance scaling is to design complex single-core processors, including leveraging deep pipelines to improve the frequency, utilizing sophisticated superscalar techniques to squeezing the instruction level parallelism. Yet, since the switching power is proportional to the frequency and the sophisticated superscalar techniques involve complex combinational logic and large memory arrays, the traditional performance scaling methodology induces sharp increase in power consumption. Also, it only gains diminishing performance returns as the already exploited instruction parallelism nearly reaches the intrinsic limitation of sequential programs.

To efficiently reap the benefits of billions of transistors, computer architects are actively pursuing multicore designs to keep sustained performance growth. Instead of increasing the clock frequency or proposing sophisticated superscalar techniques, multicore designs increase computational throughput by packing multiple cores on the same chip. These multiple cores can be assigned with the multiple threads of parallel applications to exploit the thread level and task level parallelism. This can achieve significantly higher performance than using a single core. Also, multicore processors require a more modest engineering effort, since different amounts of the same core can be stamped down to form a family of processors. Since Intel's cancellation of the two single-core processors Tejas and Jayhawk in 2004 and switching to developing dual-core processors, the community has been rapidly embracing the multicore design methodology. The current processors have already integrated tens or hundreds of cores; the computer architecture enters into the many-core era.

In contrast to challenges of single-core processor designs appearing in the last century, many-core processors face several new challenges, ranging from the high-level parallel application programming to low-level logic implementations. Central to many of these challenges is the ability to communicate on-chip in a low latency, high throughput, low-power and robust fashion. In fact, semiconductor technology scaling trends suggest that communication, not computation, will dominate the correctness and performance, as well as delay, area and power budgets. Communication is also critical to support the distributed design style where many heterogeneous or homogeneous cores or IP blocks are merged into a single die to create a complete system.

In many-core platforms, the traditional bus communication paradigm faces several challenges, including poor scalability, low bandwidth, large latency, high power consumption. Bus limits the amount of connected nodes. Bus is efficient to provide shared communication for less than ten nodes. If more nodes are needed to be connected, bus has serious competition conflicts, which dramatically increases the latency and reduce the throughput. The global long bus across the chip has large capacitance and consumes large amounts of energy. Also, the large coupling capacitance between bus wires induces great interference on signal transmission, reducing the communication reliability.

To address the limitations of the traditional bus, Network-on-Chip (NoC) is proposed as a simple, efficient and scalable communication paradigm for many-core platforms. NoCs separate the computation from the communication. It is easy to overcome the bus' limitations. NoCs eliminate the centric control unit to reduce competition conflicts, which improves the scalability, as well as the latency and throughput. Compared with the global bus wires across the whole chip, it is more convenient to manage the resistance and capacitance of the distributed wire segments between routers; this can achieve high power efficiency, reliable synchronization and robust communication.

Due to the many desirable properties, NoC has quickly crystallized into a significant research domain of the computer architecture community. The architectural exploration of NoCs follows the lines of everything else in computer architecture: it involves a trade-off between performance and overhead. More particularly, the performance includes the latency, throughput and flexibility, and the overhead mainly consists of the area, power consumption and programmability. The next section gives an overall

description of the NoCs design space exploration.

F.3 The Design Space Exploration of NoCs

Figure F.1 illustrates a many-core platform with NoC connection. The 16 processing cores are connected by a 4x4 mesh network. Each network node includes a processing node and a network router. The processing node maybe replaced by other hardware units, such as a special accelerator or a memory controller. The router is composed of the buffer, crossbar, allocator and routing unit. The NoC design exploration mainly consists of the network topology, the routing algorithm, the flow control mechanism, and the router micro-architecture.

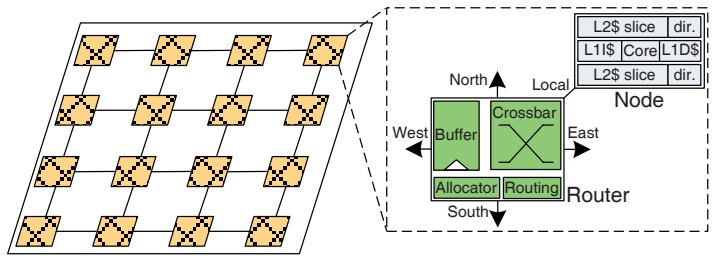

FIGURE F.1 A many-core platform with Noc communication paradigm. The 2D Mesh is one of the most common NoC topology. We will see other typical topologies in Figure F.2. A network node usually need a processing unit for efficient routing computation and a router for packet and flow control information transmission.

Topology

The NoC topology has profound effect on the overall network cost and performance. It determines the physical layout and connections between nodes and channels in the network. The message traverse hops as well as each hop's channel length depend on the topology. Thus, the topology influences the latency and power consumption significantly. Furthermore, the topology determines the amount of alternate paths between nodes, which affects the network traffic distribution and hence the network bandwidth.

Any NoC topology must match well onto the two-dimensional surface of an integrated circuit. This requirement excludes several excellent topologies proposed for high performance parallel computers; an example of such is the hyper cube. Therefore, NoCs generally apply regular and simple topologies. Figure F.2 shows the structure of three typical NoC topologies, a ring, a 2D mesh and a 2D torus. The ring topology is widely used with small amount of network nodes, including the 6-node Ivy Bridge and the 12-node Cell. The ring network has a very simple structure, and it is easy to implement the routing and

flow control, or even apply centralized control mechanism. Yet, due to the high average hop count, the scalability of a ring network is limited; the 2D mesh or torus networks are usually used with tens of hundreds of nodes.

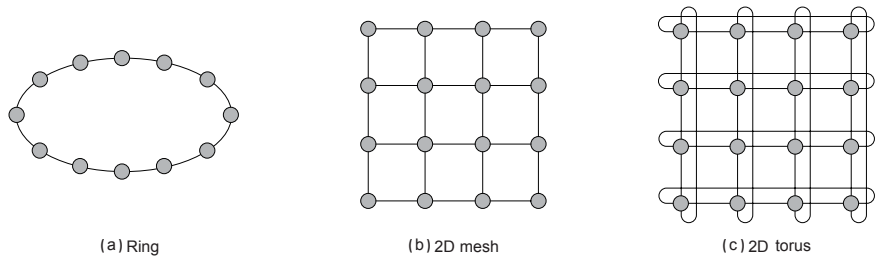

FIGURE F.2 Three typical NoC topologies. All these three topologies are very simple and are widely used in current NoC design. However, they are suitable for different situations because of their own characteristics.

The 2D mesh is quite suitable for the wire routing in a multi-metal layers CMOS technology, and it is also very easy to achieve deadlock freedom. Furthermore, its scalability is better than the ring. These factors make the 2D mesh topology widely used in several industrial products, including the TRIPS, Teraflops, and Tile64. A limitation of the 2D mesh topology is that its center portion is easily congested and becomes a bottleneck. The node-symmetry of a 2D torus helps to balance the network utilization. The torus' wraparound links convert plentiful on-chip wires into bandwidth, and reduce the hop count and latency. Yet, the torus needs additional effort to avoid deadlock. See S. Ma, Z.Wang, Z. Liu, N. Enright Jerger (2013), for more detail.

Routing Algorithm

Once the NoC topology is determined, the routing algorithm calculates out the traverse paths for packets from the source nodes to the destination nodes. This section leverages the 2D mesh as the platform to introduce routing algorithms.

There are several classification criteria for routing algorithms. As shown in Figure F.3, according to the traverse path length, the routing algorithms can be classified as minimal routing and non-minimal routing. The routing paths of non-minimal routing maybe outside the minimal quadrant defined by the source node and the destination node. The Valiant routing algorithm is a typical non-minimal routing. It firstly routes the packet to a random intermediate destination ('inter. dest' shown in Figure F.3a), then routes the packet from this intermediate destination to the final destination.

In contrast, the packet traverse path of minimal routing is always inside the minimal quadrant defined by the source node and the destination node. Figure F.3b shows two minimal routing paths for one source and destination pair. Since minimal routing's packet traverse hops are less than non-minimal routing, its power consumption is generally lower. Yet, non-minimal routing is more appropriate to achieve global load balance and fault tolerant.

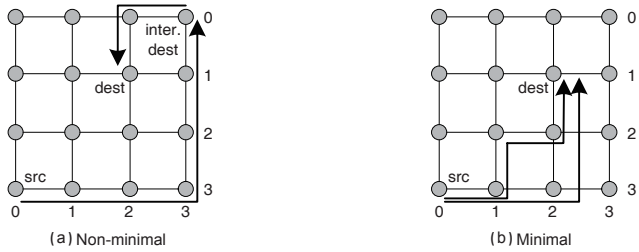

FIGURE F.3 Non-minimal routing and minimal routing. Routing algorithms can be classified as minimal routing and non-minimal routing according to the traverse path length, the routing algorithms. There are some other routing algorithm classification strategies.

The routing algorithms can be divided into deterministic routing and non-deterministic routing based on the provided path count. The deterministic routing offers only one path for each source and destination pair, while the non-deterministic routing may offer more than one path. When calculating the routing paths, some non-deterministic routing algorithms consider the network status. These routing algorithms are named as adaptive routing. Other non-deterministic routing algorithms do not consider the network congestion, and they are classified as oblivious routing.

One critical issue of the routing algorithm design is to achieve deadlock freedom. Deadlock freedom requires that there is no cyclic dependency among the network resources. Some proposals eliminate the cyclic dependency by forbidding certain turns inside the network. These routing algorithms are generally classified as partially adaptive routing since they do not allow the packets to utilize any path between the source and the destination. Other proposals, named fully adaptive routing, allow the packet to utilize all minimal paths between the source and the destination. They achieve deadlock freedom by utilizing the virtual channels to form acyclic dependency graphs.

Flow Control

Flow control allocates the network resources, such as channel bandwidth, buffer capacity, and control state, to traversing packets. The store-and-forward, virtual cut-through, and wormhole are three main flow control types. Only after receiving the entire packet, the store-and-forward applies the buffers and channels of next hop. This mechanism induces the serialization latency at each packet forwarding hop.

To reduce the serialization latency, the virtual cut-through applies the buffers and channels of next hop once receiving the header flit of a packet. The virtual cut-through allocates the buffers and channels at the packet granularity; it needs the downstream router to have enough buffers for the entire packet before forwarding the header flit. In virtual cut-through flow control, the router must at least have enough buffers for one entire packet.

In contrast, the minimal buffer requirement of wormhole flow control is one buffer slot. As the name implies, this technique sends the packet like a worm crawling forward. It can send out the head flit if there is at least one buffer slot in the

downstream. However, when the congestion occurs, one packet may keep in several routers, which exacerbates the network congestion, this problem is named as head of line (HOL) blocking.

To reduce the effect of HoL blocking, several virtual channels can be configured for one physical channel. Multiple virtual channels share the bandwidth of one physical channel. They allow one packet to bypass the current blocked packet; this alleviates the HoL blocking. In addition to the wormhole flow control, the virtual channel can be leveraged with both the store-and-forward and the virtual cut-through.

F.4 Router Micro-architecture

Figure F.4 illustrates the structure of a canonical virtual channel NoC router, which is composed of the input units, routing computation logic, VC allocator, switch allocator, crossbar and output units. The input unit consists of the input buffer and the link control logic. If the incoming flit is a head flit, the routing computation logic calculates out the output port for this packet. Some routing algorithms may provide the output VCs as well.

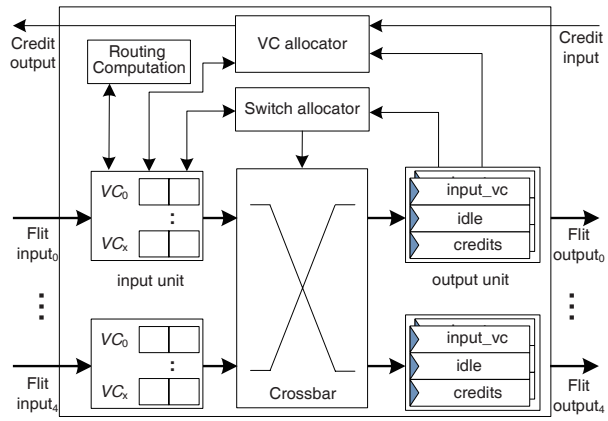

FIGURE F.4 The structure of a typical virtual channel NoC router. The canonical NoC virtual channel router was first described by Peh and Dally. The router is input-queued and has five ports, of which four are network ports and one is an injection port. Key architectural elements of the router include the virtual channel FIFOs, routing computation unit, VC allocation logic, crossbar allocation logic and the crossbar itself.

After finishing the routing computation, the head flit requests the output VCs. The VC allocator handles the requests from multiple head flits. It guarantees that one output VC is at most allocated to one input VC, and each input VC is at most granted by one output VC. The VC allocation is performed at packet granularity. The switch allocator

handles the switch traversal requests from multiple flits. It allocates each output physical channel to at most one flit. Both the VC allocator and the switch allocator consist of several arbiters.

The output unit tracks the status of downstream VCs. The 'input vc' register records the input VC that a downstream VC is allocated to. The one-bit 'idle' register indicates whether the downstream VC receives the tail flit of last allocated packet. The 'Credits' register records the credit amount.

F.5 Performance Metric

Capacity and latency are two common performance metrics used for NoC evaluation. Capacity is defined as the injection bandwidth offered to each of the N terminal nodes for uniform random traffic. It can be calculated from the bisection bandwidth (B_B) or the bisection channel count (B_C) and the channel bandwidth (b) as:

$$\text{Capacity} = 2B_B/N = 2bB_C/N \tag{F.1}$$

Since global wiring requirements depends on the channel bisection, equation (F.1) expresses a tradeoff between the NoC performance and the implementation complexity.

The average contention-free latency (T_0) from source s to destination d depends on several parameters, including: the average hop count (H) from s to d and router traversal latency (t_r), which determine the routing latency; the channel traversal latency (T_c); and the packet length (L) and channel bandwidth (b), which determine the serialization latency (T_s). T0 is calculated as:

$$T_0(s, d) = H(s, d)t_r + T_c(s, d) + L/b \tag{F.2}$$

As the load on the network increases, greater resource contention contributes to longer router traversal latencies.

F.6 Concluding Remarks

Although NoCs can borrow concepts and techniques from the well-established domain of computer networking, it is impractical to blindly reuse features of "classical" computer networks and symmetric multiprocessors. In particular, NoC switches should be small, energy-efficient, and fast. Neglecting these aspects along with proper, quantitative comparison was typical for early NoC research but nowadays they are considered in more detail. The routing algorithms should be implemented by simple logic, and the number of data buffers should be minimal. Network topology and properties may be application-specific.

Further Reading

G.M. Chiu. [2000]. *The Odd-Even Turn Model for Adaptive Routing*. Parallel and Distributed Systems, IEEE Transactions on, 11(7):729 –738.

W. Dally and C. Seitz. [1987]. *Deadlock-Free Message Routing in Multiprocessor Interconnection Networks*. IEEE Transactions on Computer.

W. Dally and B. Towles.[2003]. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc.

W. J. Dally and C. L. Seitz.[1986]. *The Torus Routing Chip*. Distributed Computing, 1:187–196.

S. Damaraju et al. [2012]. *A 22nm IA Multi-CPU and GPU System-on-Chip*. In ISSCC 2012.

J. Duato. [1993]. *A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks*. Parallel and Distributed Systems, IEEE Transactions on, 4(12):1320 –1331.

J. Duato. [1995]. *A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks*. Parallel and Distributed Systems, IEEE Transactions on, 6(10):1055 –1067.

B. Fu et al. [2011]. *An Abacus Turn Model for Time/Space-Efficient Reconfigurable Routing*. In ISCA 2011.

C. Glass and L. Ni. [1992]. *The Turn Model for Adaptive Routing*. In ISCA 1992.

P. Gratz et al. [2007]. *On-Chip Interconnection Networks of the TRIPS Chip*. Micro, IEEE, 27(5):41 –50.

Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. [2007]. *A 5-ghz Mesh Interconnect for a Teraflops Processor*. Micro, IEEE, 27(5):51–61.

J. A. Kahle et al. [2005]. *Introduction to the Cell Multiprocessor*. IBM J. Res. Dev., 49(4.5):589 –604.

P. Kermani and L. Kleinrock. [1979]. *Virtual Cut-Through: A New Computer Communication Switching Technique*. Computer Networks, 3:267–286.

S. Ma, N. Enright Jerger, and Z. Wang. [2012]. *Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-Chip*. In HPCA, 2012.

S. Ma, N. Enright Jerger, Z.Wang, L. Shen, and N. Xiao. [2013]. *Novel Flow Control for Fully Adaptive Routing in Cache-Coherent NoCs*. IEEE Transactions on Parallel and Distributed Systems, 99(Preprint).

S. Ma, Z.Wang, Z. Liu, N. Enright Jerger. [2013]. *Leaving One Slot Empty: Flit Bubble Flow Control for Torus Cache-coherent NoCs*. IEEE Transactions on Computers, 99(Preprint).

L. Valiant and G. Brebner. [1981]. *Universal Schemes for Parallel Communication*. In STOC 1981.

D. Wentzlaff et al. [2007]. *On-Chip Interconnection Architecture of the Tile Processor*. Micro, IEEE, 27(5):15 –31.