Chapter 2
A Baseline NoC Architecture

A generic NoC implementation consists of a number of Processing Elements (PE) arranged in a mesh-like grid, as shown in Fig. 2.1 The PEs may be of the same type, e.g., CPU, or of different type, e.g., audio cores, video cores, wireless transceivers, memory banks, etc. Each PE is connected to a local router through a Network Interface Controller (NIC); each router is, in turn, connected to adjacent routers forming a packet-based on-chip network. The NIC module packetizes/de-packetizes the data into/from the underlying interconnection network. The PE together with its NIC form a network node. Nodes communicate with each other by injecting data packets into the network. The packets traverse the network toward their destination, based on various routing algorithms and control flow mechanisms.

The heart of an on-chip network is the router, which undertakes the crucial task of steering and coordinating the data flow. The architecture employed by conventional NoC routers [48] is illustrated in Fig. 2.2 The router operation revolves around two fundamental regimes: (a) the datapath, and (b) the associated control logic.

The datapath consists of a number of input and output channels to facilitate packet switching and traversal. In general, the router has P input and P output channels (or ports). In most implementations, P = 5; four inputs from the four cardinal directions (North, East, South and West) and one from the local Processing Element (PE), which is attached to the NoC router. To minimize router complexity and traffic congestion, NoC routers are usually assumed to connect to a single PE. The input/output channels may consist of unidirectional links (as shown in Fig. 2.2), bidirectional, or even serial links.

Buffering within a network router is necessary due to congestion, output link contention, and intra-router processing delays (e.g., routing computation), which impede data flow. In the case of virtual channel-based NoC routers, each input port consists of a number of FIFO buffers, with each FIFO corresponding to a virtual channel (see Fig. 2.2). Hence, each input port has v virtual channels, each of which has a dedicated k-flit FIFO buffer (a flit is the smallest unit of flow control; one network packet is composed of a number of flits). Given the very limited buffer space in resource-constrained on-chip networks, routers tend to employ wormhole flow control, which relaxes the constraints on buffer size as compared to store-and-forward and virtual cut-through.

C. Nicopoulos et al., *Network-on-Chip Architectures:*
A Holistic Design Exploration, Lecture Notes in Electrical Engineering 45,
DOI 10.1007/978-90-481-3031-3_2, © Springer Science+Business Media B.V. 2009

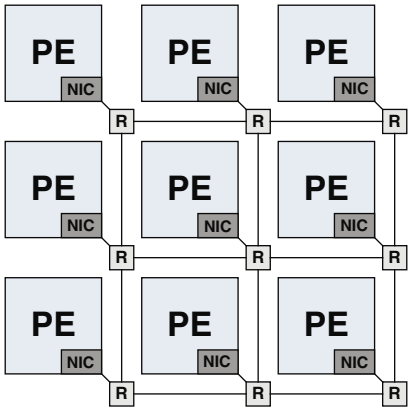

Fig. 2.1 A generic NoC architecture (3×3 mesh)

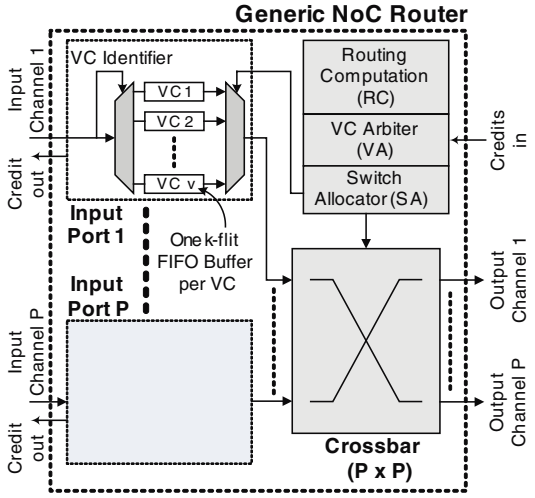

Fig. 2.2 A conventional NoC router

The router control logic forms the heart of the NoC router, and comprises four components: (a) the Routing Computation (RC) unit, (b) the Virtual Channel Arbitration (VA) logic, (c) the Switch Allocation (SA) logic, and (d) the Crossbar (XBAR).

The RC unit is responsible for directing the header flit of an incoming packet to the appropriate output Physical Channel (PC) and/or dictating valid output Virtual Channels (VC) within the selected PC. Output VCs are, essentially, the input VCs of the adjacent routers. The routing is done based on the packet's destination address, which is present in the header flit, and can be deterministic (e.g., dimension-order routing) or adaptive (e.g., load-balancing). RC is a "per-packet" operation: it is performed once for each packet within a router (i.e., only on the header flit of each packet).

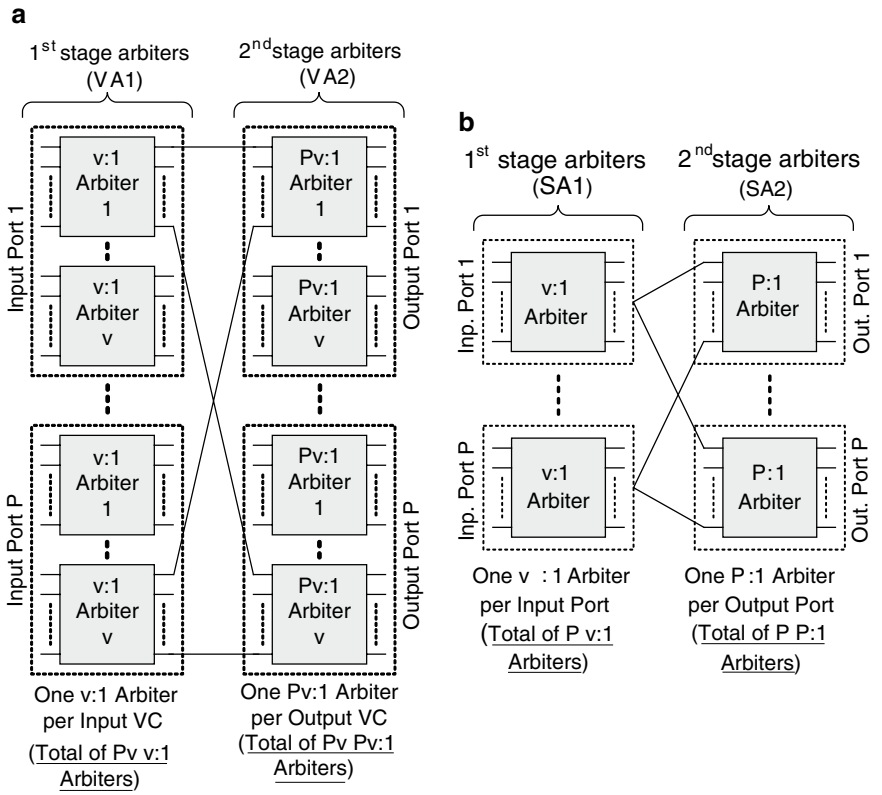

Fig. 2.3 NoC router arbitration/allocation logic. (**a**) VC arbitration (VA). (**b**) Switch allocation (SA)

The Virtual Channel Arbitration (VA) module arbitrates amongst all packets requesting access to the same VCs and decides on winners. Since the routing function may not specify a particular output VC in the requested output physical port [49], two arbitration stages are generally required, as shown in Fig. 2.3a. The first stage (VA1) reduces the number of requests from each input VC to one; this ensures the request of a single VC at a particular output port by each input VC. A total of Pv v:1 arbiters are, therefore, required in the first arbitration stage (see Fig. 2.3a): one arbiter for each input VC. Subsequently, the winning request from each input VC proceeds to the second arbitration stage (VA2). A total of Pv Pv:1 arbiters are required in this stage: one arbiter for each output VC (each arbiter is of Pv:1 size to accommodate the worst-case scenario in which all input VCs request the same output VC). Just like RC, VA is also a "per-packet" operation; it is only performed on header flits.

The SA unit arbitrates amongst all VCs requesting access to the crossbar and grants permission to the winning flits. Switch allocation is also performed in two stages. The first – local – stage (SA1) accounts for the sharing of a single port by a number of VCs; all VCs within the same input port compete against each other locally for

access to the output physical channels. The second – global – stage (SA2) arbitrates between the winning requests from each input port for each output port. The SA2 stage sets the crossbar control signals accordingly. The SA unit is very similar in structure to the VA unit. It consists of logically identical arbiters arranged in cascaded fashion. The difference lies only in the size and number of arbiters used: P v:1 arbiters are required for SA1 (i.e., one arbiter per input port), and P P:1 arbiters are required for SA2 (i.e., one arbiter per output port), as shown in Fig. 2.3b. As opposed to the VA's "per-packet" operation, SA is a "per-flit" operation, i.e., it is performed on all flits traversing the router, not just header flits. The SA winners are then able to traverse the crossbar and are placed on the respective output links.

Simple router implementations require a clock cycle for each component within the router, as illustrated at the top of Fig. 2.4. Lower-latency router architectures parallelize the VA and SA using speculative allocation, which predicts the winner of the VA stage and performs SA based on that [49] (3-stage Router in Fig. 2.4). Further, look-ahead routing can also be employed to perform routing of node $i + 1$ at node i. These two modifications have led to two-stage and even single-stage [48] routers, which parallelize the various stages of operation, as shown in the lower half of Fig. 2.4.

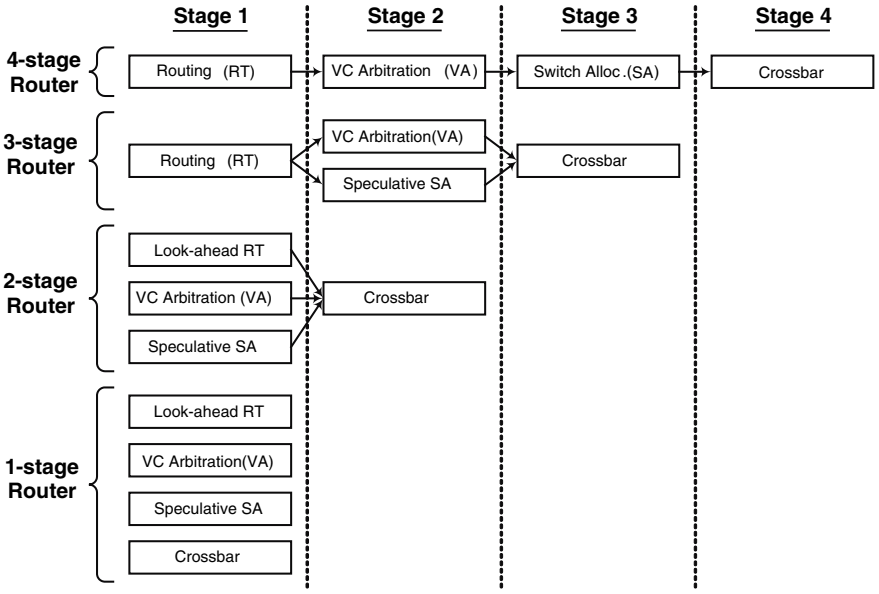

Fig. 2.4 Various NoC router pipeline implementations