9

Efficient and Low-Power NoC Router Architecture

Gul N. Khan

CONTENTS

TNF_09_K338522_C009_docbook_indd.indd 211 27-06-2018 21:56:43

9.1 Introduction

Network-on-Chip (NoC) architecture provides a communications infrastructure for the cores of a multi-core System-on-Chip (SoC). The NoC resources are connected to the SoC cores enabling them to communicate among each other concurrently by sending messages asynchronously. NoC systems improve the scalability and power efficiency of complex SoCs as compared to other conventional communication systems. Figure 9.1a illustrates an SoC including some IP cores which are connected through a 4×5 mesh NoC architecture. The NoC includes a network of switches (routers) which are interconnected by data links. Figure 9.1b illustrates a typical NoC router that consists of some input- and output-ports, an arbiter and a crossbar switch [1]. The input- and output-ports can be simple data buses that connect a router to its channels, but at least one of them should include a circuit to perform buffering and traversal of incoming flits. In all the designs presented in this paper, the input-ports only utilize buffering organization, and the output ports are simple data buses. A most viable communication

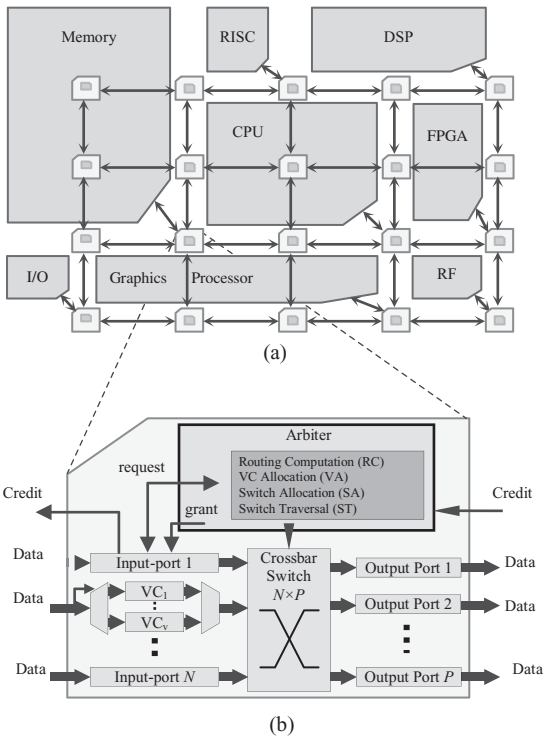

FIGURE 9.1
(a) SoC with 4×5 Mesh NoC architecture (b) NoC router architecture with N inputs, P outputs and v VCs.

mechanism employed in NoCs is packet-based wormhole routing [2]. The messages in wormhole routing are organized as multiple packets where each packet consists of some flits. A flit is a basic unit of data that is generally transferred at clock rate. The first flit of a packet is called the header flit and holds the route information of its associated packet. The remaining flits are called body flits except the last flit, that is called the tail flit. The body and tail flits contain data and can contain two pieces of information: tail state and VC identification.

When the header flit of a packet passes through a route consisting of routers, the route path is reserved for that packet. The route path remains reserved until all the packet flits pass through it. Such a kind of data flow does not always provide optimum performance. In fact, messages from other sources that share their routes or part of their routes with the reserved route must wait until the route becomes free. These waiting conditions continue even when there is no communication through the reserved route. This means that the data flow is not always populated. In other words, the reserved route is sometimes idle and blocks other packets from passing through it. This kind of data flow incurs higher latency and lower utilization of shared NoC resources. One way of alleviating this problem and improving NoC throughput is to utilize Virtual Channel (VC).

In VC organization, the flits of one packet can interleave with the flits of another packet over a physical channel by using a rotating flit-by-flit arbitration. The routing path of each flit can be guaranteed because the flits belonging to a packet are attached with an identical VC identification (*VC-ID*) tag at each router. Then these flits become differentiable at downstream routers. The *VC-ID* tags exist in all the flits of a packet, and these tags are identical when the flits enter a router. Figure 9.2a illustrates the conventional microarchitecture of a VC decoder that is a simple de-multiplexer. In the figure, the *VC-ID* is connected to the selection port of the multiplexer and causes the incoming flit to go to the associated VC buffer. In fact, the flits of a packet are always stored in the same VC buffer.

9.1.1 Static and Dynamic VC Organization

One of the typical forms of buffering is static, i.e. the numbers of VCs and their buffers stay constant during communication (Figure 9.2a). Various studies have shown that for a large number of static VCs, communication load is difficult to balance across them [2]. Some VCs remain idle while the others are overloaded. Therefore, it is better to allocate more buffer storage to busy VCs and less to idle VCs. Moreover, static VC buffers are the expensive components of NoC routers and they become more expensive for larger flit size or when the VC buffer depth becomes larger. The two above drawbacks of static VCs necessitate an adaptive VC organization to achieve VC flow control with maximum buffer utilization. A well-known adaptive VC organization is called Dynamically Allocation Multi Queue (DAMQ).

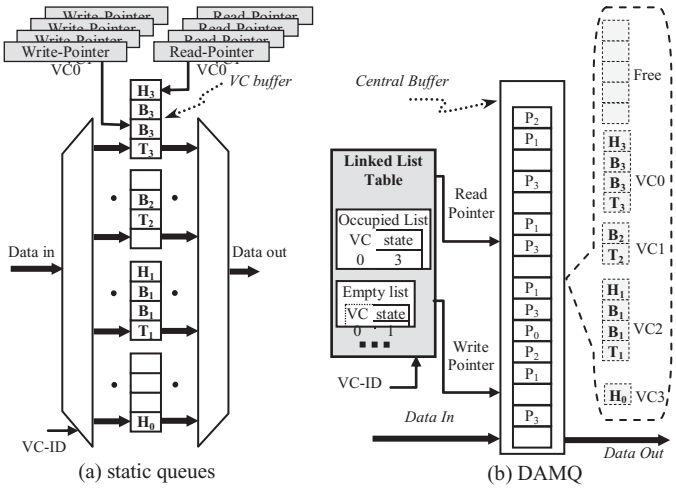

FIGURE 9.2
Input-ports with static and dynamic queues.

Most of the DAMQ organizations are table based [3, 4]. In table-based organization, a central buffer includes multiple VC queues, and a table is utilized in order to keep the flits of each VC in a First Come, First Served (FCFS) order as illustrated in Figure 9.2b. In fact, the table keeps the address of incoming flits in a FCFS order. The dynamic VCs of DAMQ buffers improves the port buffer utilization through sharing its buffer slots among all the VCs of the port and allocating more buffer slots to the active VCs. Large VC buffer depth will keep more flits of a packet and leads to a free route of the packet in wormhole NoC communication. A larger number of free routes reduces contention and, eventually, improves overall NoC performance. Despite the performance merits of DAMQ organizations, they have a number of limitations, listed below:

- The first problem is the complex hardware due to the linked list and dynamic queue management [5, 6].
- Setup limitation is a problem with some DAMQ mechanisms. For example, limitations in the minimum buffer space of each VC, number of VCs and number of flits per packet are observed for some DAMQ schemes.
- Head of Line (HoL) blocking is a problem in the communication of some DAMQ schemes. Assuming that a VC (queue) can receive more than one packet, if the header packet faces a blockage, the other packets in the VC must wait until the blockage is removed.
- There are interventions among the VCs of a DAMQ-based input-port that can lead to higher traffic congestion as compared to static VCs [7].
- Each flit arrival/departure has a large delay due to the complex design of DAMQ-based VCs.

9.1.2 Timing Problem of Table-Based Adaptive Virtual Channels

The flit arrival/departure of a dynamic input-port can be easily compared with the static VC input-ports. Figure 9.2 shows the architectures of these input-ports. The control logic of the static input-port is simpler, where each VC can be configured by using a parallel FIFO buffer, as shown in Figure 9.2a [7]. The number of VCs is equal to the number of FIFOs, as each FIFO represents a VC. The *read-pointer* and the *write-pointer* point to the location of a FIFO, where a flit is read or written, respectively. A pointer works like a simple counter, which is incremented circularly and continuously for each read and write operation. The flit arrival/departure is also simpler in a static input-port. If arbitration takes one step, the arrival/departure of flits in a squeezed pipelined scheme consumes two clock edges, as illustrated in Figure 9.3a. At the entrance of an input-port, the arriving flit is decoded according to its *VC-ID* (VC identification) and by means of the first de-multiplexer. Then it waits to be latched in the FIFO buffer (VC) before the first clock edge. At the first clock edge, the flit is stored in the VC where a request corresponding to that flit is simultaneously issued to the arbiter. At the second clock edge, the arbiter allocates the address for the crossbar switch (output) and ID for the downstream router VC, then issues a *grant* signal. The *grant* signal causes the flit to travel out of the router. For proper operation of the decoder at the entrance of the input-port, the *VC-ID* should be issued before latching the flit in the buffer. Assuming that the flit and its *VC-ID* are transferred at the same clock transition, each flit arrival/departure requires a two-clock event delay in the static router. We have assumed that the FIFOs are dual-port, where the arrival of one flit can coincide with the departure of another flit.

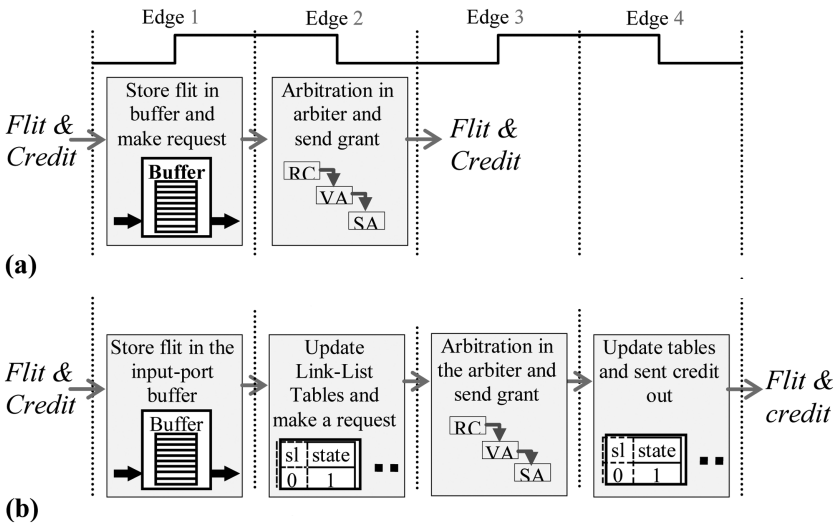

FIGURE 9.3
Static vs. conventional dynamic router pipelines.

In the case of dynamic input-ports, the VC buffers are allocated dynamically, resulting in complex control logic. A linked-list-based DAMQ has been employed as a conventional DAMQ in many research projects [4, 7–9]. Using this mechanism, a central buffer (queue) maintains multiple VCs, and the data flow is directed by Linked-List tables as illustrated in Figure 9.2b [4]. The *read-pointer* and *write-pointer* are updated based on the contents of the Linked-List tables. In a squeezed pipeline design, if arbitration takes one step, the arrival/departure of flits will take four clock edges, as illustrated in Figure 9.3b. A header flit arrives at the input-port and waits to be latched in a VC (buffer). Then the flit is latched into the input-port buffer at the first clock edge. On the second clock edge, the linked-list tables are updated according to the *VC-ID*, which leads to a request signal being issued to the arbiter. The arbiter assigns a proper address for the crossbar switch (output) and ID for the VC at the third clock edge and then issues a *grant* signal. The *grant* signal causes the flit to exit the router, and the Linked-List tables are updated at the fourth clock edge. In a linked-list DAMQ-based VC organization, the *read-* and *write-pointers* cannot be updated at read or write events; instead, they are updated one clock event after the read and write (i.e. after updating the tables). However, in the static queue mechanism, the read and write pointers can be incremented at the read or write events. This causes the pipeline stages in traditional DAMQ (or table-based) input-ports to take two more clock events than those of static input-ports. The same communication characteristics are expected for other table-based DAMQ input-ports such as ViChaR [3]. The VC organization approach presented in this chapter does not employ tables, and its flit arrival/departure delay is equal to that of static VCs but with all the advantages of dynamic VCs.

9.1.3 Data Flow Arbitration

After buffering a flit in a VC of the input-port, the VC issues a request to the arbiter for accessing the shared resources. The arbiter can perform arbitration and allocation through four pipelined stages, as illustrated in Figure 9.4. Each flit of a packet proceeds through Routing Computation (RC), Virtual channel Allocation (VA), Switch Allocation (SA), and Switch Traversal (ST) stages [1]. The RC and VA stages process the header flit (once per packet). The body and tail flits of a packet pass through RC and VA without any processing, whereas SA and ST stages operate on each flit of the packet. For wormhole routing, the destination information of a packet is embedded in the header flit that passes through all the pipeline stages. In our design, the first three stages, i.e. RC, VA and SA, proceed in parallel in one clock cycle, as illustrated in Figure 9.4. However, in past designs, each stage used to take one or more clock cycles. The ST is the last stage and it can be overlapped with the first stage of the following flit [10]. Among the four arbiter stages, VA and SA stage modules perform arbitration.

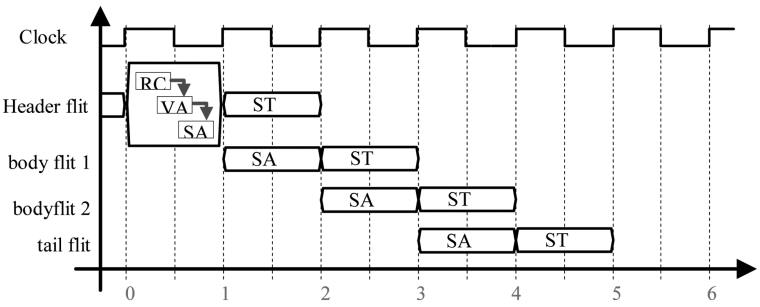

FIGURE 9.4
Pipeline stages of a VC-based arbiter for a four-flit packet that takes five cycles to be arbitrated with no communication stall.

9.1.4 NoC Switch Allocator

The SA module can be designed as a unit to maximize matching among the requesters and the resources. However, such designs are difficult to be parallelized or pipelined, and too slow for the applications where latency is critical [10]. Latency-sensitive applications typically employ fast and fair matching designs such as separable SA modules. A separable SA (switch allocator) is significantly less complex due to separate requesters for some groups. Moreover, the critical path delay of a separable SA can be improved by choosing a fair and fast arbitration among each group. The separable SA modules are usually utilized and investigated as part of NoC research. In a separable SA, two sets of arbiters can perform arbitration: one for the inputs and the other for outputs, as illustrated in Figure 9.5. In an input-first separable SA, arbitration is first performed to select one request at each input-port. The outputs of these *input-arbiters* become the inputs to a set of output arbiters to select a single request for each output-port [10]. In order to ensure fairness and to perform arbitration in a single iteration, a Round Robin (RR) scheme can be utilized, as shown in Figure 9.5. An input-first separable SA micro-architecture employed in our NoC router design is shown in Figure 9.6. The *input-arbiter*s perform arbitration among the VCs of each input-port of the router where the decoder modules generate the requested outputs of the winner VCs of input-ports. Each bit of the decoder output corresponds to an output-port, where an active bit of the decoder output shows the requested output by the winner VC of the relevant input. The second set of arbiters, *output-arbiter*s, perform arbitration among the winner input-ports for the output-ports.

NoC structures are viewed as a suitable solution to meet the wiring challenges of multi-core and many-core systems. NoC designs that consume minimal power and IC area along with higher performance are required for SoC design, especially for low-power applications. Current NoC system and router designs are not optimal. The main problem with the current NoC

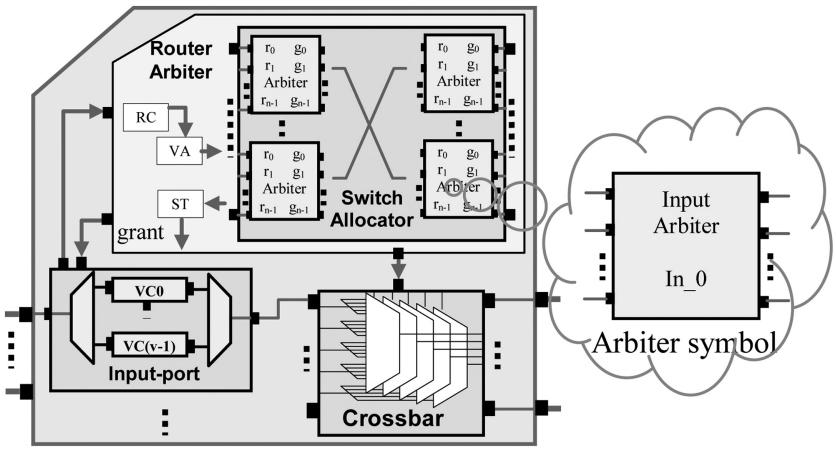

FIGURE 9.5
Wormhole *v*-VC router with Switch Allocator.

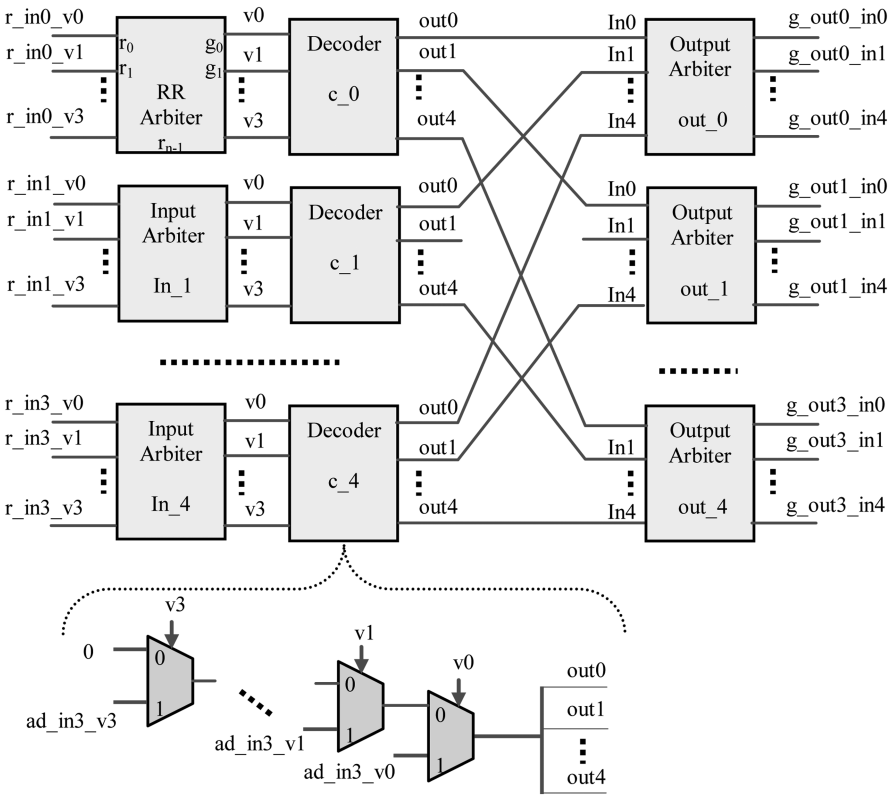

FIGURE 9.6
Micro-architecture of a 5×5 separable SA.

design is related to lower performance during high contention and traffic congestion. The router is the key component of an NoC, and its efficient design and implementation will improve the overall NoC performance. NoC performance and energy budget depends heavily on the usage of buffer resources by the routers. One must utilize the buffer intelligently for NoCs by employing adaptive or dynamic VCs as they have maximum buffer utilization. Current DAMQ buffer design suffers a number of problems, such as complexity, lower buffer utilization, setup limitation, and HoL blocking. Arbitration is another important activity in NoC routers. It also has a few problems, such as complexity, lower speed, weak fairness, traffic starvation, and pipelining difficulty. These arbitration drawbacks and points related to current NoCs have motivated us to investigate the high-performance components of NoC as well as routers including input-port organization and arbitration. The remainder of this chapter is organized as follows. The past research and NoC router design-related techniques are described in Section 9.2. The router and NoC architecture is described in Section 9.3, whereas the router and its various mechanisms are evaluated by experimental results in Section 9.4. The conclusions are drawn at the end of the chapter.

9.2 Overview and Past Works

DAMQ is a unified and dynamically allocated buffer structure that was originally presented by Frazier and Tamir [11]. It is a single storage array that maintains multiple FIFO queues. Nicopoulos *et al.* introduced a centralized shared buffer-based Virtual Channel Regulator (ViChaR) to implement a DAMQ mechanism [3]. In ViChaR, the information of incoming buffer is saved in a table and two trackers. The VC control table module holds the VC slot IDs of all the current flits and it can grow very large for small flit-size big packets. In other words, the ViChaR approach suffers from setup limitations and complexity. Xu *et al.* have presented an application that employs ViChaR architecture [6] where VCs are assigned based on the designated output port of a packet in an effort to reduce the HoL blocking. Unlike ViChaR, this design uses a small number of VCs. Their buffer design is similar to ViChaR except that each VC can store multiple packets and fixed number of VCs employed. Their buffer design uses a smaller arbiter and hybrid (in between static and dynamic) VC allocation scheme.

A 'self-compacting buffers' approach is presented by Park *et al*. to implement DAMQ switching elements [12]. There is no reserved space dedicated for any VC, and a VC can receive any number of flits to occupy the whole channel buffer. Data in the self-compacting buffer is stored in a FIFO manner within the region for each VC. When a flit is to be inserted in the middle of the buffer, the required space is created by moving down all the flits residing

below the insertion address. Similarly, when a read operation is conducted from the top of a region, the data removed from the buffer may result in empty holes in the middle and the data below the read address is shifted up to fill the hole. This approach has some drawbacks. Firstly, when the buffer capacity is increased, its fall-through time also increases, leading to higher latency [13]. Due to which, the latency of the self-compacting buffer depends on its depth rather than the number of stored items. Another drawback is its higher dynamic power consumption due to data shifts in the buffer. Frias and Diaz have proposed an interesting buffer architecture to alleviate the above drawbacks of the self-compacting buffer [14]. They proposed a new cell that has the capability of performing all the required data moves. The novelty of their approach is at the transistor level rather than the gate level.

Evripidou *et al.* have presented a new version of the linked-list mechanism, which mimics the DAMQ organization presented by Frazier and Tamir. The linked-list buffer is expensive in terms of hardware but it leads to higher performance [11]. Our implementation of the Linked-List-based DAMQ (LLD) mechanism is presented here to illustrate the complexity and cost of the LLD mechanism as compared to the proposed VC architectures presented in this chapter. Five lookup tables are used to implement the LLD input-port organization as shown in Figure 9.7. The *VC-State* and *Slot-State* tables keep a Boolean value for each VC and slot (empty/occupied). The *Header-List* table keeps the addresses of slots that contain the header flits of VCs. The *Tail-List* table keeps the addresses of slots that point to the tail flits of VCs. The *Linked-List* table keeps the addresses of the next slot of each slot, or it links

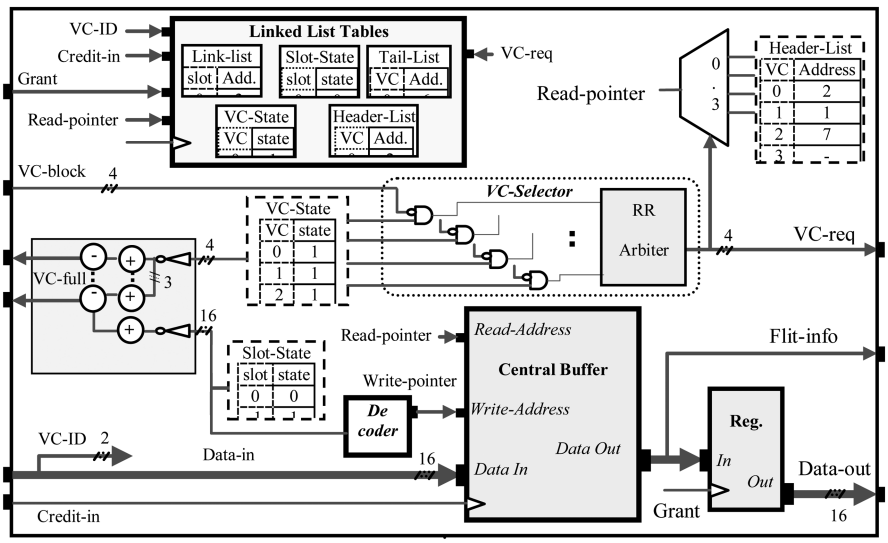

FIGURE 9.7
16-bit 16-slot 4-VC LLD input-port micro-architecture.

the addresses of slots that are associated with each VC in a FIFO manner. The *Slot-State* table has a record of the occupied slots in the central buffer. The pipelined communication illustrated in Figure 9.3b shows the case of LLD routers; flit-data is stored for two clock events and transferred at two clock events. The tables are updated at the negative clock edge, and the signals are detected and issued at the positive clock edge.

The flit arrival and departure is detected by *credit-in* and *grant* signals, respectively. Assume that a VC is empty and ready to accept data. Upon the arrival of a flit for the VC, three things happen simultaneously. First of all, the corresponding bit of the VC is set in the *VC-State* table indicating an occupied VC. Then the content of the *write-pointer* is stored in the *Tail-List* and *Header-List* tables. Finally, the corresponding bit is set in the *Slot-State* table. The *write-pointer* is then updated and it will point to the next free slot to accommodate the incoming flit. When another incoming flit tries to reside in the VC, three things occur at the negative clock edge. The content of the *write-pointer* is stored in a location of the LLD table where the *Tail-List* table points to it and the *write-pointer* is stored in the *Tail-List* table. Finally, the *Slot-State* table is updated, which leads to the updating of the *write-pointer.* When a flit exits from the VC, three types of event take place. If the header and tail addresses are the same (the last flit), the corresponding bit is reset in the *VC-State* indicating the empty VC. Otherwise, the location of the LLD table is identified by the *Header-List* table, and it is stored as the new header address of the VC in the *Header-List* table. Finally, the corresponding bit of the *Slot-State* table is reset, which causes the *write-pointer* to be updated.

The *VC-Selector* module in the router input-port selects a VC for arbitration. It issues the request signal, *VC-req*, and is used to generate the *read-pointer* address as illustrated in Figure 9.7. The *VC-Selector* module has a logic circuit that operates on the contents of *VC-State* table and creates the VC availability signal (*VC-ava*). The *VC-block* signal is reversed and ANDed with its corresponding bit in the *VC-State* table as illustrated in Figure 9.7. The Round Robin (RR) arbiter module selects a VC request among the active VC requests and follows RR arbitration. A credit signal (*VC-full*) is required for each VC to close it in the case of a full condition. The *VC-full* module, shown in Figure 9.7 reserves a slot for each VC. Assume that S0 to S15 represent the cell states of the *Slot-State* table, and V0 to V3 represent the cell states of the *VC-State* table. The state of *VC-full* signal is determined by the following logic equation.

$$\text{VC-full}[i] = \left((S_0 + S_1 + \ldots + S_{15}) - (V_0 + V_1 + \ldots + V_3) - V_i\right) \quad \text{where } i \in \{0 \ldots 3\} \tag{9.1}$$

A circuit-based DAMQ input-port architecture, Effective Dynamic Virtual Channel (EDVC) is presented by Oveis-Gharan and Khan [15]. The EDVC mechanism utilizes the common features of the DAMQ input-port to create

a dynamic flow control. The EDVC mechanism has some drawbacks. First of all, the *read-pointer* and *write-pointer* architectures grow large with the size of the input-port buffer. For example, if the size of the input-port buffer increases n times, the multiplexers employed in EDVC *read-pointer* and *write-pointer* will grow exponentially (n^2 times), which will in turn increase the hardware overhead and the critical path delay, which has a direct effect on the speed of the EDVC router. The second drawback of EDVC design is its higher latency at lower flit-injection rates. The authors later presented a Rapid Dynamic Queue (RDQ) approach to overcome these drawbacks [16]. RDQ VC structure and organization is similar to EDVC mechanism where RDQ demonstrates much higher performance under different injection rates and consumes economical hardware in all the configurations of an input-port buffer.

9.2.1 Heterogeneous NoC Router

A number of researchers have focused on the design and organization of routers that have direct impact on the NoC power, performance and area [17–19]. Virtual Circuit Switching (VCS) is proposed by Jiang *et al*. [17]. Their approach intermingles the Circuit Switching (CS) and Packet Switching (PS) to obtain low latency and power consumption in NoCs. They have also proposed a path allocation algorithm to determine VCS connections and CS connections in a mesh-connected NoC. VCs in the VCS approach are exploited to form a number of VC connections by storing the interconnect information in routers. Flits can directly traverse the routers by using only the Switch Traversal (ST) stage. The main advantage of the VCS approach is that it can have a similar router pipeline as a circuit switching router, and can have multiple VCS connections to share a common physical channel. Basically, VCS connections cooperate with PS connections and CS. When flits on CS or VCS connections arrive at routers, crossbar switches are immediately configured so that the CS or VCS flits can bypass directly to the ST stage. When there is no CS or VCS flit, the corresponding ports of crossbar switches are released to PS connections. The VCs of routers in VCS connections are preconfigured in such a way that they are only connected to particular downstream virtual channels. Crossbar switches are preconfigured during the switch allocation (SA) stage before VCS flits require to pass through. As VCS connections are established over virtual channels, a physical channel can be shared by n VCS connections where n is the number of virtual channels. Other communications competing for the same physical channel must be conducted through the packet switching. There are some drawbacks of the VCS approach. First of all, VCS router architecture requires more hardware as compared to conventional wormhole routers because they must have the circuitry for both packet and circuit switching. Moreover, extra hardware is needed to distinguish VCS packets from PS packets. Moreover, despite the extra hardware of VCS, the approach is only efficient in deterministic communications suitable

to application-specific NoCs. However, our proposed NoC routers in this chapter are efficient for any type of NoC communications. Moreover, they improve the performance while their architectures are simpler and accommodate lower hardware overhead.

Another heterogeneous NoC router architecture has been introduced by Ben-Itzhak *et al*. [18]. They exploit a shared-buffer technique to handle the bandwidth heterogeneity of an NoC link. Their approach reduces the number of shared buffers required for a conflict-free router without affecting the performance. Reducing the number of shared buffers also reduces the crossbar size and overall it has lower chip area and power consumption. The heterogeneous router supports different capacities and different numbers of VCs for each link while keeping the router frequency fixed. Their router architecture utilizes serial-to-parallel converters in order to store the incoming flits in different input buffer slots at each link clock cycle, and parallel-to-serial converters can be used in order to transmit several flits in a Time-Division Multiplexing (TDM) way depending on the link frequency. The shared-buffer technique presented by Ben-Itzhak *et al*. optimizes the number of shared buffers related to arrival and departure conflicts discussed by Ramanujam *et al*. [19]. The drawbacks of the VCS approach discussed earlier can be considered for the heterogeneous approach. In other words, the heterogeneous router architecture is tailored for specific types of NoC communication. However, NoC has emerged as a scalable and global communication architecture to address the SoC design challenges. Its features enable it to be easily expandable and it can provide services for a variety of SoC communications. The heterogeneous feature of this approach violates the scalability and reusability of NoC and additional research and investigation for heterogeneous NoC routers is required.

9.2.2 Router Buffer Organization

A number of researchers have focused on the design and organization of router buffers due to their close relationship with NoC power, performance and area. Complex router architectures are efficient for certain NoC configurations or data flow circumstances. CUTBUF NoC router architecture has been presented by Zoni *et al*. [20], where VCs are dynamically assigned to an input-port depending on the actual input-port load and reusing of each queue by packets of different types. Their approach significantly reduces the number of physical buffers in routers, saving area and power without affecting the NoC performance. It is assumed that a conventional VC can be re-allocated to a new packet only if the tail of its last allocated packet has left. However, a reserved VC in CUTBUF protocol is released when either the tail flit traverses the same pipeline router stage, or when the related packet gets blocked. In this way, the CUTBUF scheme increases the buffer utilization as well as preventing HoL blocking, where a VC can be re-allocated to a new packet under certain specific conditions. The newly allocated packet to an occupied

VC will not be blocked because the previous packet is guaranteed to traverse the router. To implement this mechanism, the following condition is checked to allow a new packet to allocate a non-empty VC. The earlier packet will be leaving the router when its tail flit is stored in the input buffer VC, and the downstream router has enough credits to store all the flits of the VC.

A conventional NoC protocol has also been investigated in the Packet-Based Virtual Channel (PBVC) approach [21]. A VC in a PBVC scheme is reserved when a packet enters the router and released when the packet leaves. A VC will hold the flits of only one packet at a time that will remove the HoL blocking. The PBVC technique is more efficient for DAMQ-based schemes where an input- or output-port employs a centralized buffer whose slots are dynamically allocated to VCs. The experimental results have shown that for HoL-specific traffic, the average latency and throughput are improved in the PBVC approach as compared to conventional DAMQ-based NoCs. Yung-Chou and Yarsun have also presented a new DAMQ-based buffer organization called DAMQ-MP that can accommodate multiple packets greater than the available number of VCs [22]. Their approach can solve certain on-chip communication issues, such as heavy network congestion or short packets, to improve the performance. They have introduced the DAMQ-MP data flow as follows. Whenever the tail flit of a packet enters the input buffer via one of the VCs, this packet releases the VC by alerting a notification signal to the upstream router. Therefore, a new packet from the upstream router can be sent out and enters the free VC. In other words, the new packet does not need to wait until the tail flit of the previous packet gets out of the VC. This approach can save a lot of time, especially when the routing computation for the arriving packets is high. The above data flow organization means that the DAMQ-MP is efficient in various cases, including small packets, large buffer capacity, heavy-congested traffic (such as a saturated network), and a small number of VCs. As one may notice, the DAMQ-MP approach is the reverse of the PBVC approach. The DAMQ-MP approach lets a VC accept a new packet when the VC is not empty, whereas the PBVC approach prevents a VC from accepting a new packet when the VC is not empty. In fact, both approaches are efficient under different schemes and traffic patterns.

RoShaQ, another interesting router architecture that allows the sharing of multiple buffer queues, has been presented by Tran and Baas [23]. In this approach, each input-port allocates one buffer queue and shares all the remaining queues. The router architecture maximizes buffer utilization by sharing multiple buffer queues among input-ports. Buffer sharing reduces the packet stall time at input-ports. The RoShaQ is able to achieve higher throughput when the network load is heavy. For light traffic load, the RoShaQ router achieves low latency by allowing packets to effectively bypass the shared queues. In this way, their proposed router achieves higher performance when the traffic load becomes heavy and for low-load traffic. A RoShaQ NoC is also deadlock-free, as for a light load the packets normally bypass the shared queues and RoShaQ acts like a wormhole router to meet

the deadlock-free condition. In the case of a heavy load, when a packet cannot win the output port, it is allowed to occupy only a shared queue which is empty or contains packets having the same output port. Clearly, in this case the RoShaQ works as an output-buffered router which is also shown to be deadlock-free. The approach combines two switching mechanisms, VC-based and simple wormhole, to improve the performance. When the first packet enters an input-port, it is serviced through wormhole switching without involving VC pipelines. However, when the second packet enters (assume the first packet is still there), it is serviced through VC-based switching involving VC pipelines. RoShaQ routers must accommodate both circuits related to wormhole switching and VC-based switching, leading to more hardware. Moreover, an extra circuit is needed to distinguish wormhole packets from VC-based packets. Therefore, the RoShaQ router architectures become more complex, with more hardware as compared to conventional VC-based routers. The performance improvement of RoShaQ is due to more hardware overhead rather than an efficient or novel architecture.

9.2.3 NoC Router Arbiters

Arbiters are commonly used to allocate and access shared resources. Whenever a resource (such as a buffer, channel or a switch-port) is shared, an arbiter is required to assign the access to the resource at a particular time. A wormhole *v*-VC router was presented in Section 9.1, where the *router arbiter* module has a *switch allocator* consisting of two sets of simple arbiters. A simple arbiter arbitrates among a group of requesters for a single resource, as illustrated in the form of a symbol for an *n*-input in the right side of Figure 9.5. The arbiter accepts *n* requests (r_0, r_1,…, r_{n-1}), arbitrates among the asserted request lines, selects an r_i for service, and then asserts the corresponding grant line g_i. Arbiters can be categorized on the basis of fairness (weak, strong or FIFO) arbiters [1, 24]. For a weak fairness arbiter, every request is eventually granted. The requests of a strong fairness arbiter are granted equally often. The requests of FIFO fairness are granted on a first come, first served basis. In terms of priority, the arbiters can be grouped into fixed and variable priority architectures. For a fixed priority arbiter, the priority of requests is established in a linear order. Round Robin (RR) is a well known variable priority arbiter. The functionality of an RR arbiter can be explained in this way: a request that is most recently granted will have the lowest priority in the next arbitration cycle. The RR arbiters are simple, easy to implement, and they are also starvation-free. When the input requests are large in number, the structure of an RR arbiter grows, requiring larger chip area, higher power consumption, and longer critical path delay [1].

The micro-architectures of popular RR arbiters such as Matrix and Round-Robin (RoR) have been extensively presented and evaluated [1, 25, 26]. Matrix and RoR arbiters have the same functional behavior and perform strong fairness arbitration. The Matrix arbiter is claimed to be useful for a small

number of inputs as it is fast and economical. Fu and Ling evaluated and compared the RoR and Matrix arbiters in terms of resource, performance and power consumption for an FPGA platform [25]. They concluded that the Matrix arbiter consumes more resources for the same power consumption and it can process data quickly as compared to the RoR arbiter. The same evaluation is made by Oveis-Gharan and Khan [26]. They also presented a strong fairness round-robin arbiter design, IRR. Their approach is simpler, faster and consumes less hardware as compared to the RoR and Matrix arbiters. The IRR arbiter is employed later in our NoC router design presented in Section 9.3. A High-speed and Decentralized Round-robin Arbiter (HDRA) has been presented by Lee *et al*., and is illustrated in Figure 9.8 [27].

Each boxed circuit represents a filter circuit whose main components are a D-type flip-flop and a multiplexer. The circuit filters out the input without request or the one whose request has already been granted in the current arbitration cycle. The unfiltered inputs with their requests participate in the arbitration again in the next cycle by resetting their corresponding D-type flip-flops to low (or at zero), which is done by enabling the *ack* signals from high to low level. The HDRA arbiter will reset itself asynchron ously by the *self-rst* input. The *sys-rst* indicates the system reset signal and is used initially before each arbitration cycle for all the requests. A four-input HDRA arbiter has a simpler architecture because the *act*, *rnext* and *self-rst* are connected together. The HDRA architecture has been used in various applications, e.g. in the router model of simulation framework implemented by Guderian and others [28].

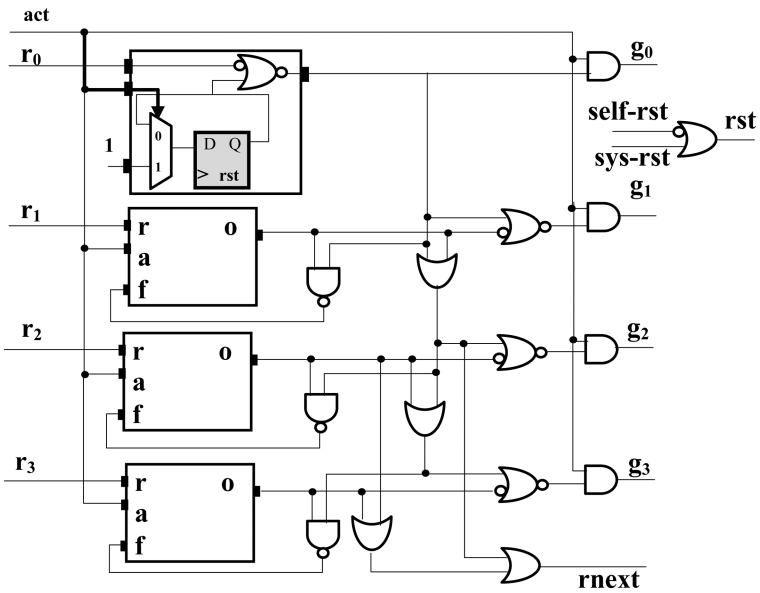

FIGURE 9.8
Four-input HDRA architecture.

9.3 Low-Power NoC Router RDQ-IRR

The novel NoC router micro-architecture presented in this chapter consists of input-ports, an arbiter and a crossbar switch, as illustrated in Figure 9.9b. The router employs VC organizations shown in Figure 9.9c, which is an amended version of Rapid Dynamic Queue (RDQ) employed in input-port modules [16]. An Index-based Round Robin (IRR) arbiter is employed for arbitration, as shown in Figures 9.9a.

9.3.1 RDQ: Rapid Dynamic Queue Input-Port

The RDQ mechanism utilizes the common features of a DAMQ input-port to create a dynamic flow control [16]. For example, The VC identification (*VC-ID*) is saved with the flit in the input-port buffer, as shown in Figure 9.10, which is used to issue the VC *request* signals for router arbiter. A small *Slot-State* table (having Boolean values) is used to manage the input-port buffer mechanism. Each *Slot-State* bit corresponds to a buffer slot representing its occupancy state. The *read-pointer* points to the occupied locations of buffer per clock cycle. The *write-pointer* points to empty location of buffer until a write occurs. When a flit faces blockage, its VC will stop issuing requests and receiving new flits. Moreover, the address of the header flit of a blocked VC is stored in the *Blocking-Header* table. The highlighted locations in Figure 9.10 illustrate that the VC2 is blocked, and its header flit address is stored in the *Blocking-Header* table. To maintain the order of flits associated with the blocked VC, the following steps are performed. When the blockage of a VC (e.g. VC2) is removed, and the IRR *read-pointer* points to the header flit (location 2), the VC request become free, leading the *read-pointer* to read the whole buffer once and send out the flits of the freed VC (VC2). Then the VC starts accepting any new flits. *VC-full* and *VC-block* signals assist during the blockage conditions, as illustrated in the blocking circuit associated with a VC shown in Figure 9.11. Each VC has a *VC-full* signal, which is issued to the upstream router and represents the state of the VC to accept a new flit. The *VC-block* signal comes from the arbiter, indicating that the VC flit cannot exit the router. An asserted *VC-block* signal causes the VC to stop requesting (*stop-req* signal is asserted) the arbiter and will not receive any new flit. The RDQ mechanism also prevents a data flow condition where a reserved empty VC can become closed. Various conditions associated with the VC closing and requesting operations are listed in Table 9.1 below.

The closing and requesting operations are actuated by the *VC-full* and *VC-req* signals respectively. There are a number of conditions determining the state of the *VC-block*. As mentioned earlier, the *VC-block* becomes set when the VC cannot succeed in winning a free VC of the downstream router input-port. Consider the first condition, where the *VC-block* is reset, indicating no blockage in communication. For the second condition, the

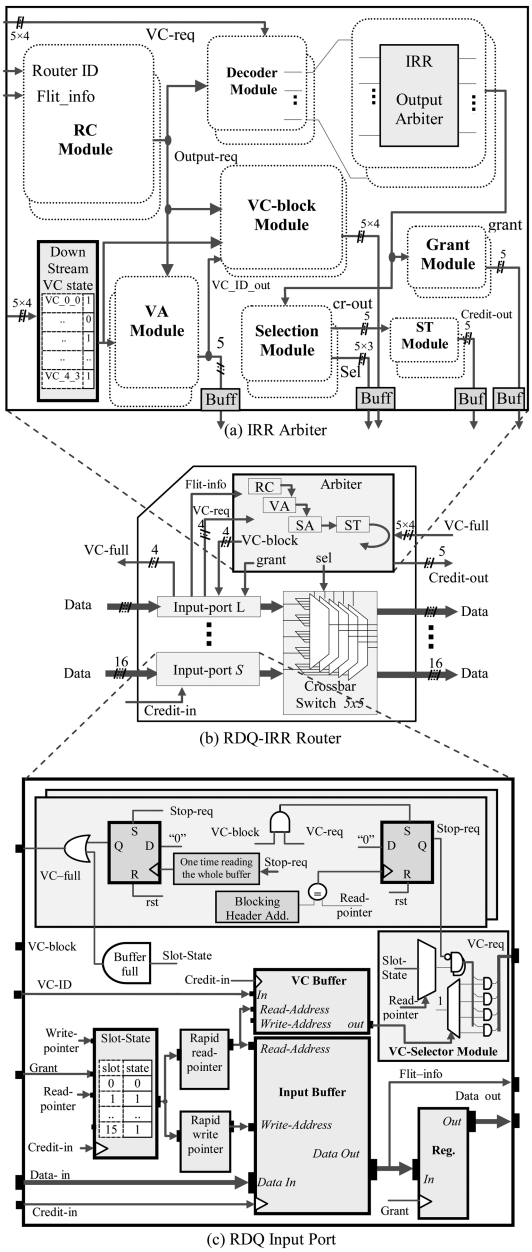

FIGURE 9.9
RDQ-IRR NoC router micro-architecture.

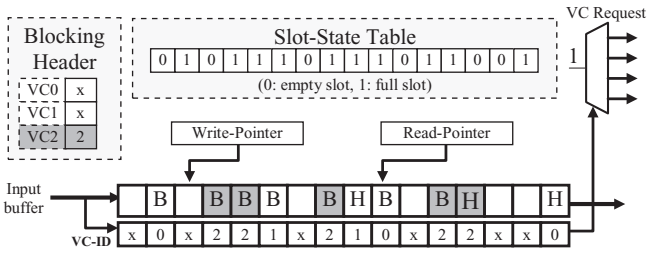

FIGURE 9.10
RDQ input-port operation.

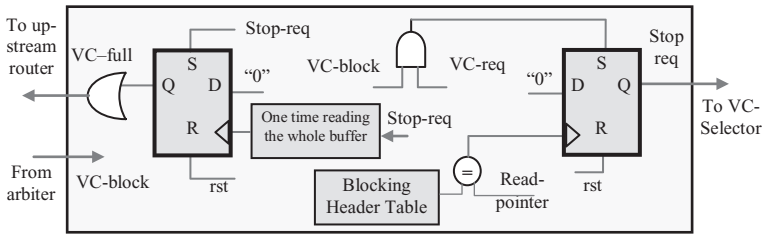

FIGURE 9.11
Blocking circuit associated with a VC.

TABLE 9.1

VC Closing and Requesting Operations of RDQ Mechanism

Cond.	*VC-Block*	*Stop-Req*	*VC-Req*	*VC-Full*
1	0	0	X: Normal 1: *read-pointer* points to a flit 0: *read-pointer* does not point to a flit of the VC.	X: Normal 1: buffer is full. 0: buffer is not full.
2	1	0	0: *read-pointer* points to no flit of the blocked VC.	X: Normal 1: buffer is full. 0: buffer is not full.
		1	0 to1 then returns 0: *read-pointer* points to a flit of blocked VC. The flit location is stored in blocking Header table.	1: stop incoming flit for the VC
3	1 to 0	1: *read-pointer* ≠ *Header flit Address*	0: stop request	1: stop incoming flit for the VC
		0: *read-pointer* = *Header flit Address*	Normal	1: stop incoming flit for the VC
		0: *read-pointer points to the occupied slots*	Normal	Normal

VC-block becomes set. In that case, the *read-pointer* does not point to a flit of the VC, and the VC is open for the incoming flits. As soon as the *read-pointer* points to a flit of the VC, the *VC-req* switches from 0 to 1 and leads to the *stop-req* signal being set and consequently the *VC-req* returns to 0. Meanwhile, the flit location is stored in the *Blocking-Header* table. In other words, the router stops receiving flits for that VC as well as requesting the arbiter. The order of VC flits inside the port buffer is kept until the blockage exists. In the third condition, the *VC-block* switches from 1 to 0, i.e. the VC can succeed in the arbiter. In this case, the *read-pointer* does not point to a flit of the blocked VC, the VC issues no request to the arbiter and receives no incoming flit. The *read-pointer* continues pointing to the occupied slots until it reaches the header flit of the blocked VC, as illustrated in the upper part of Figure 9.9c. At this point, the VC can only issue a request to the arbiter so that its flits can exit the router without allowing any new flit to enter the VC. The *read-pointer* continues to point to the occupied slots until it reads the whole buffer. When it points to the least significant occupied slot of the buffer two times, the closing and requesting operations of the VC return to the normal condition.

9.3.2 NoC Router Arbitration

The router arbiter can perform arbitration through four pipelined stages: Routing Computation (RC), Virtual-channel Allocation (VA), Switch Allocation (SA), and Switch Traversal (ST), as illustrated earlier in Figure 9.4. The structure of RC can be a simple multiplexer, as shown in Figure 9.12, due to XY routing considered for communication in a 2D mesh NoC. The destination ID, *Flit_info* (stored in the header flit of a packet), is compared with the ID of the current router. Then, the requested output-port is generated at the RC output according to XY routing algorithm. The structure of VA can be a fixed-priority arbiter, as shown in Figure 9.13. The first free VC of the downstream router input-port is selected in an ascending order. The SA module includes decoder, IRR *output-arbiter* modules, and the post-SA circuits (*Selection* and *Grant* modules) as shown in Figure 9.9a. The decoder module was introduced in Section 9.1.4 above, and the remaining modules are discussed in the following sections.

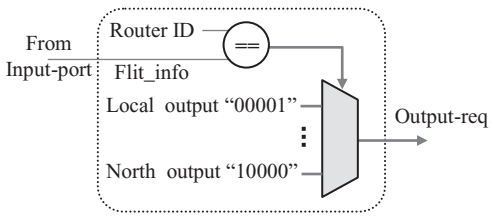

FIGURE 9.12
RC generates a free Output VC.

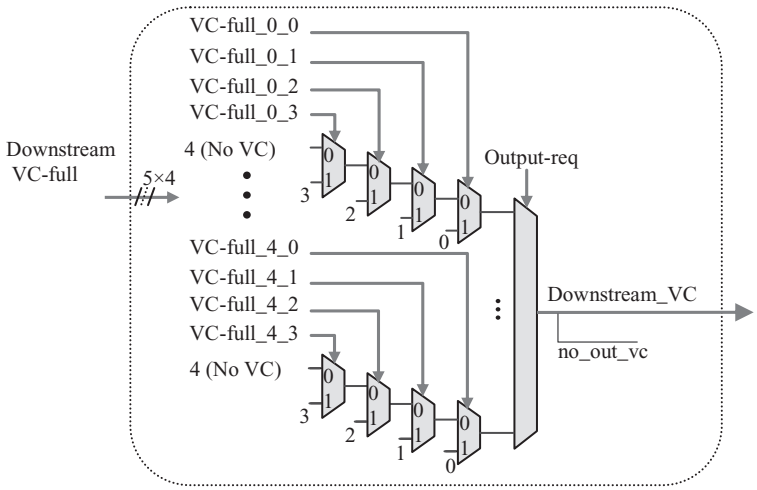

FIGURE 9.13
VC Allocator generates a free downstream VC.

9.3.2.1 VC Arbitration

In our router design, the VC arbitrations are implemented in the input-port by means of *VC-Selector* modules. We employ a central buffer to store all the VC flits of an input-port and arbitration is performed in one clock event. When a *grant* signal is issued to an input-port, the *read-pointer* is pointing to the winner VC flit, or the winner flit is loaded to the output-port of the buffer. The input-port micro-architectures of Figures 9.7 and 9.9c illustrate this scheme, where the *VC-Selector* chooses a VC for requesting to the arbiter, and simultaneously the flit of the VC is loaded to the buffer output. The *VC-req* signals in the LLD input-port of Figure 9.7 are used to generate the *read-pointer*, and these signals for the RDQ input-port of Figure 9.9c are employed to cater for blocking. One may think that the *VC-Selector* modules can be accommodated in the switch allocator, and the *VC-req* signals can be fed back to the input-port. The problem of this design is the dependency of input-port and arbiter on each other in terms of hardware and speed, as part of the critical paths of input-ports is shared with the arbiter. To prevent such sharing, we implement the VC arbitration as part of the input-port, and remove the *input-arbiter* modules from SA, as illustrated in Figure 9.14. In this way, the input-port and arbiter will operate independently.

9.3.2.2 Post-Switch Allocation

The *Selection* module generates the credit and selection address of the crossbar multiplexers related to an output-port of the router. Its micro-architecture is presented in Figure 9.15. When g_in3_out1 is asserted, the circuit generates the number 3 at the Sel output that leads input-port 3 to be connected

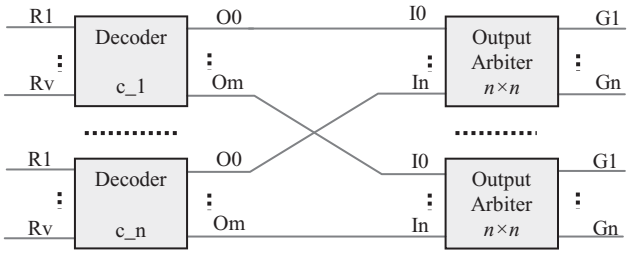

FIGURE 9.14
$n \times m$ SA micro-architecture for n inputs, m outputs and v VCs per input-port.

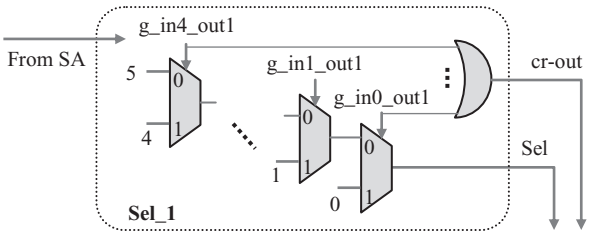

FIGURE 9.15
Output-port 1 of selection module.

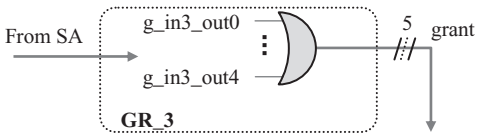

FIGURE 9.16
Input 3 of grant module.

to output-port 1 of the crossbar switch (see Figure 9.5). Moreover, a credit signal, *cr-out*, is issued to the ST module. Then the ST module issues a credit, *credit-out* signal to the first output-port to store the flit in the associated downstream VC at two clock events later. Figure 9.16 shows the Grant module that generates the *grant* signal associated with the third input-port of the router. The asserted g_in3_out1 allows the flit of third input-port to transmit via the crossbar switch and pass through the first output-port.

9.3.2.3 IRR Output Arbiter

An Index-based Round Robin (IRR) arbiter employs a least recently served priority scheme and achieves strong fairness arbitration [26]. The IRR arbiter is employed by the *output-arbiter* of the SA module (Figure 9.9a). The IRR arbiter has smaller arbitration delay, lower chip area and it also consumes less power as compared to other RR arbiters. Figure 9.17 shows the

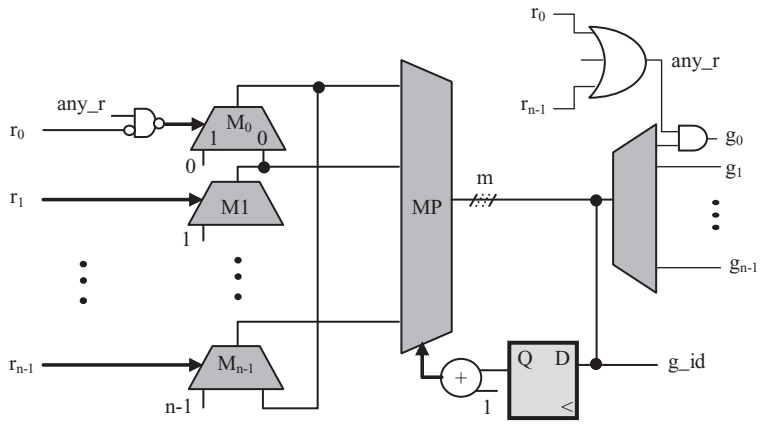

FIGURE 9.17
n-input IRR arbiter, where $m = log_2(n)$.

micro-architecture of an IRR arbiter that takes one clock cycle for arbitration. The index of the winner asserted request is switched to the output as the index of *grant* and *g_id*. Then the *g_id* is decoded to create the *grant* signals. If the next index of the granted request is employed for the next priority selection, the current granted request receives the least priority, and its next request receives the highest priority among all the requests. To accomplish it, the *g_id* array is stored in a register, whose output is incremented and connected to the selection port of the multiplexer (MP), as shown in Figure 9.17. In this way, the arbiter follows either the least recently served priority or a round-robin scheme. To keep the priority unchanged, the priority register output is fed back into a multiplexer to cater for no request. It guarantees strong fairness arbitration for our IRR design.

9.3.2.4 FIFO Arbitration and VC Selector

The selection of a VC from the input-port VCs in LLD [4] and ViChaR [3] is implemented in the *VC-Selector* module by employing a Round Robin (RR) arbiter to ensure fairness and avoid traffic starvation, as shown in Figure 9.7. However, the selection of a VC from the input-port VCs in an RDQ mechanism happens in the data flow process. It is based on the location of *read-pointer* and the state of data in the input-port, as indicated in Figure 9.9c. The *VC-req* is updated according to the location of the flit data in the input-port buffer. This feature of RDQ *VC-Selector* requires the arbitration among the input-port VCs to follow a FIFO fairness priority [24].

9.3.2.5 Rapid Read and Write Pointers

The *rapid-read-pointer* module only points to the occupied locations and the *rapid-write-pointer* points only to the empty locations of the input-port buffer

shown in Figure 9.9c. An RR priority scheme is followed, where a slot that is just read has the lowest priority for the next cycle [29]. In this way, each asserted bit of *Slot-State* table is pointed per clock cycle in an ascending and circular order. Figure 9.17 can also be used to illustrate the micro-architecture of an n-bit *rapid-read-pointer*, where the requests (r_0,r_1 …, r_{n-1}) and *gr_id* are substituted with the *Slot-State* content and *rapid-read-pointer* output respectively. A similar *rapid-read-pointer* architecture is proposed for *rapid-write-pointer*, except the places of input-ports of each M_0 to M_{n-1} multiplexers are exchanged. Moreover, the *rapid-write-pointer* output is directly connected to the selection port of *MP*. In this way, the *rapid-write-pointer* module points to a *Slot-State* empty slot and stop there until the slot becomes full, then it jumps to next empty slot on the following clock cycle and in an RR priority order.

9.3.3 Pipelined RDQ-IRR Router

The buffering pipeline of the RDQ-IRR router consumes two clock events in a squeezed scheme if the arbitration takes one clock event (step), as illustrated in Figure 9.3a. We mentioned earlier that the LLD and ViChaR port models are table-based and their router pipeline takes four clock events for one clock event arbitration, as shown in Figure 9.3b. The arbitration stages in our RDQ router implementation follow the timing diagram of Figure 9.4. The pipeline stages for LLD and ViChaR routers take two clock events longer than that of the RDQ router. Each head flit of a packet must proceed through the stages of Routing Computation (RC), Virtual-channel Allocation (VA), Switch Allocation (SA), and Switch Traversal (ST). The micro-architectures of the aforementioned stages are described here.

9.3.3.1 Pipeline Stages Micro-architectures

The arbiter sub-modules, such as RC, VA, Decoder, *output-arbiter*, Selection (see Figure 9.15) and Grant (see Figure 9.16) illustrate that the routing from the inputs of RA to the outputs of Selection, Grant or VA modules are not sequential. In other words, the outputs of Selection, Grant or VA modules are not sequential. Therefore, the RC, VA and SA stages can determine the winner VC and winner input-port of each output-port in a single iteration. In this way, the data flit related to winner VCs can exit the router at the following clock event, and the VCs can make new requests. There is no intermediate register among the paths from the inputs of RC to the outputs of SA, and the registers related to the *output-arbiter*s only keep the state of SA for the following clock event (Figure 9.17 and 9.14). In this way, all the arbitration stages are performed in one clock event. At the end of the SA stage, if the output of the *output-arbiter* associated with a VC is active, the associated downstream router VC, *grant*, and crossbar addresses are issued. The following additional actions are performed for the head and tail flit:

- If the request is from a header flit, the associated RC and VA outputs are also reserved.
- If the request is from a tail flit, the reservations are also removed.

The *credit_out* (credit) signals associated with *grant*s are issued at the ST stage to store the data in the downstream input-port at the following clock event. The *cr-out* signal, generated in the *Selection* module of Figure 9.15 is repeated in the ST module at the following clock event.

9.3.3.2 RDQ-IRR Router Arbiter

In conventional DAMQ routers (such as LLD), when the requested output of a VC is blocked (i.e. no output credit), the arbiter issues a block signal that causes the input-port to select any other available VC for service. No output credit means the corresponding *VC-full* signal of the downstream router is set. However, the RDQ-IRR arbiter issues the block signal under two conditions, i.e. either on losing switch arbitration to some other input-port or having no output credit. This approach requires some extra hardware in the arbiter of the RDQ router as compared to an LLD router. The LLD arbiter updates a table at each clock cycle, where the table consists of an array of registers where each bit represents the blocking state of a VC of the input-ports. Figure 9.18a shows a typical LLD blocking circuit of one-bit register associated with VC1 of input-port 3. If there is no output credit at each clock cycle, the *VC-block* signal is asserted, otherwise it is de-asserted. In our RDQ-IRR approach, the blocking circuit requires extra hardware (one OR gate per VC), as shown in Figure 9.18b. When the requested output is closed or the input VC loses arbitration to other input-ports, the *VC-block* is asserted; otherwise, it remains de-asserted.

A specific situation can arise in our RDQ-IRR NoC, when a packet blockage condition travels back to relevant up-stream routers related to the blocked packet. However, our VC organization maintains the order of flits associated with the blocked packet. The blockage situation can lead to chain-blocking and a back-pressure is created in the NoC [16]. The implementation of this situation has direct affect on the performance of RDQ-IRR NoC, specifically at low injection rates.

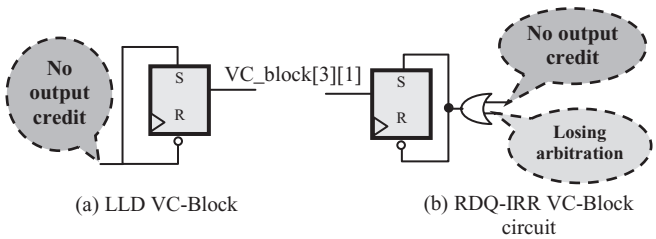

FIGURE 9.18
VC-block module associated to VC1 of input-port 3.

9.3.4 Simpler Communication in RDQ-IRR Router

The RDQ-IRR data flow mechanism can be easily explained if there is no contention in the NoC communication. In such a situation, the RDQ-IRR buffer works like static VCs. We employ asynchronous communication among the routers. The following functions describe the working of RDQ-IRR given in Figure 9.9 according to the timing diagram of Figure 9.3a.

Flit Arrival State: Clock-edge 1

1. A *Credit-in* signal causes the incoming flit (e.g. flit F1) and its *VC-ID* to be saved in a slot pointed by the *write-pointer*. The corresponding bit of the *Slot-State* table is also set.
2. When the *read-pointer* points to an occupied slot, a request signal is issued by the *VC-Selector* module according to the *VC-ID*. The *read-pointer* also causes the flit to appear at the output of the central buffer and the flit information (*flit-info* signals) is read by the arbiter.
3. The RC module of the arbiter reads the flit information and determines its requested output-port.
4. The VA module reads the RC output and determines if a free VC in the downstream router input-port is available.
5. If a VC is available then the SA module (decoder and *output-arbiter*) reads the RC output and *VC-req* signals and performs arbitration among the winner VCs for the output-ports.
6. The *Selection* module reads the SA output and determines the associated address for the crossbar switch. The *Grant* module also reads the SA output and sets the *grant* signals for the winner VCs.
7. The *VC-block* module generates the *VC-block* signals under two conditions:
   - Losing switch arbitration to some other input-ports (i.e. *VC-req* is set and its associated *grant* is reset).
   - No output credit. It is determined by RC, VA and *Downstream-VC-state* outputs as illustrated in the *VC-block* module in Figure 9.9.
8. The ST module reads the *cr-out* signals and keeps their record to issue *credit-out* signals after two clock events.

Flit Departure State: Clock-edge 2

1. A *grant* signal causes the associated flit to exit the input-port and the corresponding bit of the *Slot-State* table is reset.
2. The *Sel* signals cause the crossbar to switch the input-ports to their associated output-ports.

AU: The list in the "Flit Departure State: Clock-edge 2" section has been renumbered. Please confirm.

3. The *VC-ID-out* signals carry the *VC-ID* of the flit.
4. In case of no output credit or loss of arbitration, the *VC-block* signals cause the associated VC to become blocked (no *grant* signal).

Credit and Next Flit Arrival State: Clock-edge 3

1. The *credit-out* signals are issued by the ST module so that the transferred flits are stored in the downstream router input-ports' buffers.
2. Steps (i) to (viii) are repeated for the next incoming flits.

9.3.5 RDQ NoC Design

NoC architectures are commonly presented in Globally Asynchronous Locally Synchronous (GALS) design style [30], and we have also followed the GALS style in our router design for 2D-mesh NoC. For GALS-based NoCs, routers that are locally synchronous are easier to design where the NoC architectures are globally asynchronous. It means that routers are independent of the NoC in terms of their clock, and the faster clock rates of routers lead to a faster NoC. The NoC router circuitry can be divided into synchronous and asynchronous components. The speed of the asynchronous components is almost proportional to the semiconductor technology of silicon. The state of synchronous components changes with their clock signal, and the speed of these circuits depends on their maximum possible clock rates (f_{max}). The f_{max} is determined by the critical path, which is related to the slowest logic path in the circuit. If a router consists of some pipelined components, the synchronous components determine the speed of the router. The synchronous components of an NoC router are those that utilize synchronous data buffers (such as registers or RAM). The crossbar switch has an asynchronous architecture, as illustrated in Figure 9.5, and it does not affect the speed of the router. However, input-port and arbiter utilize synchronous buffers to temporarily store data flits or information and affect the speed of a router. These two components are investigated in terms of their pipelined stages and critical path delay in the following sections.

We propose an efficient and fast DAMQ router architecture called RDQ-IRR that employs RDQ-based input-ports and IRR arbiters in NoC routers. The features of our approach are listed below:

- The arbitration among the VCs of RDQ-IRR routers follows a FIFO style.
- The arbitration in switch allocation module of RDQ-IRR routers ensures strong fairness and avoids traffic starvation.
- The RDQ-IRR router arbiter architecture benefits from easy separation or pipelining. It is very fast, which makes it a proper choice for NoC applications where latency is critical.

- The flit arrival/departure in an RDQ-IRR router is faster than that of other table-based DAMQ routers. In addition to saving one clock cycle for each flit arrival/departure at the input-port, the critical path delays of RDQ-IRR router components are lower as compared to those of other DAMQ routers.
- In addition to higher performance metrics, RDQ-IRR routers are simpler and faster, as well as consuming less power and chip area.

9.4 RDQ-IRR-based NoC Experimental Results

We set up eight types of NoCs based on the conventional LLD, ViChaR [3, 4, 11] and novel EDVC [15] and RDQ approaches discussed in this paper. The hardware structure and performance of these NoCs are evaluated and compared to illustrate the efficiency of our RDQ-based approaches. The NoC architectures follows GALS scheme and the links between routers are assumed to have no effect on the performance and hardware requirements of the NoCs. The architectural details of eight NoCs are presented. The RDQ-IRR NoC employs EDVC based RDQ input-port [15] and IRR arbiters [] in its routers. The remaining NoCs utilize LLD [4, 11] and ViChar [3] input-ports and one of the RoR, Matrix or HDRA arbiters in their router structures, and they are called LLD-RoR, LLD-Matrix and LLD-HDRA, as well as ViChaR-RoR, ViChaR-Matrix and ViChaR-HDRA, according to the input-ports and arbiters utilized in their routers. The crossbar switch module has identical structures for all the eight NoCs due to the same input-port buffer width (16 bits) and same 2D-mesh NoC topology.

9.4.1 RDQ-IRR-based NoC Hardware Requirements

We have evaluated all the above-mentioned eight NoCs in terms of main hardware characteristics such as power consumption, chip area, and speed, which are determined by Verilog implementation and employing Synopsys Design Compiler. ASIC technology libraries such as 15 nm NanGate are used to obtain the evaluation results [31]. The setup constraints and semiconductor technology parameters, such as global operating voltage of 0.8 V and time period of 1 *nsec*, are applied for all the NoC components evaluated. The width of the slot buffer is equal to the flit size of 16 bits. The hardware characteristics of various modules of all the NoC routers are evaluated, and the results are presented in Tables 9.2 and 9.3. Table 9.2 results are organized according to number of VCs and buffer slots utilized in each input-port. For example, the input-port (*port_LLD_HDRA_4 v_4 s)* related row in the table provides chip area, power and critical path delay of an LLD-HDRA input-port utilizing four VCs and having four slots in its buffer. The LLD-HDRA

input-port is based on LLD DAMQ utilizing HDRA arbiter in its *VC-Selector* (see Figure 9.7).

EDVC and RDQ input-ports do not utilize separate *VC-Selector*. An important characteristic of high-scaled CMOS technologies (such as 15 nm technology) is that the static (leakage) power supersedes the dynamic power at 1 GHz frequency. For example, the average dynamic power of input-ports is almost 10% of their total power, which indicates that the power is more or less proportional to the hardware overhead of the design rather than its functionality. In other words, more cells consume more static power and the synthesis results given in Tables 9.2 and 9.3 also confirm this. The ViChaR-based routers are expensive in terms of hardware cost as compared to other routers. The higher cost of ViChaR is due to the large control table (even bigger than that of LLD) [15]. The LLD- and ViChaR-based routers that utilize Matrix arbiters consume more chip area but have lower critical path delay. This is due to the fact that the Matrix arbiter uses more registers than other arbiters. For example, the number of registers employed by Matrix, RoR, HDRA and IRR arbiters is 28, 8, 8 and 3 respectively. The EDVC- and RDQ-based input-ports employ optimal hardware as compared to the other input-ports with the same number of VCs and buffer slots. This is due to their lower hardware overhead, as discussed earlier in this chapter. The RDQ router architecture has extra hardware in terms of an AND gate and some registers per VC (see Figure 9.11). The registers are used in the Blocking Header table. In this way, EDVC input-ports consume less hardware than RDQ input-ports for four and eight buffer slots. However, RDQ read- and write-pointers scale with the index of slot number, as opposed to those of EDVC, that scale with the number of slots. In this way, RDQ input-ports become optimal as compared to EDVC input-ports for 16 and 32 buffer slots.

The main difference among the various arbiter structures is the RR arbiters used in SA modules. A few extra OR gates are used in RDQ and EDVC VC-block modules. For example, the *IRR_8v* arbiter utilizes an IRR arbiter in its SA module and five OR gates in VC-block modules, as compared to the *RoR_8v* arbiter that utilizes an RoR arbiter in its SA module. This trend among the arbiter characteristics can be observed in Table 9.3. The Matrix-based arbiters consume more chip area but have lower critical path delay as compared to RoR- and HDRA-based arbiters. However, *IRR_4v* and *IRR_8v* arbiters consume optimal power and chip area as compared to other arbiters. An important point relates to the identical critical path delays for 4-VC and 8-VC arbiters, as shown in the results of Table 9.3. This is due to the same RR arbiters (5-input) being used in both 4-VC and 8-VC arbiters. Figure 9.14 shows that the size of RR *output-arbiters* (used in SA) is equal to the number of input-ports of the router.

The crossbar module does not have any critical path delay as it does not utilize any register in its structure. One can observe that the critical path delays of the arbiters shown in Table 9.3 are less than those of the corresponding input-ports of Table 9.2 despite their higher area and power

TABLE 9.2

Input-port hardware characteristics

			ASIC Design 15 nm NanGate Library		
VCs	**Slots**	**Input-Port Model**	**Area (μm^2)**	**Power[a] (uW)**	**Critical path (ps)**
4	4	port_LLD_RoR_4v_4s	376	172	111
		port_LLD_Matrix_4v_4s	382	173	97
		port_LLD_HDRA_4v_4s	377	183	109
		port_ViChaR_RoR_4s	391	202	135
		port_ViChaR_Matrix_4s	397	206	116
		port_ViChaR_HDRA_4s	392	220	130
		port_EDVC_4v_4s	290	64	70
		port_RDQ_4v_4s	301	65	53
	8	port_LLD_RoR_4v_8s	632	252	123
		port_LLD_Matrix_4v_8s	638	254	112
		port_LLD_HDRA_4v_8s	633	264	121
		port_EDVC_4v_8s	596	96	88
		port_RDQ_4v_8s	594	98	81
	16	port_LLD_RoR_4v_16s	1197	435	147
		port_LLD_Matrix_4v_16s	1202	435	133
		port_LLD_HDRA_4v_16s	1198	453	140
		port_EDVC_4v_16s	1281	175	132
		port_RDQ_4v_16s	1181	171	97
	32	port_LLD_RoR_4v_32s	2409	851	195
		port_LLD_Matrix_4v_32s	2415	847	195
		port_LLD_HDRA_4v_32s	2410	863	195
		port_EDVC_4v_32s	2714	359	168
		port_RDQ_4v_32s	2385	332	114
8	8	port_LLD_RoR_8v_8s	881	392	172
		port_LLD_Matrix_8v_8s	937	437	142
		port_LLD_HDRA_8v_8s	887	421	174
		port_ViChaR_RoR_8s	1194	741	195
		port_ViChaR_Matrix_8s	1237	777	162
		port_ViChaR_HDRA_8s	1198	771	190
		port_EDVC_8v_8s	623	102	89
		port_RDQ_8v_8s	644	105	84
	16	port_LLD_RoR_8v_16s	1424	555	179
		port_LLD_Matrix_8v_16s	1469	595	149
		port_LLD_HDRA_8v_16s	1430	586	173
		port_ViChaR_RoR_16s[b]	4578	3016	302
		port_ViChaR_Matrix_16s[b]	4814	3217	232
		port_ViChaR_HDRA_16s[b]	4589	3054	265
		port_EDVC_8v_16s	1333	185	132
		port_RDQ_8v_16s	1271	183	97

(*Continued*)

TABLE 9.2 (CONTINUED)

Input-port hardware characteristics

			ASIC Design 15 nm NanGate Library		
VCs	**Slots**	**Input-Port Model**	**Area (μm²)**	**Power[a] (uW)**	**Critical path (ps)**
	32	port_LLD_RoR_8v_32s	2667	983	202
		port_LLD_Matrix_8v_32s	2700	1021	202
		port_LLD_HDRA_8v_32s	2673	1024	202
		port_ViChaR_RoR_32s[b]	19004	13258	454
		port_ViChaR_Matrix_32s[b]	19888	14062	327
		port_ViChaR_HDRA_32s[b]	19024	13291	343
		port_EDVC_8v_32s	2820	377	169
		port_RDQ_8v_32s	2515	340	114

[a] Frequency for power estimation = 1 GHz; the static power is around 10% of total power.
[b] The number of VCs of ViChaR input-port is equal to the number of slots of input-port.

TABLE 9.3

Arbiter Hardware Characteristics

		ASIC Design 15 nm NanGate Library		
VCs	**Arbiter Model**	**Area (μm²)**	**Power (uW)[a]**	**Critical path (ps)**
4	Matrix_4v	773	436	36
	RoR_4v	701	377	58
	HDRA_4v	710	432	56
	IRR_4v	712	366	42
8	Matrix_8v	2092	877	36
	RoR_8v	2020	817	58
	HDRA_8v	2029	874	56
	IRR_8v	2031	806	42
4/8	crossbar	104	35	0

[a] Frequency for power estimation= 1 GHz; the static power is around 10% of the total power.

characteristics. For example, the critical path of arbiter *Matrix_4v* is almost half of the *port_LLD_Matrix_4v_4s* critical path. This is due to a couple of features of our IRR arbiter. First of all, the SA module is separable, which reduces the SA logic complexity. Secondly, the *input-arbiter*s of SA are accommodated in the input-port. Therefore, considering the critical path delays of input-ports, arbiters and crossbar switch modules, the critical path delays of input-ports determine the maximum operating frequency, f_{max}, of the routers as listed in Table 9.4.

The power and area characteristics of each router given in Table 9.4 are calculated by adding the characteristics of five input-ports, an arbiter, and a crossbar switch listed in Tables 9.2 and 9.3. Table 9.4 also lists the advantage

rate of our RDQ-IRR router as compared to the other routers. One can observe that on average the RDQ-IRR routers have 11% less chip area, 50% less power consumption, and 103% faster frequency as compared to LLD and ViChaR routers for 4-VC router implementations. For the 8-VC buffer, our RDQ-IRR routers have at least 36% less chip area, 69% less power consumption, and 129% faster frequency as compared to LLD and ViChaR routers.

9.4.2 Performance Evaluation of RDQ-IRR NoC

Latency and throughput are the main performance parameters of NoCs, which are measured for RDQ-IRR NoC evaluation. The NoCs are implemented in System Verilog and we employ ModelSim to obtain these performance parameters. We explore and compare various NoCs mentioned earlier such as RDQ-IRR, EDVC-IRR, LLD-RoR, LLD-Matrix, LLD-HDRA, ViChaR-RoR, ViChaR-Matrix and ViChaR-HDRA for 8×8 NoC mesh topologies and for Uniform-random, Tornado and Complement traffic patterns [6, 15]. The throughput is measured by the rate of packets received to the maximum number of packets being injected at a specific time. The packet communication is based on wormhole switching where the channel width is equal to the flit size (16 bits). A packet consists of 16 flits, and each input-port has a central eight-slot buffer. There are four VCs per input-port except for ViChaR, which has eight VCs (the number of VCs in ViChaR is equal to the number of input-port buffer slots). The throughput and latency are measured for flit injection rates per time unit. For example, flit injection rate 8 means that each node (source core) injects eight flits per time unit. The maximum injection rate is considered based on the capability of delivering the flits by NoC routers. We have already mentioned that the flit arrival/departure for RDQ and EDVC routers is one cycle as compared to two cycles for LLD- and ViChaR-based routers. Therefore, if a time unit is assumed to equal 16 clock cycles, the LLD- and ViChaR-based sources can inject a maximum of eight flits, and the injection of more than eight flits is impossible. However, the RDQ- and EDVC-based source cores can inject a maximum 16 flits per time unit, and the injection of more than 16 flits is not possible. For the sake of fair comparison, we consider a maximum of eight-flit injection rate for RDQ and EDVC in our simulation.

The average latency is measured by the average time delays associated with the departure and arrival of a specific number (e.g. 2048) of packets in the NoC. The average latency estimation is formulated as follows:

$$\text{Average latency} = (\text{TRT} - \text{TST}) / \text{NSF} \tag{9.2}$$

where:

TRT is the total received time of flits,
TST is the total ideal sent times of flits,
NSF is the total number of sent flits.

TNF_09_K338522_C009_doobook_indd.indd 243

TABLE 9.4

Router Characteristics and Advantage Rate

			ASIC Design 15 nm NanGate Library			RDQ-IRR Advantage Rate		
# of VCs	**# of Slots**	**NoC Router Model**	**Area (μm²)**	**Power [a] (uW)**	**Delay (ps)**	**Area (saving)**	**Power (saving)**	**Frequency (faster)**
4	4	LLD-RoR	2685	1272	111	14%	43%	109%
		LLD-Matrix	2787	1336	97	17%	46%	83%
		LLD-HDRA	2699	1382	109	14%	47%	106%
		ViChaR-RoR	2760	1422	135	16%	49%	155%
		ViChaR-Matrix	2862	1501	116	19%	52%	119%
		ViChaR-HDRA	2774	1567	130	16%	54%	145%
		EDVC-IRR	2266	721	70	−2%	−1%	32%
		RDQ-IRR	2321	726	53	NA	NA	NA
	32	LLD-RoR	12850	4667	195	1%	56%	71%
		LLD-Matrix	12952	4706	195	2%	56%	71%
		LLD-HDRA	12864	4782	195	1%	57%	71%
		EDVC-IRR	14386	2196	168	11%	6%	47%
		RDQ-IRR	12741	2061	114	NA	NA	NA
8	8	LLD-RoR	6529	2812	172	18%	51%	105%
		LLD-Matrix	6881	3097	142	22%	56%	69%
		LLD-HDRA	6568	3014	174	18%	55%	107%
		ViChaR-RoR	8094	4557	195	34%	70%	132%
		ViChaR-Matrix	8381	4797	162	36%	72%	93%
		ViChaR-HDRA	8123	4764	190	34%	71%	126%
		EDVC-IRR	5250	1351	89	−2%	−1%	6%
		RDQ-IRR	5355	1366	84	NA	NA	NA

(*Continued*)

27-06-2018 21:56:49

TNF_09_K338522_C009_doobook_indd.indd 244

TABLE 9.4 (CONTINUED)

Router Characteristics and Advantage Rate

			ASIC Design 15 nm NanGate Library			RDQ-IRR Advantage Rate		
# of VCs	**# of Slots**	**NoC Router Model**	**Area (μm²)**	**Power [a] (uW)**	**Delay (ps)**	**Area (saving)**	**Power (saving)**	**Frequency (faster)**
	32	LLD-RoR	15459	5767	202	5%	56%	77%
		LLD-Matrix	15696	6017	202	6%	58%	77%
		LLD-HDRA	15498	6029	202	5%	58%	77%
		ViChaR-RoR	97144	67142	454	85%	96%	298%
		ViChaR-Matrix	101636	71222	327	86%	96%	187%
		ViChaR-HDRA	97253	67364	343	85%	96%	201%
		EDVC-IRR	16235	2726	169	9%	7%	48%
		RDQ-IRR	14710	2541	114	NA	NA	NA

[a] Frequency for power estimation = 1 GHz; the static power is around 10% of total power.

27-06-2018 21:56:49

The maximum latency in our design cannot exceed a specific limit. This is due the fact that an injection rate of more than 8 in LLD and ViChaR is impossible. In other word, considering Equation 9.2, the TRT is not changed when the flit injection rate becomes more than 8 (in LLD and ViChaR designs), but the TST can become zero theoretically (i.e. all the flits are injected in time zero). The NSF is fixed and equal to the multiplication of three parameters: the number of injected packets (e.g. 2048), the number of sources (for 8×8 NoC, it is 64), and the flits/packet (i.e. 16). Therefore, the maximum average latency is fixed and equal to TRT/NSF.

LLD-RoR, LLD-Matrix and LLD-HDRA NoCs behave similar to each other in terms of functionality at the same frequency. It is due to the fact that RoR, Matrix and HDRA arbiters behave in a functionally similar way. For instance, LLD-Matrix with faster clock rate supersedes the LLD-RoR and LLD-HDRA NoCs in terms of performance. The same conclusion can be drawn for the ViChaR-Matrix NoC. Therefore, four fast NoCs (LLD-Matrix, ViChaR-Matrix, EDVC-IRR and RDQ-IRR) were selected for evaluation, and the comparison is presented in Table 9.4. The LLD-Matrix, ViChaR-Matrix, EDVC-IRR, and RDQ-IRR run at 565, 457, 718 and 1000 MHz clock respectively with 4-VC configuration. These clock rates correspond to the critical path delays listed in Table 9.4.

The performance metrics of each NoC depend on the functional behaviour of its data flow mechanism and the timing characteristics associated with the router. The critical path delays associated with the router of each NoC must be considered in the evaluation of NoC performance. Therefore, we tested these NoCs under different clock rates according to the critical path delays associated with their routers. In this experiment, the performance parameters of aforementioned NoCs are evaluated and the results are shown in Figures 9.19 and 9.20. The clock rate is in linear relation to the performance metrics. Consider n packets passing through the NoC system during t time at a clock rate of f. Then $p \times n$ packets will pass through the NoC system at $p \times f$ clock rate during t time. One can observe that the RDQ-IRR latency and throughput results presented in Figures 9.19 and 9.20 show better performance than those of others. At higher injection rates, more flits are injected, and the NoCs become populated, producing higher contention. The back-pressure mechanism associated with the blocked packets in RDQ helps to improve the performance of RDQ-IRR-based NoC as follows [15]. First, the probability of a monopoly of an input-port buffer by a growing VC is reduced. Secondly, the free packets receive more buffer-free space to pass through the NoC. Moreover, the RDQ-IRR NoC frequency employed for the results presented in this section is higher than those of the other approaches, and the flit arrival/departure time for RDQ-IRR and EDVC-IRR routers is one cycle as compared to two cycles for LLD-Matrix and ViChaR-Matrix routers. The EDVC-IRR NoC has lower performance as compared to the RDQ-IRR due to the fact that the RDQ mechanism is an improved version of EDVC [16]. The average RDQ-IRR latencies are 78%, 83% and 81% less than those of

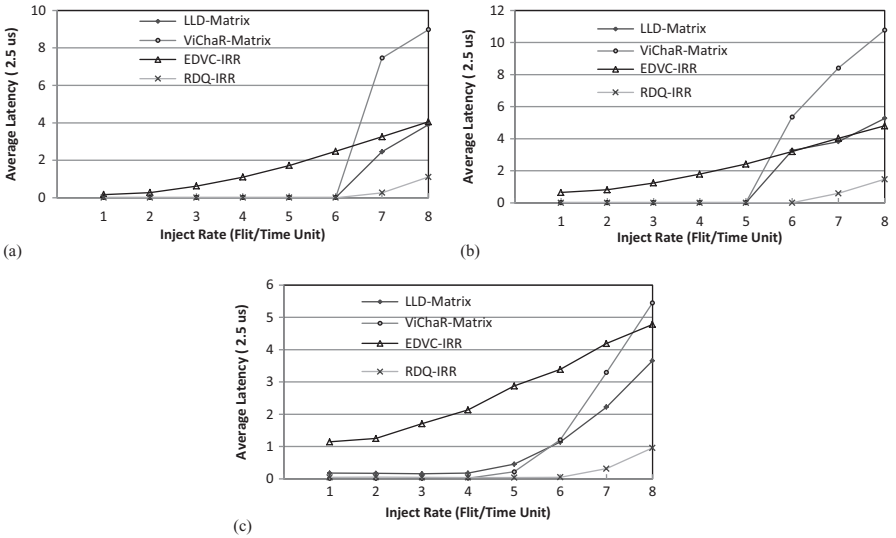

FIGURE 9.19
Latency for Tornado and Complement and Uniform Random traffic in 8×8 mesh topology. (a) Tornado traffic (b) Complement traffic, (c) Uniform-random traffic.

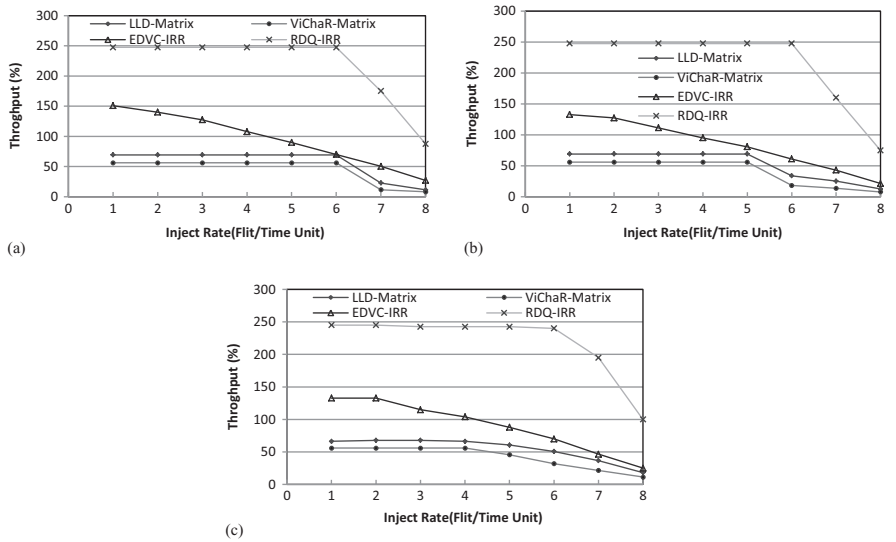

FIGURE 9.20
Throughput for Tornado, Complement and Uniform Random traffic for 8×8 mesh NoC. (a) Tornado traffic (b) Complement traffic (c) Uniform-random traffic.

LLD-Matrix for Tornado, Complement and Uniform-random traffic patterns respectively. Average throughputs of RDQ-IRR are 74%, 75% and 75% higher than those of LLD-Matrix for Tornado, Complement and Uniform-random traffic patterns respectively.

We have also investigated the performance impact of our RDQ-IRR approach for three application-specific NoCs: an MPEG-4 decoder [32] (MPEG-4), an Audio/Video Benchmark application [32] (AV), and an enhanced Video Object Plane Decoder [33] (DVOPD) with the capability to decode two streams in parallel. The MPEG-4, AV, DVOPD applications are mapped to 3×4, 4×4, and 4×7 2D-mesh topology NoCs respectively. The core graphs of MPEG-4, AV and DVOPD applications are given in Figures 21 and 22. The communication of packets is based on wormhole routing and follows a deterministic XY routing algorithm. The arrows show the direction of packet from source cores to sinks. They also specify the number of VCs needed for each channel. For example, three arrows toward the north input channel of router #5 (Figure 9.21 left) means three packets may pass through the channel at the same time. Three packets require three VCs to service them without any blockage. The other setup conditions of NoCs are the same as the previous experiment, i.e. 8-slot/buffer, 4-VC/input-port, 16-bit/flit, and 16-flit/packet. The latency is measured through a time slice in which all the destinations receive their assigned packets, e.g. when MPEG-4 and AV destinations receive 55472 and 380128 packets respectively.

The contentions in these three NoC applications are low due to two conditions. First, we have set up 4-VC per input-port that is more than the maximum requested VCs in these applications. For instance, the maximum requested VCs in MPEG-4, AV and DVOPD are 3, 2 and 2 respectively, as one can see from Figures 9.21 and 9.22. Secondly, the packet flows in many routes are not crowded, e.g. the west input channel of router #11 of MPEG-4 carries the highest packet flow. Under the above conditions, NoC frequency

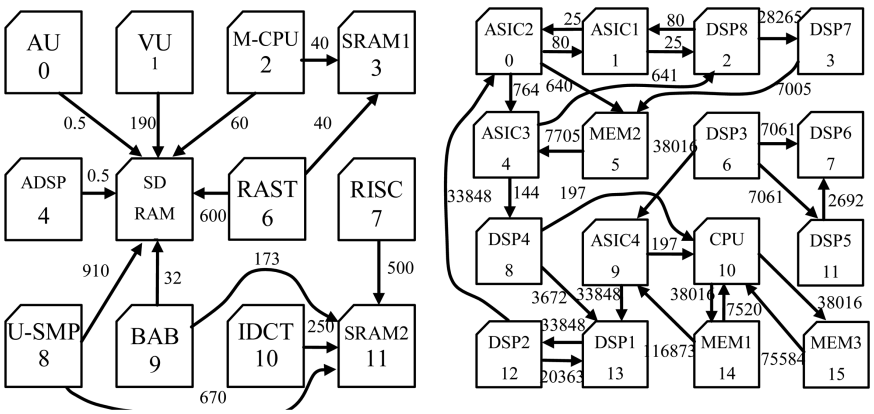

FIGURE 9.21
Mapping of MPEG-4 (left) and AV (right) core graphs to a 3×4 and 4×4 mesh topology NoC.

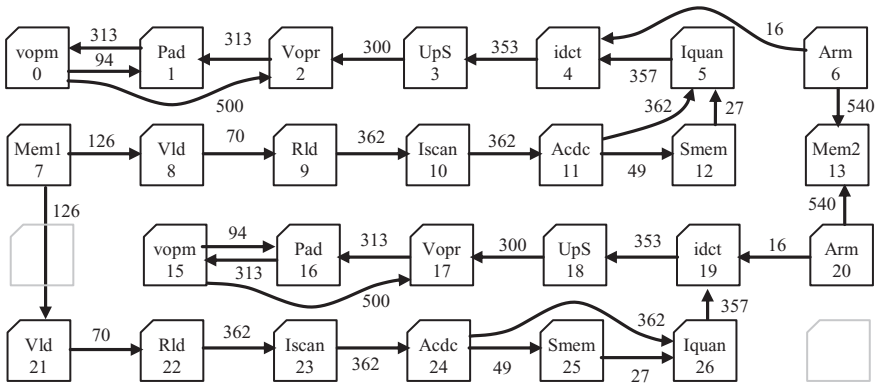

FIGURE 9.22
DVOPD core graph map in a 4×7 mesh NoC.

and flit arrival/departure have great impact on the performance of NoCs. As already mentioned, the LLD-Matrix, ViChaR-Matrix, EDVC-IRR, and RDQ-IRR run at 514, 451, 820 and 1000 MHz clock respectively, and the EDVC-IRR and RDQ-IRR routers take one cycle, as compared to two cycles for the LLD-Matrix and ViChaR-Matrix routers. The simulation results presented in Figure 9.23 support our above analysis. The results show that the average latency of RDQ-IRR is 72%, 92% and 78% less than the average latency of LLD-Matrix for MPEG-4, AV and DVOPD applications.

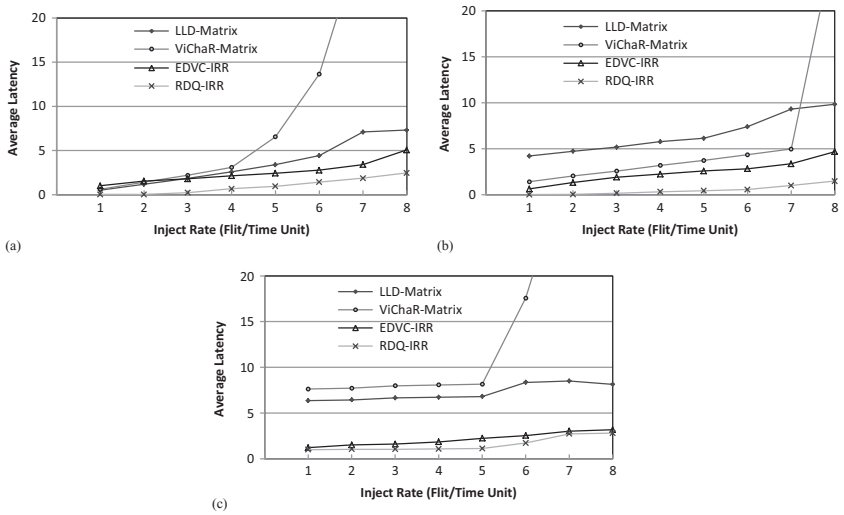

FIGURE 9.23
Latency for MPEG-4, AV and DVOPD Traffic for 3×4, 4×4 and 5×5 mesh NoCs respectively: (a) MPEG-4 traffic; (b) AV Benchmark traffic; (c) DVOPD traffic.

9.5 Conclusions

We have presented a novel NoC router micro-architecture that employs an efficient buffering (RDQ) organization along with a faster arbitration (IRR) technique. The RDQ input-port has been presented as a solution to the EDVC [15] drawbacks. The IRR arbiter employs a least-recently-served priority scheme and achieves strong fairness arbitration. The addition of these two components (i.e. RDQ input-port and IRR arbiter) has been investigated for their impact on the efficiency of NoC systems. The experimental results related to required NoC chip area and power consumption have also been presented. These verify that our RDQ-IRR NoC is low-power and economical. The RDQ-IRR routers have at least 14% less chip area, 43% less power consumption, and 83% faster frequency as compared to the LLD and ViChaR routers for 4-VC and 4-slot implementations. Moreover, the RDQ-IRR NoCs operate at on average 74%, 75% and 75% higher throughputs and 78%, 83% and 81% lower latencies than those of LLD for Tornado, Complement and Uniform-random traffic patterns respectively. The effectiveness of our RDQ-IRR NoCs has also been investigated for some applications, such as MPEG-4, AV benchmark and DVOPD. The average RDQ-IRR latency decreases by 72%, 92% and 78% as compared to that of LLD-Matrix NoCs for MPEG-4, AV and DVOPD applications.

Acknowledgments

The author acknowledges and thanks his graduate students for generating the experimental results presented in this chapter. The financial and equipment support provided by the Electrical and Computer Engineering Department & FEAS at Ryerson University, NSERC and CMC Canada.

References

1. W.J. Dally and B. Towles. (2004). Arbitration. In: *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, pp. 349–362.
2. W.J. Dally and B. Towles. (2004). Buffered flow control. In: *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, pp. 233–256.
3. C.A. Nicopoulos, P. Dongkook, K. Jongman, N. Vijaykrishnan, M.S. Yousif, and C.R. Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers, in *Proc. 39th IEEE/ACM Int. Symp. on Microarchitecture*, pp. 333–346, Orlando, FL, Dec. 2006.

4. M. Evripidou, C. Nicopoulos, V. Soteriou, and J. Kim, Virtualizing virtual channels for increased network-on-chip robustness and upgradeability, in *Proc. IEEE Symp. on VLSI*, pp. 21–26, Amherst, MA, Aug. 2012.
5. Y. Choi and T.M. Pinkston, Evaluation of queue designs for true fully adaptive routers, *Journal of Parallel and Distributed Computing*, vol. 64, no. 5, pp. 606–616, May 2004.
6. Y. Xu, B. Zhao, Y. Zhang, and J. Yang, Simple virtual channel allocation for high throughput and high frequency on-chip routers, in *Proc. Int. Symp. on High Performance Computer Architecture*, pp. 1–11, Bangalore, India, Jan. 2009.
7. M. Lai, Z. Wang, L. Gao, H. Lu, and K. Dai, A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers, in *Proc. 45th annual Design Automation Conf.*, pp. 630–633, Anaheim CA, Jun. 2008.
8. Y. Tamir and G.L. Frazier, Dynamically-allocated multi-queue buffers for VLSI communication switches, *IEEE Trans. on Computers*, vol. 41, no. 6, pp. 725–737, Jun. 1992.
9. M. Lai, L. Gao, W. Shi, and Z. Wang, Escaping from blocking: A dynamic virtual channel for pipelined, in *Proc. Int. Conf. Complex, Intelligent and Software Intensive Syst.*, pp. 795–800, Barcelona, Spain, Mar. 2008.
10. D.U. Becker and W.J. Dally, Allocator implementations for network-on-chip routers, in *Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis*, pp. 1–12, Portland, OR, 2009.
11. G.L. Frazier and Y. Tamir, The design and implementation of a multiqueue buffer for VLSI communication switches, in *Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pp. 466–471, Cambridge, MA, 1989.
12. J. Park, B.W. O'Krafka, S. Vassiliadis, and J. Delgado-Frias, Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers, in *Proc. Supercomputing*, pp. 713–722, Washington, DC, Nov. 1994.
13. P. Forstner. (1999). FIFO architecture, functions, and applications. www.ti.com/lit/an/scaa042a/scaa042a.pdf, Last accessed Dec. 2017.
14. J. G. Delgado-Frias and R. Diaz, A VLSI self-compacting buffer for DAMQ communication switches, in *IEEE Proc. of the 8th Great Lakes Symposium on VLSI*, Lafayette, LA, 1998.
15. M. Oveis-Gharan and G.N. Khan, Efficient dynamic virtual channel organization and architecture for NoC systems, *IEEE Trans. on VLSI Systems*, vol. 24, no. 2, pp. 465–478, Feb. 2016.
16. M. Oveis-Gharan and G.N. Khan, Dynamic VC organization for efficient NoC communication, in *Proc. IEEE 9th Int. Symp. on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 151–158, Turin, Italy, Sep. 2015.
17. G. Jiang, Z. Li, F. Wang, and S. Wei, A low-latency and low-power hybrid scheme for on-chip networks, *IEEE Trans. on VLSI Systems*, vol. 23, no. 4, pp. 664–677, Apr. 2015.
18. Y. Ben-Itzhak, I. Cidon, A. Kolodny, M. Shabun, and N. Shmuel, Heterogeneous NoC router architecture, *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2479–2492, 2015.
19. R. Ramanujam, V. Soteriou, B. Lin, and L. Peh, Design of a high-throughput distributed shared-buffer NoC router, in *Proc. 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 69–78, Grenoble, France, May 2010.

20. D. Zoni, J. Flich, and W. Fornaciari, CUTBUF: Buffer management and router design for traffic mixing in VNET-based NoCs, *IEEE Trans. on Parallel and Distributed Sys.*, vol. 27, no. 6, pp. 1603–1616, Jun. 2016.
21. M. Oveis Gharan and G. N. Khan, Packet-based adaptive virtual channel configuration for NoC systems, *International Workshop on the Design and Performance of Network on Chip, Procedia Computer Science*, vol. 34, pp. 552–558, 2014.
22. T. Yung-Chou and H. Yarsun, Design and evaluation of dynamically-allocated multi-queue buffers with multiple packets for NoC routers, *Sixth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pp. 1–6, Beijing, China, Jul. 2014.
23. A.T. Tran and B.M. Baas, Achieving high-performance on-chip networks with shared-buffer routers, *IEEE Trans. on VLSI Systems*, vol. 22, no. 6, pp. 1391–1403, Jun. 2014.
24. K.A. Helal, S. Attia, T. Ismail, and H. Mostafa, Priority-select arbiter: An efficient round-robin arbiter, in *IEEE 13th Int. Conf. on New Circuits and Systems (NEWCAS)*, pp. 1–4, Grenoble, France, Jun. 2015.
25. Z. Fu and X. Ling, The design and implementation of arbiters for network-on-chips, in *2nd Int. Conf. on Industrial and Information Systems*, pp. 292–295, Dalian, China, 2010.
26. M. Oveis-Gharan and G.N. Khan, Index-based round-robin arbiter for NoC routers, in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 62–67, Montpellier, France, July 2015.
27. Y. Lee, J. M. Jou, and Y. Chen, A high-speed and decentralized arbiter design for NoC, in *Proc. IEEE Int. Conf. on Computer Systems and Applications*, pp. 350–353, Rabat, Morocco, 2009.
28. F. Guderian, E. Fischer, M. Winter, and G. Fettweis, Fair rate packet arbitration in network-on-chip, in *Proc. IEEE Int. System-on-Chip Conference (SOCC)*, Taipei, Taiwan, pp. 278–283, 2011.
29. L. Shaoteng, A. Jantsch, and L. Zhonghai, A fair and maximal allocator for single-cycle on-chip homogeneous resource allocation, *IEEE Trans. on VLSI Systems*, vol. 22, no. 10, pp. 2229–2233, Oct. 2013.
30. M. Fattah, A. Manian, A. Rahimi, and S. Mohammadi, A high throughput low power FIFO used for GALS NoC buffers, in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 333–338, Lixouri, Greece, July 2010.
31. NANGATE. 2014. Nangate Releases 15nm Open Source Digital Cell Library. [ONLINE] Available at: www.nangate.com. Accessed December 2017.
32. M. Oveis-Gharan, and G.N. Khan, Statically adaptive multi FIFO buffer architecture for network on chip, *Microprocessors and Microsystems*, Vol. 39, No. 1, pp. 11–26, Feb. 2015.
33. A. Pullini, F. Angiolini, P. Meloni, D. Atienza, S. Murali, L. Raffo, G. D. Micheli, and L. Benini, NoC design and implementation in 65nm technology, in *Proc. 1st Int. Symp. on Networks-on-Chip*, Princeton, NJ, pp. 273–282, 2007.

TNF_09_K338522_C009_docbook_indd.indd 252 27-06-2018 21:56:51