Hindawi Publishing Corporation
VLSI Design
Volume 2009, Article ID 415646, 10 pages
doi:10.1155/2009/415646

Research Article

Low-Cost Allocator Implementations for Networks-on-Chip Routers

Min Zhang and Chiu-Sing Choy

Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

Correspondence should be addressed to Min Zhang, mzhang@ee.cuhk.edu.hk

Received 24 June 2009; Revised 18 October 2009; Accepted 19 December 2009

Recommended by Maurizio Palesi

Cost-effective Networks-on-Chip (NoCs) routers are important for future SoCs and embedded devices. Implementation results show that the generic virtual channel allocator (VA) and the generic switch allocator (SA) of a router consume large amount of area and power. In this paper, after a careful study of the working principle of a VA and the utilization statistics of its arbiters, opportunities to simplify the generic VA are identified. Then, the deadlock problem for a combined switch and virtual channel allocator (SVA) is studied. Next, the impact of the VA simplification on the router critical paths is analyzed. Finally, the generic architecture and two low-cost architectures proposed (the look-ahead, and the SVA) are evaluated with a cycle-accurate network simulator and detailed VLSI implementations. Results show that both the look-ahead and the SVA significantly reduce area and power compared to the generic architecture. Furthermore, cost savings are achieved without performance penalty.

1. Introduction

In order to utilize the exploding number of transistors in a single piece of silicon, integrating multiprocessing elements (PE) in Systems-on-Chip (SoCs) becomes inevitable. According to [1], the number of PEs in SoCs will increase to about 80 in 2010, 270 in 2015, and 880 in 2020. Networks-on-Chip (NoCs) has been proposed to replace the shared-bus and the point-to-point links to provide on-chip interconnection for future SoCs.

As compared to traditional off-chip networks, the resource limitations—area and power limitations—are major constraints in NoCs design, especially NoCs for SoCs and embedded devices [2, 3]. Thus, whereas previous work focused on how to improve NoCs network performances [4, 5], we concentrate on how to reduce NoCs design costs.

Generic architectures of a virtual channel (VC) allocator (VA) and a switch allocator (SA) were presented in [6]. However, experiments show that they are large in area and high in power. We presented two low-cost virtual channel (VC) allocators (the look-ahead and the unfair) in [7]. In the look-ahead VC allocator, a large number of arbiters in the second stage of a generic VA are removed and arbiters in the first stage of a generic VA are replaced by comparators to reduce design costs. In the unfair VA, costs are decreased by substituting simple unfair arbiters for some matrix arbiters. Following on this previous work, the contributions of this study include the following aspects.

(i) Opportunities and methods to simplify the VA are presented in a more detailed way.

(ii) A combined VA and SA allocator (SVA) is proposed and the deadlock problem for the SVA is discussed.

(iii) The impact of the VA simplification methods on the router critical paths is analyzed.

(iv) More comprehensive evaluations are performed on network performance, delay, area, and power for a wide range of design parameters and traffic patterns.

The paper is organized as follows. Sections 2 and 3 review the related work and background, respectively. In Section 4, we present how VA arbiters are simplified. In Section 5, we discuss deadlock problem of the SVA. Section 6 analyzes VA simplification effects on router critical paths. Section 7 describes experiment platform and results and Section 8 concludes the paper.

2. Related Work

Due to low-latency and low-buffer requirements, VC wormhole (WH) routers suggested by Dally [8] are widely used in NoCs studies. Based on the VC flow control, much work has been done to improve network performances. The authors in [5] presented a single-cycle router for low-latency designs. Lu et al. in [9] proposed a layered switching technique where the salient features of both the wormhole and the virtual cut-through switching techniques were exploited. A dynamic VC regulator was described in [10] to dynamically adjust buffer architectures according to network loads.

Some researchers shed light on reducing design costs for VC routers. Several power-efficient components like a segmented crossbar, a cut-through crossbar, and a write-through input buffer were proposed in [11]. Buffers for input ports were customized to reduce large buffer costs in [12]. Cost-effective flit admission and ejection blocks were presented in [13]. However, these studies focused on components in the data path (crossbar, buffer, etc.) of a router. Few researchers address low-cost components design in the control path. In this paper, we propose low-cost allocator architectures, which are orthogonal to the previous low-cost datapath techniques and combining them together can generate more cost savings.

3. Background

3.1. Router Microarchitecture. Figure 1 illustrates the microarchitecture of a generic VC WH router. The router has p_i input and p_o output channels/ports, supporting V VCs per port. The FIFOs in each input port buffer arriving flits (buffer write (BW) stage). The routing computation directs the head flit of an incoming packet to the appropriate output physical channel (PC) (routing computation (RC) stage). The VC allocator arbitrates among those *input VCs* (VCs of input ports) requesting the same *output VC* (VCs of output PCs (in fact, VCs of an output PC are VCs of the connected input port at the downstream router)) and assigns free output VCs to successful input VCs (VA stage). The switch allocator distributes output PCs and the crossbar to input VCs (SA stage). After virtual channel and switch allocations, the crossbar will pass flits to appropriate output PCs (switch traversal (ST) stage), and flits will traverse output PCs to the next router (link traversal (LT) stage). Figure 2 shows the pipeline of a router. Each head flit needs to pass five stages whereas those body/tail flits need to pass four stages. The RC stage can be removed because look-ahead routing [5] is adopted.

3.2. Generic VA and SA Architectures. Figures 3 and 4 show design complexities of a generic VA and a generic SA, respectively. When a deterministic routing algorithm is used, the case that the RC returns all free VCs of a single output PC is the most general. In this case, the VA performs arbitration in two stages (Figure 3(a)). In the first stage, each input VC selects one free VC. Since there are at most V free VCs in an output PC, a $V : 1$ arbiter is needed for every input VC. In the second stage, each output VC is asked to grant access to one of all requests. The number of requests is p_iV in the worst case; so each output VC needs a $p_iV : 1$ arbiter. A $p_iV : 1$ arbiter is pretty large and can be simplified by organizing it as a tree of smaller arbiters [5], as shown in Figure 3 (b). In the figure, the $V : 1$ arbiters arbitrate between requests from the same input ports and the $p_i : 1$ arbiter determines the winning input port.

The generic SA is divided into two stages as well. The first stage takes into account the sharing of a single crossbar input port by V input VCs and has a $V : 1$ arbiter for each crossbar input port. The second stage then determines the links between crossbar input ports and crossbar output ports. This needs a $p_i : 1$ arbiter for each crossbar output port.

Table 1: Cost proportions in a router.

Router parameters	Area	Power
$p_i = 3, V = 4$	26.91%	28.06%
$p_i = 4, V = 4$	34.64%	30.79%
$p_i = 5, V = 2$	21.62%	20.81%
$p_i = 5, V = 4$	41.36%	32.65%
$p_i = 5, V = 6$	57.27%	37.28%

3.3. Motivations. Table 1 shows the proportions of area and power demanded for the allocation logic, including the generic VA, the generic SA, and associated logics for different numbers of ports and VCs in a router. As stated in Table 7, the router uses virtual channel flow control, XY routing, credit-based buffer management, and flit size of 36 flits. Area was taken from Synopsys DC synthesis report under worse case conditions. Power was obtained from Synopsys PrimeTime PX with UMC 130 nm library files, postsynthesis netlist, wire load model, and postsynthesis switching activities as inputs. The switching activities of router_{30} (a 3-port router; we define the left-bottom router, as router_{00} for the 4×4 mesh throughout the paper), router_{31} (a 4-port router), and router_{21} (a 5-port router) were derived from simulations of an entire NoCs assuming uniform traffic pattern running at saturation points. (Unless otherwise stated, results in the paper are obtained when router radix is 5, the number of VCs is 4, packet length is 4, and network runs at the saturation point of uniform traffic at 250 MHz). It can be seen that the allocation logic consumes significant amounts of area and power for all cases. Furthermore, the allocation logic cost proportionally increases with the number of ports or VCs. Therefore, it is important to reduce this cost.

Obviously, an VA is much more costly than a SA because it includes a large number of big ($p_iV : 1$) arbiters. To our credit, it is possible to totally remove all arbiters in a VA and to make a VA and a SA share the same arbiters, the one shown in Figure 4. The sharing is identified after a careful study of the working principle of a VA and utilization statistics of VA arbiters. This is explained in detail in the next section.

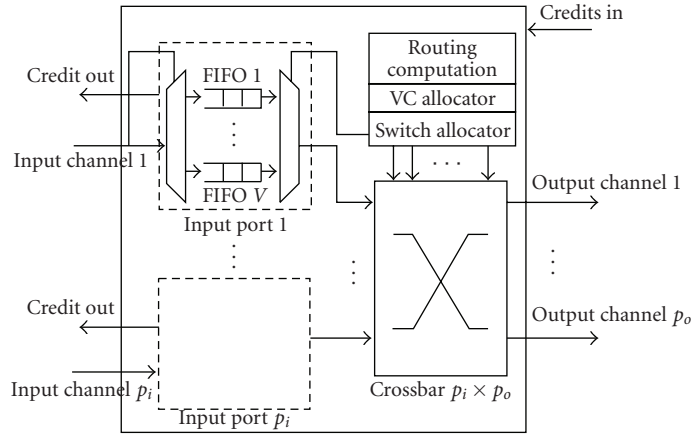


Figure 1: A VC WH router architecture.

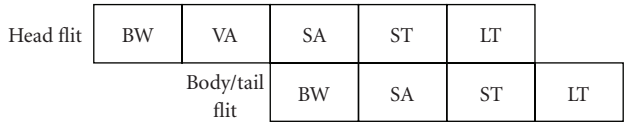


Figure 2: Pipeline diagram of a router.

4. VA Simplification

4.1. Representations. Many parameters are used in this paper and are defined in Table 2 for reference.

The second stage of a VA allocates p_oV output VCs to p_iV input VCs. Design complexity of the second stage can be represented as a $p_iV \times p_oV$ request matrix:

$$R = \begin{bmatrix} r_{11}^{11} & \cdots & r_{11}^{1V} & \cdots & r_{11}^{p_o1} & \cdots & r_{11}^{p_oV} \\ \vdots & & & & & & \vdots \\ r_{1V}^{11} & \cdots & r_{1V}^{1V} & \cdots & r_{1V}^{p_o1} & \cdots & r_{1V}^{p_oV} \\ \vdots & & & & & & \vdots \\ r_{p_i1}^{11} & \cdots & r_{p_i1}^{1V} & \cdots & r_{p_i1}^{p_o1} & \cdots & r_{p_i1}^{p_oV} \\ \vdots & & & & & & \vdots \\ r_{p_iV}^{11} & \cdots & r_{p_iV}^{1V} & \cdots & r_{p_iV}^{p_o1} & \cdots & r_{p_iV}^{p_oV} \end{bmatrix}, \quad (1)$$

where each row represents requests from an input VC while each column represents requests to an output VC. Since an input VC is not allowed to request more than one output VC, there is at most one valid request in each row. As a result, a row does not require an arbiter. But, there are at most p_iV requests to an output VC; so a column requires a $p_iV : 1$ arbiter.

4.2. Reducing Number of VA Arbiters. It is too generous to assign an arbiter for each output VC based on two observations. Firstly, the frequency that an output VC will be requested is low since a VA is performed only for head flits. Secondly, when an output VC is not being requested, its arbiter can be used by another output VC through logic sharing. In other words, the number of arbiters indeed required is determined by the number of output VCs that are simultaneously on request instead of the total number of output VCs.

We ran simulations for a 4×4 mesh NoCs under uniform, hotspot, and transpose traffic patterns and calculated pb^k by counting the number of cycles when r^k is 2 or above and then dividing by the number of simulation cycles. The

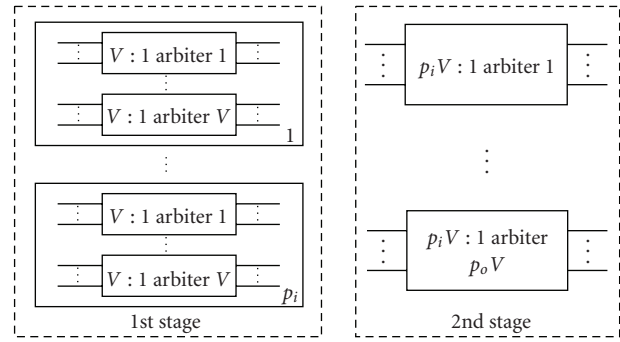


Figure 3: (a) A generic VC allocator. (b) Tree architecture of a $p_iV : 1$ arbiter.

Table 2: Parameter list.

Parameter	Description
R	The request matrix
r_{mn}^{kl}	The request from the nth VC at the mth input port to the lth VC at the kth output port. The value is 1 if the request is valid. Otherwise, the value is 0.
r_{mn}^{k}	The request from the nth input VC at the mth input port to any VC at the kth output port. The value is 1 if the request is valid. Otherwise, the value is 0.
r^{kl}	Whether there are valid requests to the lth VC of the kth output port. It is 0 when $(\sum_{m=1}^{p_i}\sum_{n=1}^{V} r_{mn}^{kl}) = 0$. Otherwise, it is 1.
r^{k}	The number of simultaneously requested VCs at the kth output port $(\sum_{l=1}^{V} r^{kl})$.
r_{m}^{k}	Whether there are valid requests from any VCs at the mth input port to any VC at the kth output port. It is 0 when $\sum_{n=1}^{V} r_{mn}^{k} = 0$. Otherwise, it is 1.
r_m	The number of output ports where a VC is requested by any VCs in the mth input port $(\sum_{k=1}^{p_o} r_m^k)$.
$pb1^k$	The probability that one VC at the kth output port is requested.
pb^k	The probability that multi-VCs at the kth output port are concurrently requested.
$pb1_m$	The probability that VCs in the mth input port request VCs at one output ports.
pb_m	The probability that VCs in the mth input port request VCs at multi output ports.

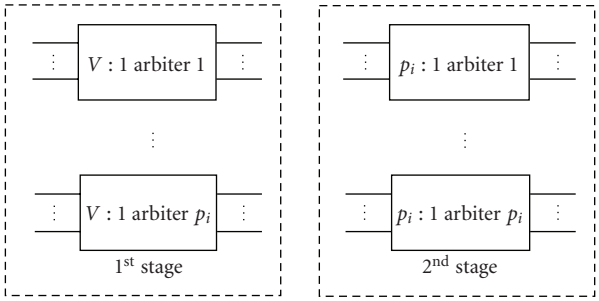


Figure 4: A generic switch allocator.

Table 3: Results of pb^k.

Traffic pattern	Output PC				
	West	North	East	South	Local
Uniform	0.21%	0.23%	0.32%	0.33%	0.16%
Hotspot	0.23%	0.10%	0.30%	0.48%	0.30%
Transpose	0	0	0	0.01%	0

results show that pb^k remains small for all routers for all injection rates. Hence, for each traffic pattern, only the pb^k for output PCs of the router$_{21}$ at saturation point is shown (Table 3)(The saturation points are at 0.652, 0.603, and 0.248 flits/(node*cycle) for uniform, hotspot, and transpose traffic patterns, resp.). It can be seen that all pb^k are smaller than 0.50% for all tested traffic patterns. In other words, in each output PC, at most one VC is requested in more than 99.50% of all cases. The key reason for small pb^k results is that pb^k represents the probability that two or more VCs in an output port are simultaneously requested. To validate this claim, we also calculated $pb1^k$ for the three traffic patterns (Table 4). We can see that results of $pb1^k$ are much larger than results of pb^k.

Table 4: Results of $pb1^k$.

Traffic pattern	Output PC				
	West	North	East	South	Local
Uniform	13.78%	12.43%	16.28%	17.53%	15.43%
Hotspot	12.68%	11.26%	16.86%	13.88%	13.41%
Transpose	0	7.90%	0	16.48%	0

As a result, it is totally unnecessary to make available all free VCs of a single output PC in the RC. *One can easily change the RC function so it makes only one free VC available for an output PC. Then, the first stage of a VA can be removed and only one p_iV : 1 arbiter is required for all VCs of an output PC.* The request matrix, R, is simplified to a $p_iV \times p_o$ matrix:

$$R = \begin{bmatrix} r_{11}^{1} & \cdots & r_{11}^{p_o} \\ \vdots & & \vdots \\ r_{1V}^{1} & & r_{1V}^{p_o} \\ \vdots & & \vdots \\ r_{p_i1}^{1} & & r_{p_i1}^{p_o} \\ \vdots & & \vdots \\ r_{p_iV}^{1} & \cdots & r_{p_iV}^{p_o} \end{bmatrix}. \quad (2)$$

4.3. Sharing of V : 1 Arbiters. The VA after simplification now requires p_op_iV : 1 arbiters. This can be similarly organized as a tree architecture which has a set of p_oV : 1 arbiters for every input port and one p_i : 1 arbiter for each output port. This is also too generous because there are totally V VCs in an input port and many of them do not have VA requests most of the time. From simulations as described previously, we calculated pb_m by counting the number of cycles when r_m is 2 or above and then dividing by the number of simulation cycles. Table 5 shows pb_m for input PCs of the

Table 5: Results of pb_m.

Traffic pattern	Input PC				
	West	North	East	South	Local
Uniform	0.20%	0.23%	0.75%	0.33%	0.15%
Hotspot	0.20%	0.15%	0.13%	0.15%	0.28%
Transpose	0	0	0	0	0

Table 6: Results of $pb1_m$.

Traffic pattern	Input PC				
	West	North	East	South	Local
Uniform	13.18%	15.91%	17.76%	16.91%	17.06%
Hotspot	11.01%	12.36%	15.31%	15.53%	20.91%
Transpose	7.99%	0	16.75%	0	0

router_{21} at saturation points. The largest pb_m is 0.75% for uniform traffic pattern and 0.28% for hotspot traffic pattern. All the pb_m are zeros for transpose traffic pattern because no packets enter the north, south, and local input PCs and packets entering the west (east) input PC only go to the north (south) output PC. Similar to pb^k, the key reason for small pb_m is that pb_m represents the probability that VCs in an input port request VCs at two or more output ports simultaneously. We can see that results of $pb1_m$ (Table 6) are significantly larger than results of pb_m. *Therefore, making all VCs of an input port to share one* V : 1 *arbiter decreases VA efficiency by only a small amount. As a result, it is possible with negligible performance cost to reduce the number of* V : 1 *arbiters at a single input port from* p_o *to 1.*

4.4. Combining VA and SA Arbiters. After the above two simplifications, a VA will consist of one V : 1 arbiter at each input port and one p_i : 1 arbiter at each output PC, which is obviously the same as the SA shown in Figure 4. Moreover, VA arbiters have the same functions as the corresponding SA arbiters: a V : 1 arbiter handles requests from VCs at an input port while a p_i : 1 arbiter deals with requests from various input ports to an output PC. The only difference is the type of requests (VA or SA requests). *Thus, a VA and an SA can share their arbiters if VA and SA requests are processed concurrently. (The concurrent processing means to process VA requests from some input VCs and to process SA requests from other input VCs in the same cycle. This is because an input VC can only be at one of three states, namely, making no request, making a VA request, and making an SA request.) This leads to a further 50% reduction of arbiters.* More importantly, as shown in Figure 5, combining a VA and a SA removes the VA pipeline stage for head flits and thus reduces packet latency. But, processing VA and SA requests concurrently may lead to deadlock and will be discussed in the next section.

5. Deadlock

5.1. Free Output VC Check. An input VC is allocated an output VC successfully when two requirements are met. One is that the VA request of the input VC wins in the SVA, and the other is that there is a free output VC in the destined output PC. Free output VC check may be done in a speculative (Figure 6(a)) or nonspeculative (Figure 6(b)) way. The speculative architecture removes free output VC check logic from the critical path and seems to be more attractive. This is why it is often selected for a generic VA. However, if an SVA is used, we found that the speculative architecture is not deadlock free and we have to use the nonspeculative architecture.

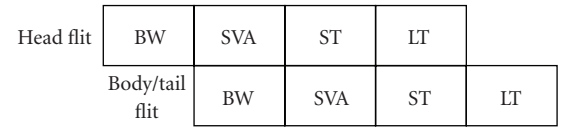


Figure 5: Pipeline diagram after combing the VA and the SA.

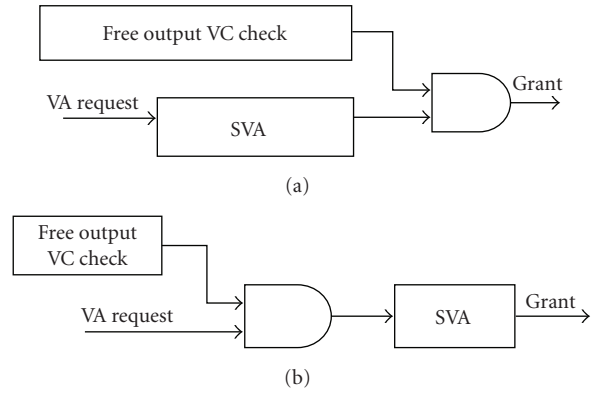


Figure 6: (a) A speculative architecture. (b) A nonspeculative architecture.

5.2. Deadlock. If the head flit of a packet succeeds to be allocated an output VC in the VA stage, the output VC is held for the whole packet (or the input VC that buffers the packet) until the tail flit of this packet is successfully allocated crossbar bandwidth in the SA stage and releases the output VC. In other words, a logic connection between an input VC and an output VC is established by a VA operation and is released by a SA operation. In general, the round-robin algorithm is applied in both VA and SA arbiters to achieve fairness. A priority table is used to accomplish the round-robin algorithm by setting the priority of the request that has just been served the lowest.

After a VA and an SA are combined, there are two groups of requests: the VA group and the SA group. Also, there are two groups of resources: the output VC group and the priority group. Requests and resources are related by hold and wait-for relations. As shown in Figure 7(a), the VA group holds the priority group and waits for the output VC group. Similarly, the SA group holds the output VC group and waits for the priority group. If a request group holds a resource group, then that resource group is waiting on the request group to release it. Thus, each hold relation induces a wait-for relation in the opposite direction [14]. Redrawing the

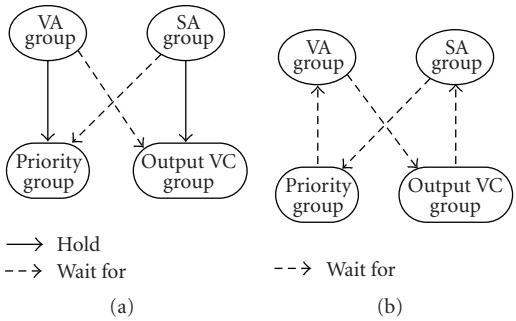


Figure 7: Hold and wait-for relationships.

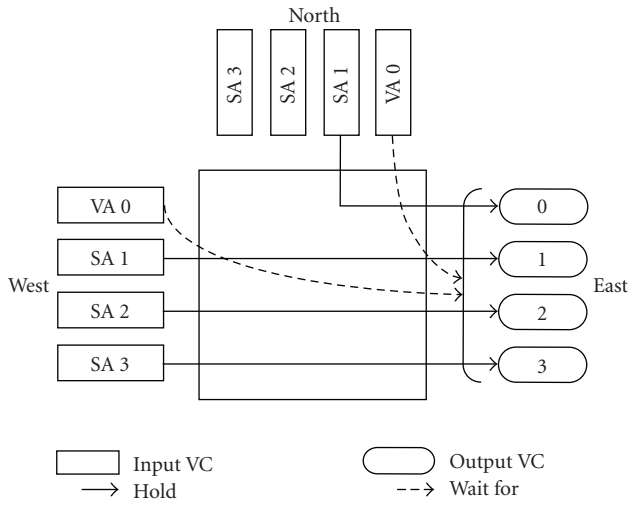


Figure 8: An example of the deadlock.

hold edges as wait-for edges in the opposite direction gives the graph of Figure 7(b). The cycle in this graph shows that the architecture is deadlocked.

Let us illustrate the deadlock in a router in a mesh network (Figure 8). In the west input port, the VC0 has a VA request for any VCs in the east output PC while the other three VCs have already held the VC1, VC2, and VC3 in the east output PC, respectively. Similarly, in the north input port, the VC0 has a VA request for any VCs in the east output PC, the VC1 holds the VC0 in the east output PC, and the VC2 and VC3 occupy the VCs in other output PCs. In both the west and the north input ports, the VC0 has the highest priority. Therefore, in the speculative architecture, the SA requests at the two input ports always fail in the SVA, causing that the four held VCs in the east output PC can no longer be released. On the other hand, although the two VA requests always succeed in the SVA, they always fail to find a free output VC because all four VCs in the east output PC are always being occupied. As a result, they always keep the highest priorities in the corresponding input ports.

But, in the nonspeculative architecture, the two VA requests are considered as invalid because there is no free output VC for them. As a result, they do not win in the SVA although they have the highest priorities. Instead, the four SA requests for the east output PC win the SVA in a round-robin way until one of them sends a tail flit and releases the output VC held by it. Then, at the next cycle, the two VA requests are all valid and one of them is allocated the released output VC. In summary, the VA group does not hold the priority group when there is no free output VC and thus breaks the cyclic dependence.

6. Critical Path Analysis

6.1. Critical Paths for the Generic VA/SA. Figure 9(a) shows the critical path for the generic VA. Firstly, an input VC checks whether there are free output VCs in the destined output PC. Then, it selects one from all free output VCs using the arbiter in the 1st stage. After that, a VA request is generated and sent to the 2nd stage arbiter for the selected output VC. Once the input VC is successfully allocated, the selected output VC, status of this output VC is updated to be busy. Figure 9(b) demonstrates the critical path for the generic SA. In the beginning, all 1st stage SA requests generated in the previous clock cycle enter the 1st stage arbiters for arbitration. Then, requests for the 2nd stage arbiters are generated and arbitrated. After that, the 1st stage requests for the next cycle are produced. The max frequency for a router with generic VA/SA architectures is determined by the critical path of the generic VA because it is longer than that of the generic SA.

6.2. VA Simplification Effect on Critical Paths. As described in Section 4, there are three methods for VA simplification: reducing number of VA arbiters, sharing of $V : 1$ arbiters, and combining VA and SA arbiters. In the following, we explain their effects on the VA/SA critical paths one by one.

Firstly, reducing the number of $p_iV : 1$ arbiters from V to 1 in an output PC does not produce additional delay for the VA because no logics are needed to detect conflicts. After changing the RC function to return a single free VC of an output PC, at most one VC in an output PC will be requested at each cycle. Thus, one $p_iV : 1$ arbiter is enough and no additional logics are required to determine which VC is to use the single $p_iV : 1$ arbiter. On the contrary, the critical path of the VA can be reduced in two aspects. One is that the 1st arbitration stage is removed. The other is that logics for VA request generation are simplified because the number of $p_iV : 1$ arbiters is largely reduced.

Secondly, sharing $V : 1$ arbiters in an input port increases the critical path of the VA. Before sharing, a $p_iV : 1$ arbiter in the 2nd arbitration stage of the VA is implemented as the tree architecture shown in Figure 3(b) where the $V : 1$ arbiter and the $p_i : 1$ arbiter are in parallel. After the sharing, a $p_iV : 1$ arbiter is realized as the architecture shown in Figure 4 where the $V : 1$ arbiter and the $p_i : 1$ arbiter are in serial. Meanwhile, additional logics after the $V : 1$ arbiter are required to generate requests for the $p_i : 1$ arbiter.

Finally, combining VA and SA arbiters increases the critical path of the SA. Before the combination, the 1st stage SA requests directly enter the 1st stage arbiters for arbitration. After the combination, the 1st stage SA requests have to wait for the results of the free output VC check block

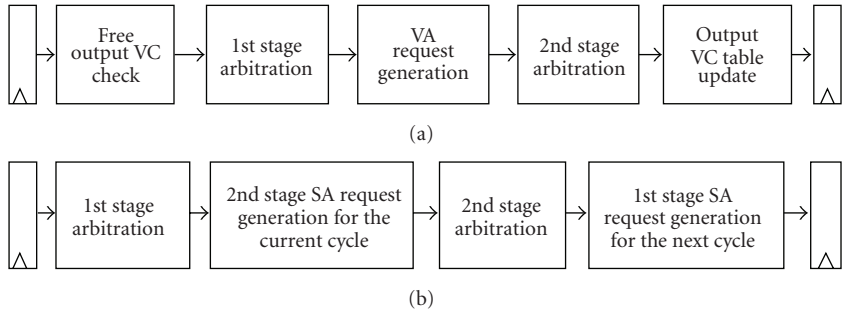


Figure 9: Critical paths for the generic VA (a) and the generic SA (b).

Table 7: Network and process parameters.

Traffic pattern	Uniform / Hotspot* / Transpose
Topology	4×4 mesh; 6×6 mesh
Flow control	Virtual channel
Routing	XY
Buffer management	Credit-based
Router radix	3/4/5 ports
Buffer architecture	2/4/6 VCs per port, 4 flits per VC
Packet length	4/8/16 flits
Flit size	32 (random payload) + 4 (overhead)
Technology	130 nm, HS
Frequency	250 MHz

*The hotspot routers are router_{11}, and router_{22}, router_{31}. They inject packets to the network with a 1.5× rate.

before they enter the 1st stage arbiters in order to avoid the deadlock problem described in Section 5.

In summary, total effects on the VA/SA critical paths depend on which simplification methods are applied.

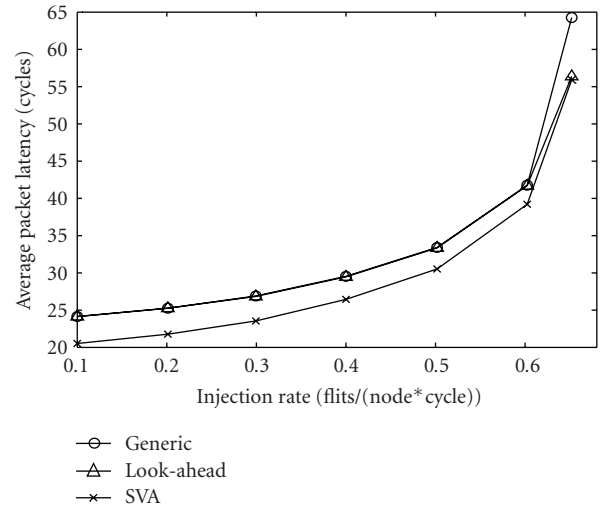
7. Evaluations

We evaluated three allocation architectures: a generic VA and a generic SA (the generic), a look-ahead VA and a generic SA (the look-ahead), and a combined VA and SA (the SVA). Evaluations were based on simulations on the entire NoCs instead of the allocation components themselves in order to get more realistic results. In the look-ahead VA, the number of $p_iV : 1$ arbiters decreases from p_oV to p_o in the second stage and the $p_iVV : 1$ arbiters in the first stage are replaced by $p_oV : 1$ comparators. All other components of the NoCs are the same in the three architectures being studied. The network and process parameters are shown in Table 7.

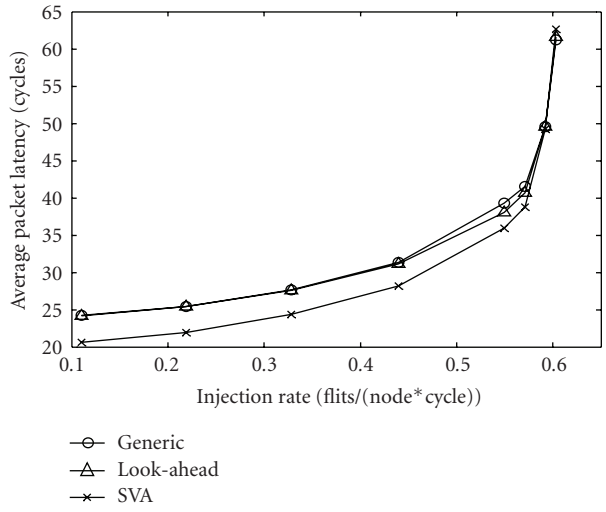
7.1. Network Performances. Network performances were obtained by a simulator built in SystemVerilog. Evaluations were performed for various network sizes, various Vs, various packet lengths, and a range of traffic patterns to validate whether the conclusion that the VA simplification presented in Section 4 has small effect on network performances is general. Saturation is defined as the highest level of injection rate for which the average throughput equals to the injection rate [14]. We only compared latencies before saturation.

Average packet latency as a function of traffic injection rate is plotted for different traffic patterns in Figure 10. The curve of the generic architecture nearly overlaps that of the look-ahead architecture for all tested traffic patterns. It means that the pb^k remains small and arbiter reduction has little impact on network latencies for different traffic patterns. The latency of the SVA is significantly smaller than the other two architectures at low and moderate network loads and becomes almost the same at high network loads. Figure 11 shows results when packet length is 8 and 16 flits, respectively. We can see that reduction of latency by the SVA remains significant at high injection rates. Figure 12 describes results for 6×6 mesh. The trend is similar to that for 4×4 mesh. The SVA can reduce latency because it removes the VA pipeline stage for head flits. In addition, similar trends can be observed for various Vs (keeping the 4×4 network size, uniform traffic pattern, and 4 flits per packet). The results for various Vs are not shown for clarity.

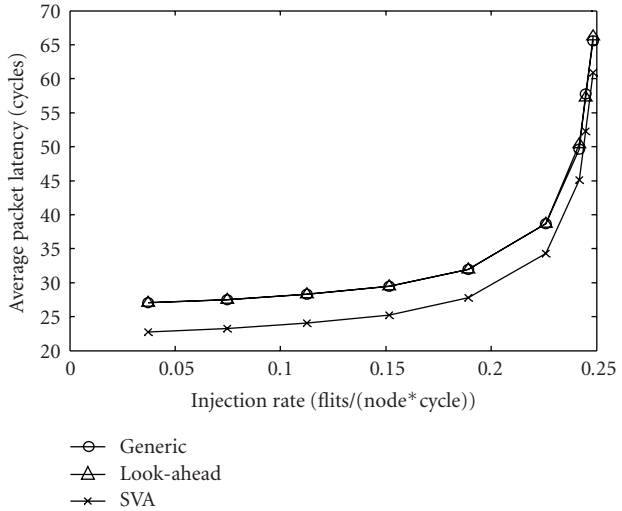
7.2. Maximum Router Frequency. Table 8 summarizes maximum router frequencies for the three allocation architectures. The frequencies were obtained from Synopsys DC with worst case synthesis condition. The generic architecture and the SVA have similar frequencies while the look-ahead architecture always has higher frequencies than the others. The reason is that only the simplification method of reducing the number of VA arbiters (*method 1*) is used in the look-ahead architecture while all three simplification means are applied in the SVA. As presented in Section 6, the *method 1* reduces the critical path delay whereas sharing of $V : 1$ arbiters (*method 2*) and combing VA and SA arbiters (*method 3*) increase the critical path delay. Therefore, router speed for the look-ahead architecture is always the highest while the speed for the SVA depends on the total effects of all the three methods. For example, when p_i is three and V is four, the SVA is faster than the generic because the delay reduction caused by *method 1* is larger than the delay increase caused by *method 2* and *method 3*. When p_i is four and V is four, the SVA is slower than the generic because the delay reduction is smaller than the delay increase.



(a)

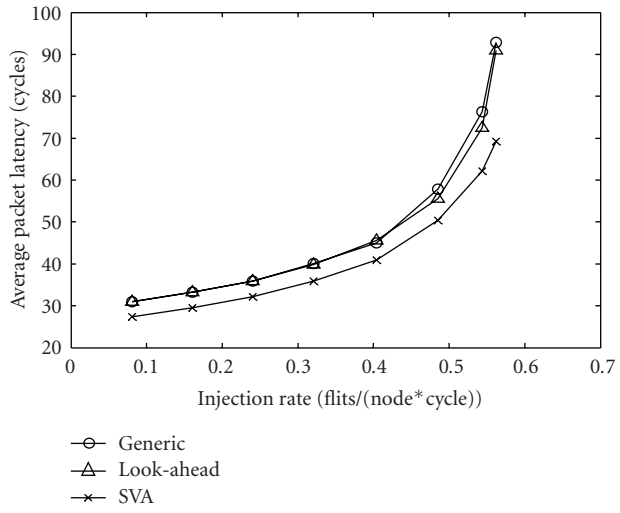


(b)

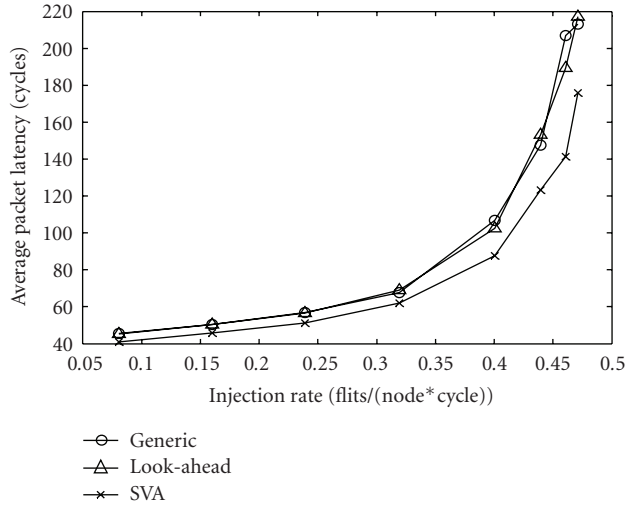


(c)

Figure 10: Average packet latency for various traffic patterns when network size is 4×4, and V and packet length are 4. (a) Uniform. (b) Hotspot. (c) Transpose.



(a)



(b)

Figure 11: Average packet latency for other packet lengths when network size is 4×4, V is 4, and traffic pattern is uniform. (a) 8 flits. (b) 16 flits.

Table 8: Max router frequency (MHz).

Router parameters	Generic	Look-ahead	SVA
$p_i = 3, V = 4$	427	526	442
$p_i = 4, V = 4$	403	476	388
$p_i = 5, V = 2$	435	526	435
$p_i = 5, V = 4$	385	435	385
$p_i = 5, V = 6$	323	385	333

7.3. Area and Power at a Certain Frequency. Table 9 shows area costs of the three allocator architectures at the frequency of 250 MHz. The look-ahead architecture reduces area by decreasing the number of $p_i V : 1$ arbiters from $p_o V$ to p_o and removing all the first stage arbiters of a generic VA. The SVA reduces even more area through sharing $V : 1$ arbiters at each input port for a generic VA and combining

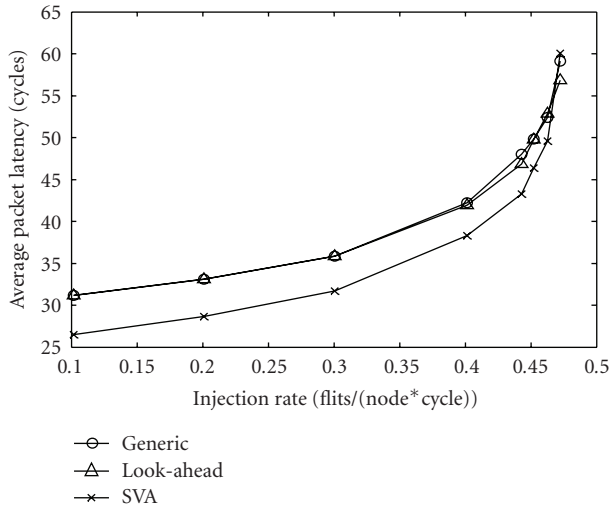


Figure 12: Average packet latency for 6×6 mesh when V is 4, packet length is 4 and traffic pattern is uniform.

Table 9: Area (gate count) of the three allocators

Router parameters	Generic	Look-ahead	SVA
$p_i = 3, V = 4$	5355	2525	2123
$p_i = 4, V = 4$	10433	4586	3785
$p_i = 5, V = 2$	3697	2973	2437
$p_i = 5, V = 4$	17676	7570	5581
$p_i = 5, V = 6$	49314	13581	9873

arbiters of a generic VA and arbiters of a generic SA. Higher proportion of area is reduced as p_i or V increases. For a 5-port, 4-VC router, the look-ahead architecture and the SVA reduce allocator area by 57.17% and 68.43%, respectively. They reduce router area by 23.65% and 28.30%, respectively, because the generic architecture consumes 41.36% area in the router.

In the same way as described in Section 3, we calculated power consumption at 250 MHz for many injection rates of uniform traffic pattern because power is highly related to network loads. Table 10 demonstrates power consumed by the three allocator architectures at zero-load and saturated-load. The look-ahead architecture reduces power at both zero-load and saturated-load for all tested parameters. The SVA has the lowest power at zero-load for all cases because it has the smallest clock network and the smallest number of registers. However, power consumption of the SVA at saturated-load surpasses that of the look-ahead architecture for the three 5-port cases and even surpasses that of the generic architecture for the 5-port, 2-VC case. The reason is as follows. On the one hand, the SVA reduces logic gates and thus reduces power consumption (positive effect). On the other hand, the sharing of logics in the SVA associates more logics together, and thus makes it possible that a logic transition of a net will lead to logic transitions of more nets. As a result, it increases average logic transition and therefore increases power consumption (negative effect). For most cases, the positive effect is larger and thus the SVA consumes smaller power consumption. For other cases, the negative effect is larger and thus the SVA consumes larger power consumption.

Table 10: Power (mW) of the three allocators (zero-load | saturated-load)

Router parameters	Generic	Look-ahead	SVA
$p_i = 3, V = 4$	0.54 | 2.13	0.39 | 1.65	0.32 | 1.58
$p_i = 4, V = 4$	0.67 | 3.51	0.46 | 2.67	0.36 | 2.64
$p_i = 5, V = 2$	0.45 | 1.44	0.43 | 1.44	0.34 | 1.61
$p_i = 5, V = 4$	0.81 | 4.93	0.52 | 3.80	0.41 | 3.96
$p_i = 5, V = 6$	1.20 | 8.65	0.56 | 5.23	0.44 | 5.46

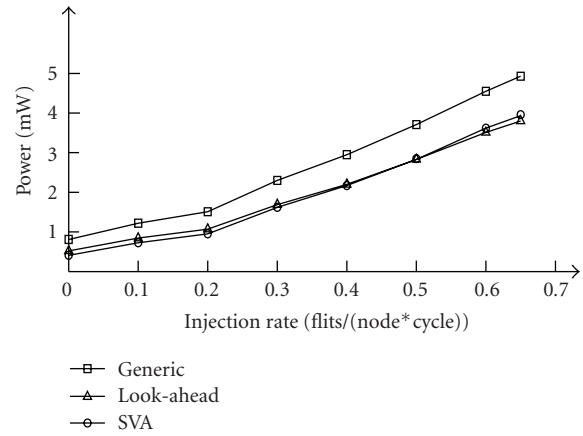


Figure 13: Power of the allocators at various injection rates.

Power as a function of injection rate for the 5-port, 4-VC case is plotted in Figure 13. It can be seen that the difference in power between the look-ahead architecture and the SVA is small. On average, the look-ahead architecture and the SVA save allocator power by 27.18% and 30.78%, respectively. Considering that the generic allocators consume 32.65% power in the router, the look-ahead architecture and the SVA reduce power by 8.87% and 10.05%, respectively for the router. Router power savings increase to 16.44% (the look-ahead) and 17.07% (the SVA) for the 5-port, 6-VC router.

7.4. Discussion. When networks run at frequencies which cannot be met by the SVA, the look-ahead architecture is the only reasonable choice. Otherwise, if networks run at frequencies which can be achieved by the SVA, the SVA is better because it provides lower packet latency (in cycles) as well as lower area and power costs for most design cases.

8. Conclusion

We concentrate on designs of low-cost switch and VC allocators in this paper. Instead of studying the allocators in isolation, we study them in the context of a NoCs. Area and power reduction opportunities are identified for the generic architecture through analyses and statistics. As a result, two low-cost allocator architectures are presented and discussed.

We did comprehensive evaluations for the allocators, including network-level performances, frequency, area, and power. Results show that both the look-ahead architecture and the SVA achieve lower costs than the generic architecture without any adverse effects on network performances. The look-ahead architecture and the SVA have different application domains that are determined by the frequency constraint.

Acknowledgment

The work reported is supported by a CUHK Direct Grant no. 2050429.

References

[1] "ITRS, International Technology Roadmap for Semiconductors," 2007.

[2] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayashima, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, 2007.

[3] S.-J. Lee, K. Lee, and H.-J. Yoo, "Analysis and implementation of practical, cost-effective networks on chips," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 422–433, 2005.

[4] U. Y. Ogras and R. Marculescu, ""It's a small world after all": NoC performance optimization via long-range link insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, 2006.

[5] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA '04)*, vol. 31, pp. 188–197, Munich, Germany, June 2004.

[6] L.-S. Peh and W. J. Dally, "Delay model and speculative architecture for pipelined routers," in *Proceedings o the 7th International Symposium on High-Performance Computer Architecture*, pp. 255–266, October 2001.

[7] M. Zhang and C.-S. Choy, "Low-cost VC allocator design for virtual channel wormhole routers in networks-on-chip," in *Proceedings of the 2nd IEEE International Symposium on Networks-On-Chip (NOCS '08)*, pp. 207–208, April 2008.

[8] W. J. Dally, "Virtual-channel flow control," in *Proceedings of the 17th Annual Symposium on Computer Architecture*, pp. 60–68, Seattle, Wash, USA, May 1990.

[9] Z. Lu, M. Liu, and A. Jantsch, "Layered switching for networks on chip," in *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07)*, pp. 122–127, San Diego, Calif, USA, June 2007.

[10] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: a dynamic virtual channel regulator for network-on-hip routers," in *Proceedings of the 39th Annual International Symposium on Microarchitecture (MICRO '06)*, pp. 333–346, Orlando, Fla, USA, December 2006.

[11] H. S. Wang, L. S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proceedings of the International Symposium on Microarchitecture*, pp. 105–116, 2003.

[12] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2919–2933, 2006.

[13] Z. Lu and A. Jantsch, "Admitting and ejecting flits in wormhole-switched networks on chip," *IET Computers and Digital Techniques*, vol. 1, no. 5, pp. 546–556, 2007.

[14] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, Calif, USA, 2004.