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.

Copyright © 2009 M. Zhang and C.-S. Choy. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 replaced 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.

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, post-synthesis netlist, wire load model, and post-synthesis switching activities as inputs. The switching activities of router30 (a 3-port router; we define the left-bottom router, as routerR0 for the 4 × 4 mesh throughout the paper), router31 (a 4-port router), and router32 (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.
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:

$$R = \begin{bmatrix}
  r_{i11}^{1V} & r_{i11}^{1V} & \cdots & r_{i11}^{p_i1V} \\
  r_{i11}^{1V} & \cdots & \cdots & \cdots \\
  \vdots & \ddots & \ddots & \vdots \\
  r_{i11}^{p_i1V} & \cdots & \cdots & r_{i11}^{1V} \\
  \vdots & \vdots & \vdots & \vdots \\
  r_{i11}^{p_iV} & \cdots & \cdots & r_{i11}^{p_iV} \\
\end{bmatrix},$$  \hspace{1cm} (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 $4 \times 4$ mesh NoCs under uniform, hotspot, and transpose traffic patterns and calculated $pbk$ by counting the number of cycles when $r^k$ is 2 or above and then dividing by the number of simulation cycles. The
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 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 $pb^{1^k}$ for the three traffic patterns (Table 4). We can see that results of $pb^{1^k}$ are much larger than results of $pb^k$.

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:

\[
R = \begin{bmatrix}
\sum_{k=1}^{p_i} r_{1k} & \ldots & \sum_{k=1}^{p_i} r_{pk} \\
\vdots & \ddots & \vdots \\
\sum_{k=1}^{p_i} r_{1k} & \sum_{k=1}^{p_i} r_{pk} & \sum_{k=1}^{p_i} r_{pk}
\end{bmatrix}
\]  

(2)

4.3. Sharing of $V : 1$ Arbiters. The VA after simplification now requires $p_o p_iV : 1$ 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 2: Parameter list.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>The request matrix</td>
</tr>
<tr>
<td>$r_{mn}^k$</td>
<td>The request from the $n$th VC at the $m$th input port to the $k$th VC at the $k$th output port. The value is 1 if the request is valid. Otherwise, the value is 0.</td>
</tr>
<tr>
<td>$r_{mn}$</td>
<td>The request from the $n$th input VC at the $m$th input port to any VC at the $k$th output port. The value is 1 if the request is valid.</td>
</tr>
<tr>
<td>$p_{b1}^k$</td>
<td>The probability that one VC at the $k$th output port is requested.</td>
</tr>
<tr>
<td>$p_{b1}^k$</td>
<td>The probability that multi-VCs at the $k$th output port are concurrently requested.</td>
</tr>
<tr>
<td>$p_{b1m}$</td>
<td>The probability that VCs in the $m$th input port request VCs at multi output ports.</td>
</tr>
</tbody>
</table>

**Table 3: Results of $pb^k$.**

<table>
<thead>
<tr>
<th>Traffic pattern</th>
<th>Output PC</th>
<th>West</th>
<th>North</th>
<th>East</th>
<th>South</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td></td>
<td>0.21%</td>
<td>0.23%</td>
<td>0.32%</td>
<td>0.33%</td>
<td>0.16%</td>
</tr>
<tr>
<td>Hotspot</td>
<td></td>
<td>0.23%</td>
<td>0.10%</td>
<td>0.30%</td>
<td>0.48%</td>
<td>0.30%</td>
</tr>
<tr>
<td>Transpose</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.01%</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 4: Results of $pb^{1k}$.**

<table>
<thead>
<tr>
<th>Traffic pattern</th>
<th>Output PC</th>
<th>West</th>
<th>North</th>
<th>East</th>
<th>South</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td></td>
<td>13.78%</td>
<td>12.43%</td>
<td>16.28%</td>
<td>17.53%</td>
<td>15.43%</td>
</tr>
<tr>
<td>Hotspot</td>
<td></td>
<td>12.68%</td>
<td>11.26%</td>
<td>16.86%</td>
<td>13.88%</td>
<td>13.41%</td>
</tr>
<tr>
<td>Transpose</td>
<td></td>
<td>0</td>
<td>7.90%</td>
<td>0</td>
<td>16.48%</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 4: A generic switch allocator.**
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 p1 : 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 p1 : 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 reduction of arbiters.

More importantly, as shown in Figure 5, combining a VA and a SA removes the VA pipeline stage for reduction of arbiters.
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 p1V : 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 p1V : 1 arbiter is enough and no additional logics are required to determine which VC is to use the single p1V : 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 p1V : 1 arbiters is largely reduced.

Secondly, sharing V : 1 arbiters in an input port increases the critical path of the VA. Before sharing, a p1V : 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 p1 : 1 arbiter are in parallel. After the sharing, a p1V : 1 arbiter is realized as the architecture shown in Figure 4 where the V : 1 arbiter and the p1 : 1 arbiter are in serial. Meanwhile, additional logics after the V : 1 arbiter are required to generate requests for the p1 : 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.
depend on which simplification methods are applied. The number of
in order to get more realistic results. In the look-ahead VA,
etire NoCs instead of the allocation components themselves
(the SV A). Evaluations were based on simulations on the
generic SA (the look-ahead), and a combined VA and SA
We evaluated three allocation architectures: a generic VA
and a generic SA (the look-ahead), a look-ahead VA and a
combined VA and SA and a generic SA (the generic), a look-ahead VA and a

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
total 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 \( p_iV \) to \( p_i \)
in the second stage and the \( p_iV V : 1 \) arbiters in the first stage
are replaced by \( p_iV : 1 \) 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 summarizes maximum router frequencies for the three allocation architec-
tures. 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
casted 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.

7.1. Network Performances. Network performances were
obtained by a simulator built in SystemVerilog. Evaluations
were performed for various network sizes, various \( V_s \), 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 \( pb^0 \) 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 \( 6 \times 6 \) mesh. The trend is similar to that for \( 4 \times 4 \) 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 \( V_s \) (keeping the \( 4 \times 4 \) network size, uniform traffic pattern, and 4 flits per packet). The results for various \( V_s \) are not shown for clarity.

Table 7: Network and process parameters.

<table>
<thead>
<tr>
<th>Traffic pattern</th>
<th>Uniform / Hotspot° / Transpose</th>
<th>Topology</th>
<th>4 × 4 mesh; 6 × 6 mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow control</td>
<td>Virtual channel</td>
<td>Router radix</td>
<td>XY</td>
</tr>
<tr>
<td>Buffer management</td>
<td>Credit-based</td>
<td>Buffer architecture</td>
<td>2/4/6 VCs per port, 4 flits per VC</td>
</tr>
<tr>
<td>Packet length</td>
<td>4/8/16 flits</td>
<td>Flit size</td>
<td>32 (random payload) + 4 (overhead)</td>
</tr>
<tr>
<td>Technology</td>
<td>130 nm, HS</td>
<td>Frequency</td>
<td>250 MHz</td>
</tr>
</tbody>
</table>
| °The hotspot routers are router11, and router22, router31. They inject packets to the network with a 1.5× rate.

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

7.2. Maximum Router Frequency. Table 8 summarizes max-
imum router frequencies for the three allocation architec-
tures. 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
casted 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.
Figure 10: Average packet latency for various traffic patterns when network size is $4 \times 4$, and $V$ and packet length are 4. (a) Uniform. (b) Hotspot. (c) Transpose.

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

Table 8: Max router frequency (MHz).

<table>
<thead>
<tr>
<th>Router parameters</th>
<th>Generic</th>
<th>Look-ahead</th>
<th>SVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 = 3, V = 4$</td>
<td>427</td>
<td>526</td>
<td>442</td>
</tr>
<tr>
<td>$p_1 = 4, V = 4$</td>
<td>403</td>
<td>476</td>
<td>388</td>
</tr>
<tr>
<td>$p_1 = 5, V = 2$</td>
<td>435</td>
<td>526</td>
<td>435</td>
</tr>
<tr>
<td>$p_1 = 5, V = 4$</td>
<td>385</td>
<td>435</td>
<td>385</td>
</tr>
<tr>
<td>$p_1 = 5, V = 6$</td>
<td>323</td>
<td>385</td>
<td>333</td>
</tr>
</tbody>
</table>

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 $p_1 V : 1$ arbiters from $p_0 V$ to $p_0$ 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...
Most cases, the positive effect increases power consumption (negative effect). As a result, it increases average logic transition and transition of a net will lead to logic transitions of more logics together, and thus makes it possible that a logic the other hand, the sharing of logics in the SV A associates proportion of area is reduced as arbiters of a generic VA and arbiters of a generic SA. Higher

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

<table>
<thead>
<tr>
<th>Router parameters</th>
<th>Generic</th>
<th>Look-ahead</th>
<th>SV A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i = 3, V = 4$</td>
<td>5355</td>
<td>2525</td>
<td>2123</td>
</tr>
<tr>
<td>$p_i = 4, V = 4$</td>
<td>10433</td>
<td>4586</td>
<td>3785</td>
</tr>
<tr>
<td>$p_i = 5, V = 2$</td>
<td>3697</td>
<td>2973</td>
<td>2437</td>
</tr>
<tr>
<td>$p_i = 5, V = 4$</td>
<td>17676</td>
<td>7570</td>
<td>5581</td>
</tr>
<tr>
<td>$p_i = 5, V = 6$</td>
<td>49314</td>
<td>13581</td>
<td>9873</td>
</tr>
</tbody>
</table>

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 SV A 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 SV A 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 SV A reduces logic gates and thus reduces power consumption (positive effect). On the other hand, the sharing of logics in the SV A 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 SV A

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

<table>
<thead>
<tr>
<th>Router parameters</th>
<th>Generic</th>
<th>Look-ahead</th>
<th>SV A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i = 3, V = 4$</td>
<td>0.54</td>
<td>2.13</td>
<td>0.39</td>
</tr>
<tr>
<td>$p_i = 4, V = 4$</td>
<td>0.67</td>
<td>3.51</td>
<td>0.46</td>
</tr>
<tr>
<td>$p_i = 5, V = 2$</td>
<td>0.45</td>
<td>1.44</td>
<td>0.43</td>
</tr>
<tr>
<td>$p_i = 5, V = 4$</td>
<td>0.81</td>
<td>4.93</td>
<td>0.52</td>
</tr>
<tr>
<td>$p_i = 5, V = 6$</td>
<td>1.20</td>
<td>8.65</td>
<td>0.56</td>
</tr>
</tbody>
</table>

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

7.4. Discussion. When networks run at frequencies which cannot be met by the SV A, the look-ahead architecture is the only reasonable choice. Otherwise, if networks run at frequencies which can be achieved by the SV A, the SV A 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

Submit your manuscripts at http://www.hindawi.com