Virtual Channels Planning for Networks-on-Chip

Ting-Chun Huang, Umit Y. Ogras, Radu Marculescu
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
{tingchun, uogras, radum}@cmu.edu

Abstract

The virtual channel flow control (VCFC) provides an efficient implementation for on-chip networks. However, allocating the virtual channels (VCs) uniformly results in a waste of area and significant leakage power, especially at nanoscale. To remedy this situation, we propose a novel approach for customizing the virtual channels allocation based on the traffic characteristics of the target application. Towards this end, we first develop an algorithm that calculates the port contention rates and expected bandwidth at each router in the network. Using this information, we add VCs only to the channels with the highest bandwidth usage. Our simulation results involving synthetic and real applications show more than 40% buffer savings compared to uniform VC allocation, while achieving similar performance levels.

1. Introduction

The Network-on-Chip (NoC) approach has been recently proposed for efficient communication in System-on-Chip (SoC) designs [1]. Simply stated, in NoCs the processing elements (PEs) communicate by exchanging messages over a network instead of using dedicated wires. NoC architectures have several advantages over the traditional bus-based communication. First, NoCs offer better scalability due to design reuse; this also helps minimizing the design effort at the integration stage. Second, the wire length and delay of the physical links can be better controlled. As a result, higher throughput can be typically achieved by pipelined transmission. Third, the power consumption can be reduced by using short communication links rather than sharing long buses.

Despite these advantages, link contention is a big issue in NoC designs, especially for real-time or communication-intensive applications. From this perspective, the on-chip networks must be carefully designed to meet the delay and throughput requirements. Since NoC design requirements are significantly different from those enforced in macro-networks, new design techniques are needed [1] [2].

With respect to the flow control strategy, wormhole routing is well suited for NoCs, since it can provide low latency with moderate buffering requirements. However, wormhole routing without using virtual channels (VCs) is prone to head of line (HOL) blocking which is a significant performance limiting factor. Allocating the buffer space judiciously can improve the performance [3], but this does not eliminate the HOL blocking. As a result, the link usage still remains low even when the available buffer space is large. Virtual channel flow control (VCFC), on the other hand, provides an alternative for improving the network performance.

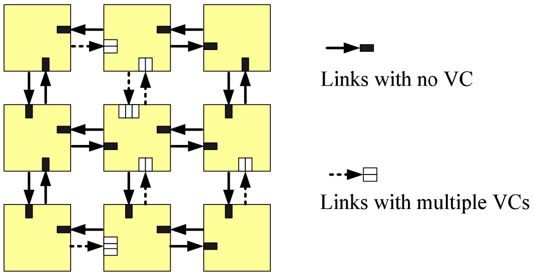


Fig 1: NoC with non-uniform VC configuration.

Simply speaking, the HOL blocking happens when the packet at the head of the FIFO in the current router is not able to propagate to the FIFO in the next router. When the HOL blocking occurs, all subsequent packets remain blocked and thus the router output ports can starve. For instance, due to HOL blocking, the throughput of the links is typically limited to 58% under uniform traffic with fixed packet length [4]. In contrast, the use of VCFC decouples the buffers from the physical communication links and routers. As such, when a certain VC is congested, the packets in other VCs can still progress through some links in the network and so the network throughput can be significantly improved.

Although the VCFC is able to increase the network throughput while reducing the transmission delay, allocating too many VCs may end up in wasting the network resources. More importantly, allocating extra VCs to the links with light traffic loads brings only marginal performance improvements. For instance, Fig.1 shows the input buffer structure of a non-uniform VC configuration for a 3×3 NoC. In this example, only the links with high utilization (shown with dashed arrows) can benefit from allocating extra VCs. Besides this, since the uniformly allocated VCs consume extra buffering space, the implementation cost and power consumption increase significantly [5][6][7]. Therefore, careful VC planning is essential in order to provide high performance with low costs and low energy consumption.

In this paper, we propose a novel technique for NoC performance improvement via application-specific VC planning. In this approach, the VCs are allocated *non-uniformly* based on the traffic characteristics; this way, we

Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED'07)
0-7695-2795-7/07 $20.00 © 2007 IEEE
IEEE COMPUTER SOCIETY

can maximize the network performance while using only limited VC resources. Specifically, our contributions are:

- Propose a non-uniform VC planning for performance optimization subject to resource constraints.
- Evaluate extensively the on-chip network performance using real and synthetic benchmarks.

The rest of the paper is organized as follows. In Section 2, we review the previous work and highlight our contributions. In Section 3, we present an overview of the proposed approach and discuss some implementation issues. The non-uniform VC allocation problem and the proposed solution are presented in Section 4. The experimental results are provided in Section 5. Finally, we summarize our main contribution and discuss future work in Section 6.

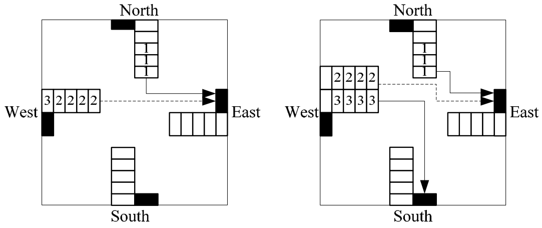
2. Related work and novel contributions

NoC communication architecture is introduced in [1] and important issues for NoC design, such as application mapping, routing, floorplanning, synchronization are discussed in [2]. Traditionally, the VCFC approach was proposed to eliminate deadlock in macro-networks [8]. It has been also shown that better performance can be achieved with wormhole routing [9] by smartly managing the transmission queues and physical links. Recently, AETHEREAL [10] and MANGO [11] used VCs to provide virtual circuit switching for guaranteed service in NoCs. Rezazad et al. in [12] show the effect of VCs with fixed total buffering space.

In the past, dynamic power was the main concern in NoCs [5]. However, as the technology shrinks, leakage power is not negligible anymore. For instance, the leakage power will take up to 17.4% of power at a router for NoCs implemented in 0.07 um [7]. In particular, about 60% of leakage power, which is equivalent to about 10% of the total router power consumption, is consumed in the buffers. Since VCFC implementation requires extra buffers, the routers power consumption will increase with increasing number of VCs [6].

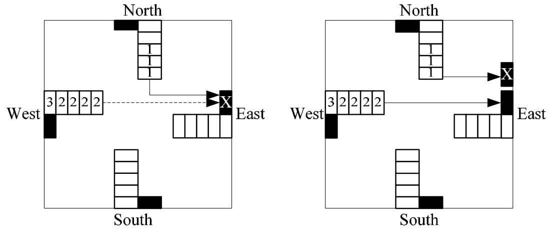
Previous approaches [12][13] use uniform allocation of VCs instead of customizing the VC configuration to optimize the performance/cost tradeoff. A first step towards buffer customization is made in [3], where the authors solve a queuing problem at each buffer and improve the network performance by reducing the queue blocking probability. However, in many practical cases, the main performance limiting factor comes from the HOL blocking instead. Since no VCs are considered in [3], the issue of customized VC planning is still an open problem. In this work, we address this problem and show how to increase the router throughput by smartly allocating extra VCs when needed.

From an implementation standpoint, the input-queued router is attractive for on-chip implementation due to its simplicity. As such, our subsequent discussion focuses on input-queued switches. We propose an application-specific, non-uniform VC configuration for on-chip mesh networks. As shown later in the paper, this greatly reduces the HOL blocking probability and enhances the router throughput. Other issues, such as router and link arbitration, also affect the network performance. We adopt the iSLIP technique proposed in [14] for router arbitration and the Round Robin algorithm for link arbitration. More advanced arbitration techniques are discussed in [15].



(a) (b)

Fig 2: Router HOL blocking. The white and black rectangles are the input and output ports.



(a) (b)

Fig 3: Router HOL blocking.

3. Overview of the approach

3.1 Basic idea

In this section, we identify the HOL blocking problem and explain how VC allocation can improve the network performance. The HOL blocking at router-level happens as shown in figures 2 and 3. In Fig.2(a), packet 1 from the North input port uses the East output port, while packet 2, also heading to the East port, is blocked. Since packet 3 is queued after packet 2, packet 3 is also blocked, although its destination, namely the South port, is available. If there are VCs implemented at the West input port, then packet 3 is able to reach its destination instead of being blocked, as shown in Fig.2(b). Hence, this greatly improves the router throughput.

Adding output VCs can also solve the congestion problem. In Fig.3(a), packet 1 is under transmission, but the East output port is blocked because the buffer at the East router is used. The blocking effect propagates to all the packets leading to the same output port, such as packet 2 and all the packets queued after packet 2. In Fig.3(b), an extra output VC at the East output port can provide an alternate connection for packet 2, and make use of the idle switch bandwidth, thus increasing the router throughput.

Compared to the straightforward buffer allocation, our work focuses on reducing the probability of HOL blocking, which is an important limiting factor for the network performance. As we show later in the paper, with suitable number of VCs allocated to the critical links, the performance of the network can be improved significantly, with a small overhead in area and energy consumption.

3.2 Implementation issues

An output-input port pair across any link must have the same number of VCs. Therefore, our VC allocation algorithm allocates the VCs in a pair-wise manner. Since different VCs belonging to the same port cannot use the switch simultaneously, the packet arbitration and flit scheduling are handled separately. When a new packet (*i.e.* head flit) arrives at or a packet (*i.e.* tail flit) leaves a router, the arbiter is responsible for setting up and tearing down the connections. After a connection is set up, the scheduler decides on flits transmission in each clock cycle. Detailed implementation issues for a router with uniform VCs can be found in [9]. Asymmetric router design with a different number of input and output port VCs can be easily obtained by a small modification of the arbiter in a typical VC router.

We also note that the proposed approach is static, *i.e.* the VC allocation is performed based on average case analysis, and the VC configuration cannot be changed online. Such a static approach may present a drawback for macro-networks where the variability in the network traffic is large. However, the NoC traffic is more predictable. Moreover, flow control techniques such as the one presented in [16], decrease the variation in the on-chip network traffic. Hence, the proposed non-uniform VC allocation technique can be employed in NoCs in conjunction with effective flow control algorithms.

4. Non-uniform virtual channel allocation

Implementing VCs needs extra buffer space and control logic to handle the flits transmission. Our basic idea for VC planning is to allocate the VC resources only when necessary. More precisely, we allocate the VCs at these router ports where the congestion is most likely to occur. From our experimental results, the effect of this non-uniform (or "on-demand") VC allocation can greatly improve the network throughput, while only allowing a limited number of VCs in the network. In what follows, we present the problem formulation and discuss its analytical solution.

4.1 Problem formulation

We summarize the notation used during the analysis in Table 1. Based on this notation, the problem of VCs planning can be formulated as follows:

Given:

- Application Communication Graph, *C(T, F)*
- Network topology, *N(PE, LK)*
- Mapping *M* and a deterministic routing algorithm, *R*
- Number of extra VCs that can be used (total budget), *S*

Determine:

A mapping from the set of links *LK* to a set of positive integers *i.e.* $g_{VC} : LK \rightarrow Z^+$, where for any $L \in LK$ $g_{VC}(L)$ gives the number of VCs attached to link *L*.

Such that:

The average packet latency *D* is minimized, subject to the number of VCs that can be added.

4.2 Probabilistic model

Our model evaluates the congestion that may occur at different routers, and we minimize the average packet latency by removing the cases of possible congestion in the network[1].

At any given router, the matrix **Φ** denotes the *traffic flow rate* through the switch; each entry λ_{ij} in the matrix **Φ** denotes the traffic rate from input port *i* to output port *j*. For application-specific traffic, the information can be obtained by enumerating all the traffic flows passing through the input port *i* and output port *j*, as follows:

Table 1: Parameter notation

Parameter	Description
C(T,F)	Application Communication Graph, a directed graph, where *T* is the set of vertices denoting tasks and *F* is the set of edges denoting the communication load
N(PE,LK)	Network topology, where *PE* is the set of processing elements and *LK* is the set of links
M	Mapping from application tasks *T* to the network *PE*s
S	VC Budget (*i.e.* in number of extra VCs)
D	Average packet delay (in cycles)
Λ	Link bandwidth (in flits/cycle)
W	Maximum number of VC per link
$R(f_c, i, j)$	Routing Algorithm $f_c \in F;\ i, j \in \{\mathrm{L, N, S,E,W}\}$ $\begin{cases} 1 & \text{if } f_c \text{ flows from port } i \text{ to port } j \\ 0 & \text{otherwise} \end{cases}$
$\lambda(f_c)$	Packet injection rate (in flits/cycle) for flow f_c.
λ_{ij}	Traffic flowing rate (in flits/cycle) from port *i* to port *j* in a router $\lambda_{ij} = \sum_{f_c \in F} \lambda(f_c) R(f_c, i, j)$
Φ	Traffic flow matrix, we combine λ_{ij} into a matrix such that $\mathbf{\Phi} = \{\lambda_{ij}\}$
p_{ij}	The conditional probability that a flit is leading to output port *j* given it is from port *i* $p_{ij} = \dfrac{\lambda_{ij}}{\sum_{k \in \{L,N,S,E,W\}} \lambda_{ik}}$
H_i^{in}	Expected contention probability for input port *i*
BW_i	Expected switch bandwidth for input port *i* (in flits/cycle)

$$\lambda_{ij} = \sum_{f_c \in F} \lambda(f_c) R(f_c, i, j) \qquad (1)$$

where $\lambda(f_c)$ is the packet injection rate for flow f_c and *R* is the routing function defined in Table 1. Using this probabilistic model, we first calculate the *expected switch bandwidth* for each link, and find the link with the highest bandwidth utilization to allocate extra VCs.

Equation (1) sums up the traffic arriving rates (in flits/cycle) for all the traffic flows going through the input port *i* and output port *j* at a specific router. For instance, when dealing with minimum routing algorithms, such as XY routing, the entries λ_{ii} are 0; that is, any input port does *not* send packets to output port in the same direction. For each router, we express the routing matrix **Φ** as:

$$\mathbf{\Phi} = \begin{array}{c} \\ \mathrm{L} \\ \mathrm{N} \\ \mathrm{S} \\ \mathrm{E} \\ \mathrm{W} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{L} & \mathrm{N} & \mathrm{S} & \mathrm{E} & \mathrm{W} \end{array} \\ \begin{bmatrix} \lambda_{LL} & \lambda_{LN} & \lambda_{LS} & \lambda_{LE} & \lambda_{LW} \\ \lambda_{NL} & \lambda_{NN} & \lambda_{NS} & \lambda_{NE} & \lambda_{NW} \\ \lambda_{SL} & \lambda_{SN} & \lambda_{SS} & \lambda_{SE} & \lambda_{SW} \\ \lambda_{EL} & \lambda_{EN} & \lambda_{ES} & \lambda_{EE} & \lambda_{EW} \\ \lambda_{WL} & \lambda_{WN} & \lambda_{WS} & \lambda_{WE} & \lambda_{WW} \end{bmatrix} \end{array}$$

The labels L, N, S, E, and W on the left hand side of **Φ** specify the direction of the input port, while the labels on the top specify the direction of the output port.

To analyze the *expected switch bandwidth* BW_i for input ports, we define the *blocking matrix*, **B**, where the entry

[1] We note that although the technique is explained using a mesh network, it can be applied to arbitrary topologies.

b_{ij} denotes the probability that the traffic flow from i to j is blocked by other traffic flows at the router output port:

$$\mathbf{B} = \begin{array}{c} \\ L \\ N \\ S \\ E \\ W \end{array} \begin{array}{c} \begin{array}{ccccc} L & N & S & E & W \end{array} \\ \begin{bmatrix} b_{LL} & b_{LN} & b_{LS} & b_{LE} & b_{LW} \\ b_{NL} & b_{NN} & b_{NS} & b_{NE} & b_{NW} \\ b_{SL} & b_{SN} & b_{SS} & b_{SE} & b_{SW} \\ b_{EL} & b_{EN} & b_{ES} & b_{EE} & b_{EW} \\ b_{WL} & b_{WN} & b_{WS} & b_{WE} & b_{WW} \end{bmatrix} \end{array}$$

As shown in Fig.4(a), the contention happens when more than one input port requests the same output port at the same time. Since the traffic arriving rate from input port k to output port i is λ_{ki}, we express the probability that the output port i is connected to input port k, during a certain cycle, as (λ_{ki} / Λ), where Λ is the link bandwidth.

Since $\sum_{k\in\{L,N,S,E,W\},k\neq i} \lambda_{kj}$ is all the traffic flow leading to output port j except the one from i, we approximate the probability that output port j is connected to some input port other than i by:

$$b_{ij} = \frac{\sum_{k\in\{\mathrm{L,N,S,E,W}\},k\neq i} \lambda_{kj}}{\Lambda} \qquad (2)$$

Actually, this is the probability that the input port i cannot use the output port j due to contention.

Now, we have the blocking matrix **B**; then our next step is to calculate the expected contention probability for each input port. Here, we define the probability matrix **P**, where entry p_{ij} denotes the conditional probability that the flits destination is j, given that the flit is at the head of input port i:

$$\mathbf{P} = \begin{bmatrix} p_{LL} & p_{LN} & p_{LS} & p_{LE} & p_{LW} \\ p_{NL} & p_{NN} & p_{NS} & p_{NE} & p_{NW} \\ p_{SL} & p_{SN} & p_{SS} & p_{SE} & p_{SW} \\ p_{EL} & p_{EN} & p_{ES} & p_{EE} & p_{EW} \\ p_{WL} & p_{WN} & p_{WS} & p_{WE} & p_{WW} \end{bmatrix}$$

Since the traffic flow for each direction is known, p_{ij} can be approximated by the ratio of λ_{ij} to the total traffic arriving rate to input port i:

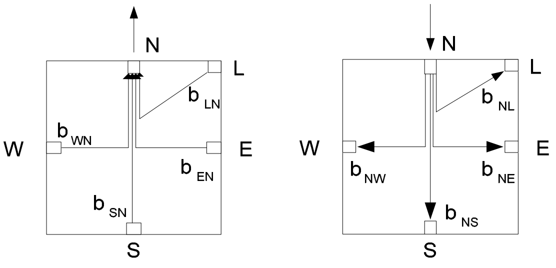
$$p_{ij} = \frac{\lambda_{ij}}{\sum_{k\in\{\mathrm{L,N,S,E,W}\}} \lambda_{ik}} \qquad (3)$$

To analyze the router contention probability at input port i in the absence of VCs, we consider the input port i of a router (*e.g.* North port in Fig.4(b)). Since a flit at input port i has a contention probability b_{ij} to output port j, the expected blocking probability of a flit for input port i is:

$$H_i^{in} = \sum_{j\in\{\mathrm{L,N,S,E,W}\}} (p_{ij} \times b_{ij}) \qquad (4)$$

H_i^{in} is the expected contention probability for flits at input port i. That is, with probability H_i^{in}, the input port will be blocked by the incoming traffic from other port. Thus, the expected maximum available bandwidth for input port i due to contention is then expressed as:

$$BW_i = \Lambda \times (1 - H_i^{in}) \qquad (5)$$



(a) (b)

Fig 4: Blocking analysis at (a) output (b) input port.

Finally, we calculate bandwidth utilization as the ratio between the traffic arrival rate at input port i and the expected bandwidth obtained from (5) as:

$$Bandwidth\ utilization = \frac{\sum_{k\in\{\mathrm{L,N,S,E,W}\}} \lambda_{ik}}{BW_i} \qquad (6)$$

To analyze the blocking probability when there are two VCs present at the input port i, we use an approximate model. Since the port is blocked when both of its VCs are blocked, the blocking probability of this port is approximated by $(H_i^{in})^2$.

Since adding VCs can increase the link throughput, our approach is able to provide good performance improvements. However, the VCs increase the link usage instead of the link bandwidth. Hence, the performance limiting factor morphs into the link bandwidth after the VCs are added to the network. Once the link bandwidth becomes the main limiting factor, adding more VCs cannot provide any further performance improvements. Another limitation comes from the assumption of independent traffic flows; this may not be the case in real applications. Yet, our approach provides a good insight to network performance when the time dependencies of traffic flows result in a complicated analysis.

4.3 VC allocation algorithm

Our VC allocation algorithm is summarized in Fig.5. The VC configuration is initialized to one channel allocated to all links in the NoC; this is actually equivalent to having no VCs in the network. Then, using the probabilistic model described in the previous section (*i.e.* given the traffic configuration, mapping, routing strategy, and the available VC budget), we evaluate the expected switch bandwidth for all input ports. Based on this analysis, we then select the link l with the highest bandwidth usage to allocate one extra VC; this is shown in the highlighted block in Fig.5.

For the selected link, we do the following operations:

- We check first if the number of traffic flows across the link is equal to one. If so, we mark this link accordingly and no more VCs will be allocated to it. Consequently, the algorithm chooses the link with next largest bandwidth usage and the process is repeated.

- Second, we check if the number of VCs at the selected link is greater or equal than maximum number of VCs per link (W). Again, if the number of VCs already allocated to this link is equal to W, we mark this link as finished and search for the next candidate.

After a link is selected to allocate one extra VC, the VC configuration is updated and the VC budget is decreased by one. A new iteration begins only if the VC budget is not used up. When the greedy process stops, the algorithm outputs $g_{VC}(L)$ which maps all the links to the number of VCs.

For a network with n nodes, the complexity of computing the expected utilization of each link is $O(n\times|F|)$, where $|F|$ (defined in Table 1) is the total number of traffic flows between the processing elements. In the worst case, all n nodes communicates with each other as in uniform traffic, *i.e.* $|F|=n^2$. Consequently, the worst case complexity of the proposed algorithm is $O(n^3)$.

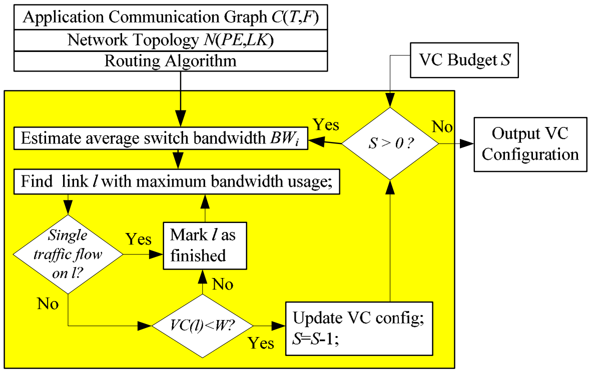

Application Communication Graph C(T,F)
Network Topology N(PE,LK)
Routing Algorithm
VC Budget S
Estimate average switch bandwidth BW_l
Yes
S > 0 ?
No
Output VC Configuration
Find link l with maximum bandwidth usage;
Single traffic flow on l?
Yes
Mark l as finished
No
No
VC(l)<W?
Yes
Update VC config; S=S-1;

Fig 5: Virtual Channel allocation algorithm.

5. Experimental results

The experimental setup is based on a cycle-accurate, flit-level C++ simulator developed to support wormhole routing with non-uniform VC configuration. During simulation, packets are generated by a Markov process with a given traffic rate at the source PEs. When blocking happens, the blocked packets wait in the buffer until the transmitting resources are available. The time it takes for a router to make routing decision is 3 clock cycles, and the bandwidth for each link is 1 flit/cycle.

5.1 Experiments using synthetic traffic

In the first set of experiments, we use *transpose* (Fig.6) and *hotspot* (Fig.7) traffic patterns on a 4×4 network.

As shown in figures 6 and 7, the average latency goes up dramatically after the packet generating rate exceeds a critical packet injection rate. When no VCs are implemented (*i.e.* "No VC" curve in Fig.6), the throughput for *transpose* is 0.57 (packets/cycle). After allocating 4 extra VCs non-uniformly with our algorithm (*i.e.* "Alloc 4 VC" in Fig.6), the throughput moves to 0.70, which represents about 22% improvement. Similarly, a significant improvement is obtained by allocating 6 extra VCs. For reference, we also show the latency when all links are allocated 2 VCs uniformly (*i.e.* the "Uniform 2 VC" curve in Fig.6).

In Fig.7 we show the effect of non-uniform VCs on *hotspot* traffic. When 1 extra channel is added ("Alloc 1 VC" in Fig.7), the throughput increases about 12.1%. The performance for 4 extra VCs ("Alloc 4 VC") and 10 extra VCs ("Alloc 10 VC") are also depicted in Fig.7. With 10 extra VCs, the network provides a similar performance improvement as for the case of uniformly allocated 2 VCs.

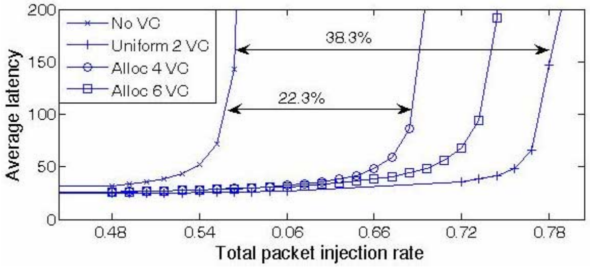


Fig 6: VC allocation for *transpose* on 4x4 mesh.

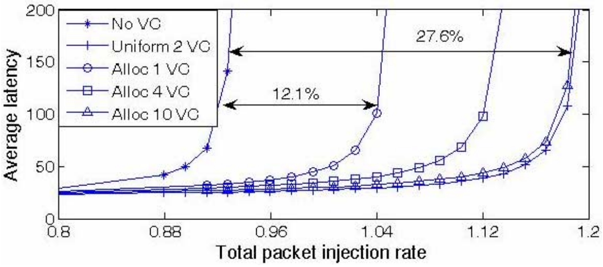


Fig 7: VC allocation for *hotspot* on 4x4 mesh.

In a 4×4 network without any VCs, total buffer space is 48m flits, if the buffer at each channel is m flits long (*i.e.* m flits/channel × 48 channels). Similarly, 96m flits are necessary for uniform 2 VCs; this implies 100% overhead compared to no VC case. For the *hotspot* traffic pattern, 12 VCs are able to provide a similar performance as the uniform allocation of 2 VCs. The overhead in buffer space for 12 extra VCs (*i.e.* 60m flit overall buffer space) is only 25% (*i.e.* 12m ÷ 48m). On the other hand, 37.5% (36m ÷ 96m) buffer space is saved compared to uniformly allocated 2 VCs for each channel.

5.2 Experiments using real traffic

We also evaluate the effect of VC allocation using benchmarks selected from the E3S suite [17]. Two benchmarks, namely *auto-industry* and *telecom*, are mapped onto a 4×4 and 5×5 mesh topologies, respectively. We performed experiments for different mappings, and obtained results similar to those shown in figures 8 and 9. Due to space limitations, we show only one experiment for each benchmark.

First, we simulate with uniformly allocated 1, 2, and 3 VCs across all links; these results are labeled as "No VC", "Uniform 2 VC", and "Uniform 3 VC", respectively, in figures 8 and 9. The results for our optimized VC allocation are labeled by "VC+". From figures 8 and 9, we can easily observe the improvement in the network throughput for different number of extra VCs. In the case of *auto-industry* benchmark, a throughput improvement of 25.7% and 29.4% is obtained with uniformly distributed 2 VCs and 3 VCs, respectively ("Uniform 2 VC" and "Uniform 3 VC" curves in Fig.8).

For the *telecom* benchmark, 35% improvement is observed with uniform 2 VCs ("Uniform 2 VC" in Fig.9). As we can see in figures 8 and 9, the improvement becomes smaller as the number of uniformly allocated VCs increases. Since the traffic characteristics are different in each example, the amount of improvement differs from case to case. As seen in figures 8 and 9 ("VC+" curves), the network throughput grows with the number of VCs added. When the appropriate number of

VCs is allocated, the approach is able to provide performance improvement similar to uniformly allocating VCs. The number of extra VCs needed to reach a similar improvement depends on the mapping and traffic characteristics.

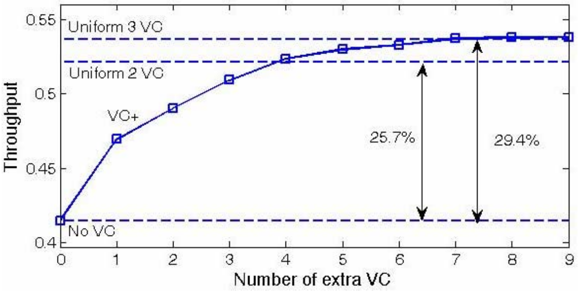


Fig 8: *auto-industry* 4x4 mesh.

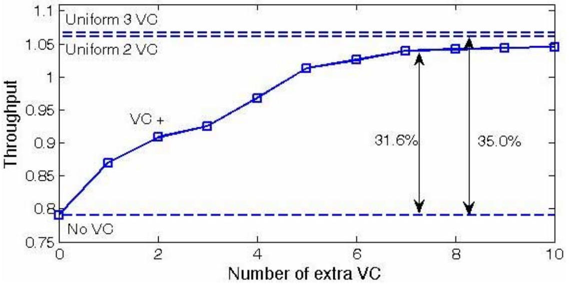


Fig 9: *telecom* 5x5 mesh.

With the proper number of VCs, we can efficiently increase the link utilization. However, if the traffic volume for a certain port is greater than the router's maximum service rate, the congestion cannot be mitigated by using VCFC. This is the reason for the saturation in performance improvement towards the end of the curves in figures 8 and 9. After we solve the HOL blocking with VCs, the limiting factor becomes the router service rate.

To further assess how close the proposed algorithm approaches the optimum solution, we simulated exhaustively all possible configuration by adding 3 VCs to 3x3 and 4x4 mesh networks. The saturation throughput of the network synthesized using the proposed algorithm is about 4% worse than the optimum solution.

As we did in previous section, we compare the buffer space requirements in Table 2. For the *auto-industry*, adding 5 VCs is sufficient to provide a similar performance to the uniform VC allocation (see Fig.8), although only 10% of extra buffering space is actually used (Table 2, "non-uniform +5 VC" column). Compared to 100% overhead in uniform allocation of 2 VCs, about 44.8% (*i.e.* (96*m* - 53*m*) ÷ 96*m*) of buffer space is saved. As for the *telecom* benchmark (Fig.9), 10 extra VCs can provide a satisfactory performance improvement with only 13% (Table 2, "non-uniform +10 VC" column) overhead in buffer space. Compared to the uniform allocation of 2 VCs, the savings in buffer space are about 43.75% ((160*m* - 90*m*) ÷ 160*m*). When we compare the results with the uniform allocation of 3 and 4 VCs, that require 200% and 300% overhead, respectively, the savings are even more significant.

Table 2: Comparison of buffer space requirements

Mesh	Uniform VC		Non-uniform VC			
size	1 VC	2 VC	+2	+3	+5	+10
4×4	48*m*	96*m*	50*m*	51*m*	53*m*	58*m*
	base	+100%	+4%	+6%	+10	+21
5×5	80*m*	160*m*	82*m*	83*m*	85*m*	90*m*
	base	+100%	+3%	+4%	+6%	+13

6. Conclusion and future work

In this paper, we have shown that customized VCs allocation can provide performance improvements similar to the uniform VC configuration for on-chip networks but with much less area overhead. The proposed approach offers designer a finer control over the tradeoffs between network performance and implementation cost. Future work will address the correlation among the traffic flows and dynamic variations in the network traffic.

Acknowledgements

This work is supported by Marco GSRC and NSF CCR-00-93104.

References

[1] W.J. Dally and B. Towles, "Route Packets, not Wires: On-chip Interconnection Networks," DAC, June, 2001.

[2] U. Y. Ogras, J. Hu, and R. Marculescu, "Key Research Problems in NoC Design: A Holistic Perspective," CODES+ISSS, Sep., 2005.

[3] J. Hu, U. Y. Ogras, R. Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Dec., 2006.

[4] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," IEEE Trans. on Communications, Dec., 1987.

[5] H. Wang, X. Zhu, L. Peh, and S. Malik, "Orion: A Power-Performance Simulator for Interconnection Networks," IEEE Symp. on Microarchitecture, Nov., 2002.

[6] N. Banerjee, P. Vellanki, and K. Chatha, "A Power and Performance Model for Network-on-chip Architectures," DATE, March, 2004.

[7] X. Chen and L. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks," ISLPED, Aug., 2003.

[8] W. J. Dally and C. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," IEEE Trans. on Computers, May, 1987.

[9] W. J. Dally, "Virtual-channel Flow Control," IEEE Trans. on Parallel and Distributed systems, March, 1992.

[10] K. Goosens, J. Dielissen, and A. Radulescu, "Aethereal Network on Chip: Concepts, Architectures, and Implementations," IEEE Design and Test of Computers, Sept-Oct, 2005.

[11] T. Bjerregaard and J. Sparso, "A Router Architecture for Connection-oriented Service Guarantees in the MANGO Clockless Network-on-chip," DATE, March, 2005.

[12] M. Rezazad and H. Sarbaziazad, "The Effect of Virtual Channel Organization on the Performance of Interconnection Networks," IEEE Symp. Inter. Parallel and Distributed Processing, April, 2005.

[13] R. Mullins, A. West, and S. Moore, "Low-Latency Virtual-Channel Routers for On-Chip Networks," IEEE Symp. on Computer Architecture, June, 2004.

[14] N. McKeown, "The iSLIP Scheduling Algorithm for Input-queued Switches," IEEE/ACM Trans. Networking, April, 1999.

[15] M. Pirvu, L. Bhuyan, and N. Ni, "The Impact of Link Arbitration on Switch Performance," International Symp. on High-performance Computer Architecture, Jan., 1999.

[16] U. Y. Ogras, R. Marculescu, "Prediction-based Flow Control for Network-on-Chip Traffic," ACM/IEEE DAC, July, 2006.

[17] R. Dick. Embedded system synthesis benchmarks suites (E3S), http://www.ece.northwestern.edu/~dickrp/e3s

[18] V. S. Adve and M. K. Vernon, "Performance Analysis of Mesh Interconnection Networks with Deterministic Routing," IEEE Trans. on Parallel and Distributed Sys., March, 1994.