A Shared Self-Compacting Buffer for Network-On-Chip Systems

Jin Liu and José G. Delgado-Frias
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
Email: {jinliu jdelgado}@eecs.wsu.edu

***Abstract*— In this paper we present a novel shared buffer scheme for systems on chip applications that require an interconnection network. The proposed scheme is based on a dynamically allocated multi queue self-compacting buffer. Two virtual channels shared the same buffer space. This in turn takes advantage of the available space. The proposed scheme outperforms existing approaches. In addition the scheme has similar performance using only half of the buffer size used in other traditional implementations.**

I. INTRODUCTION

As technology allows greater integration system-on-chip (SoC) designs are being investigated in greater detail. By the end of this decade SoC will have up to four billion transistors [1]. SoCs are anticipated to have a number of components (or modules) that will interact to compute a solution. These components will need to communicate to pass data and/or control information. Having dedicated connections between any given modules could be extremely complex as the number of modules increases. An alternative could be use an interconnection network within the chip.

Using an interconnection network on chip needs be restricted in terms of area due to the constraints of being in a single chip. Thus, it is extremely important to use schemes that require less hardware resources as well as provide a good performance. Virtual channel multiplexing across a physical channel is extensively used to boost performance and avoid deadlock. Each virtual channel is realized by a pair of buffers located on adjacent communicating nodes. As virtual channels are not equally used in many applications, if they share a common buffer, the whole buffer space will be better utilized. In this paper we present a novel scheme that is based on a dynamically allocated multiple queue (DAMQ) buffer. This scheme provides similar performance as other statically allocated multiple-queue buffers using less hardware and, therefore, requiring less hardware.

This paper has been organized as follows. In Section I, background information is provided. In Section II, the proposed buffer scheme is described. Performance evaluation results are reported in Section III. Some concluding remarks are included in Section IV.

II. BACKGROUND

Effective routing and buffer organization algorithms are two important design factors of a modern high performance wormhole switch for an interconnection network on chip. In this section we present a brief review of existing routing algorithms and DAMQ (Dynamically Allocated Multi-Queue) buffer schemes then describe our novel DAMQ buffer scheme. This new scheme is DAMQ with shared space for two physical channels ($DAMQ_{shared}$).

A. Existing routing algorithms

Routing algorithms are used to determine the messages path from source to destination. They can be implemented in two ways which are either deterministic or adaptive.

1) Deterministic routing protocols. Deterministic routing protocol chooses the path for a message only by its source and destination. All packets with the same source and destination pair will follow one single path. The packet will be delayed if any channel along this path is loaded with heavy traffic, and if a channel along this path is faulty the packet cannot be delivered. Thus the deterministic routing protocols are prone to suffer from poor use of bandwidth, blocking when alternative paths are available. They are particularly susceptible to component failures. [8] A common deterministic routing algorithm is dimension-order routing [9], in which the packet is routed in one dimension at a time, arriving at the proper coordinate in each dimension before proceeding to the next dimension.

2) Adaptive routing protocols. Adaptive routing protocols are proposed to make more efficient use of bandwidth and to improve fault tolerance of interconnection network. In order to achieve this, adaptive routing protocols provide alternative paths for communicating nodes. Thus it can overcome the congested areas in the network. Several adaptive routing algorithms have been proposed, showing that message blocking can be considerably reduced, thus strongly improving throughput. [10] Among them, routing algorithms based on Duato's design methodology [11] are very popular. These routing algorithms split each physical channel into two virtual channel sets, the adaptive and the deterministic channels. Packets are routed adaptively using any available adaptive channel or the deterministic channels

1-4244-0173-9/06/$20.00 ©2006 IEEE.

which form escape paths. When the paths of adaptive channels are blocked, a packet uses an escape channel at the congested node. If there is any free adaptive channel available at subsequent nodes, the packet can go back to the adaptive channels. As Duato's adaptive algorithms are widely used, we choose one of them to conduct our simulations.

B. Existing DAMQ buffer schemes

1) Linked list buffer scheme. In order to let multiple queues of packets share a DAMQ buffer, linked lists can be used to implement the buffer scheme [2, 3]. The basic idea of this approach is to maintain (k+1) linked lists in each buffer: one list of packets for each one of the (k-1) output ports, one list of packets for the end node interface and one list of free buffer blocks. Corresponding to each linked list there is a head register and a tail register. The head register points to the first block in the queue and the tail register points to the last block. In each output queue, next block information also must be stored in each buffer block to maintain the FIFO ordering.

2) Self-compacting buffer scheme. To reduce the hardware complexity of the linked list scheme, an efficient DAMQ buffer design self-compacting buffer (SCB) was brought by [4]. The idea for this buffer scheme is to divide the buffer dynamically into regions with every region containing the data associated with a single output channel. If two channels are denoted as i, j with $i < j$, then the addresses of buffer regions for the two channels A_i A_j will be $A_i < A_j$. There is no reserved space dedicated for any channel. Data is stored in a FIFO manner within the region for each channel. When an insertion of the packet requires space in the middle of the buffer, the required space will be created by moving down all the data which reside below the insertion address. Similarly, when a reading operation conducted from the top of a region, data removed from the buffer may result in empty space in the middle of the buffer, then the data below the read address is shifted up to fill the empty space.

3) DAMQ buffer schemes with reserved space for virtual channels scheme. A drawback of traditional DAMQ and Self-Compacting buffer schemes is when several virtual channels multiplexed across the physical channel and share a buffer, the virtual channels which have packets accepted in the buffer prior to other virtual channels may hold the whole buffer space when the output port to next hop it destines to is busy. In this case, other virtual channels can no longer take flits of other messages that may have free next hop. To overcome this problem, two schemes based on self-compacting buffer were introduced in [12]. They reserve buffer space for the virtual channels on the same physical channel; flits heading to one virtual channel can never occupy the whole shared buffer space.

III. $DAMQ_{SHARED}$ BUFFER SCHEME

DAMQ allocates buffer space when a packet is received. Compared with statically allocated multi-queue (SAMQ) scheme, the advantage of DAMQ is its efficient use of the buffer space by allocating free space to an incoming packet regardless its destination output port. However, because there is no reserved space dedicated for each output channel, the packets destined to one specific output port may occupy the whole buffer space thus the packets destined to other output ports have no chance to get into the buffer. This is the case especially for small and compact routers with limited buffer space where wormhole switching technique and virtual channel mechanism are used. In order to overcome this shortcoming a new buffer organization scheme, DAMQ with reserved space for all virtual channels ($DAMQ_{all}$) was proposed in [12]. $DAMQ_{all}$ is based on Self-compacting buffer (SCB) scheme.

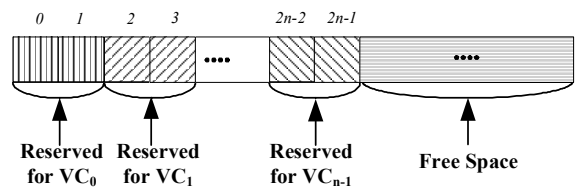

Figure 1. $DAMQ_{all}$ Buffer space at the initial state.

The virtual channels belonging to one direction of a bidirectional physical channel share a buffer. The new feature is that there is reserved space dedicated for each virtual channel, therefore at any time free space is available for the packets of "late" virtual channels which has not received packet. As shown in Fig 1, two buffer slots are reserved for each virtual channel before the buffer accepts any incoming flit. The reserved space for each virtual channel is always kept if there is no flit or only one flit in the buffer region for a specific virtual channel as shown in Fig 2.

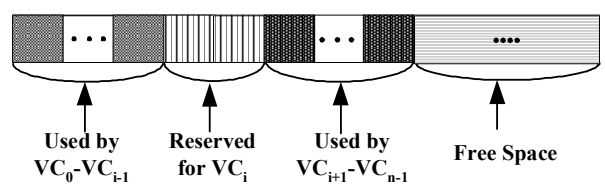

Figure 2. $DAMQ_{all}$ Buffer space status in operations.

However, in a wormhole-switched network with several virtual channels multiplexing across a physical channel, routing algorithms tend to choose one set of virtual channels over others; thus, some virtual channels are more intensively used than others. The traffic load is not evenly distributed in the entire buffer space for a physical channel. A more efficient approach to use the available buffer space is to let the virtual channels belonging to a physical channel share buffer with virtual channels of another physical channel. The DAMQ scheme is used in the allocation of space.

As shown in Fig 3. (a), the simple switch with four input and four output ports adopts $DAMQ_{all}$ buffer scheme for the input buffer; there is one dedicated buffer per physical channel, i.e. east X, west X, north Y and south Y. Each buffer has its own read port and write port, the four virtual

channels that are belonging to a physical channel have there own reserved space in the buffer for that physical channel.

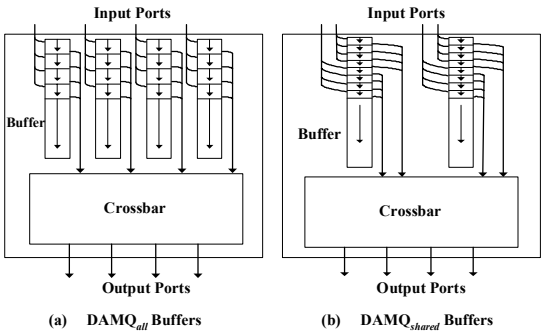

Figure 3. Switches with DAMQ$_{all}$ buffer and DAMQ$_{shared}$ buffer

Our new DAMQ$_{shared}$ buffer design combines the buffer space for virtual channels from different physical channels. Based on the statistics of virtual channel usage that we obtained from our simulations, we concluded that the pairs with better performance are east X and south Y virtual channels, and the buffer space for west X and north Y likewise. By doing that, one set of X dimension virtual channel share buffer with one set of virtual channels from Y dimension. As shown in Fig 3. (b), there are two buffers for four physical channels; each buffer is shared by eight virtual channels, and has two read ports and write ports respectively. The new buffer behaves similarly as DAMQ$_{all}$. The number of virtual channels sharing a buffer is doubled in this case.

As shown in Fig 4, two buffer slots are reserved for each virtual channel before any flit comes in the buffer. The reserved spaces for each virtual channel are arranged sequentially according to the sequence numbers of the virtual channels. Virtual channels on the X dimension have smaller sequence numbers than those on the Y dimension; the first virtual channel on Y dimension is contiguous to the last one on X dimension. One register is used to point to the head of each reserved space, i.e. the head of the buffer region for each virtual channel. If two channels are denoted as V_i, V_j with $i + 1 = j$, then the reserved region for V_j will be placed right after the reserved region for V_i. The reserved space for virtual channels on Y dimension is right after the reserved space for X virtual channels. The size of reserved space for each virtual channel can be adjusted, we choose two flits is because in our simulation experiments, two reserved flits scheme yields satisfactory performance while keeps more free space for sharing.

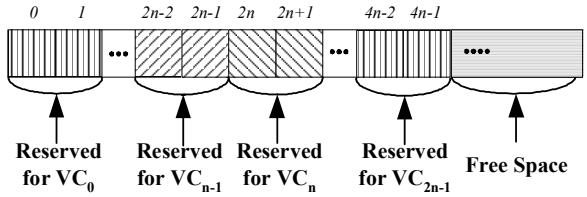

Figure 4. DAMQ$_{shared}$ Buffer space status at initial state.

The reserved space for each virtual channel is always kept if there is no flit or only one flit in the buffer region for a specific virtual channel.

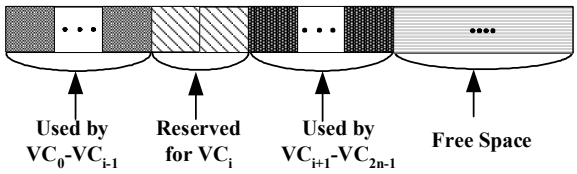

Figure 5. DAMQ$_{shared}$ Buffer space status in operations.

As shown in Fig 5, when the buffer performs shift up or shift down operations, the reserved spaces are treated same as the slots which are holding flits. Thus the order of the buffer space for virtual channels is kept conforming to the sequence of virtual channels. Once the number of current flits in buffer plus the number of reserved slots equals to the total amount of buffer slots, no more flit will be accepted unless this flit goes to a virtual channel which has any reserved space available. Therefore, one or more virtual channels which have the flits come into the buffer at earlier time can never deprive the chance for other virtual channels which get flits later than them to get buffer. Also, once the earlier coming packets are blocked in the buffer, since there is still reserved space for other virtual channels, the network traffic will keep flowing, so the performance of the switch is also enhanced. Moreover, as virtual channels from two physical channels are sharing the buffer, the buffer space is more efficiently used by the incoming flits.

A VLSI implementation of the self compacting buffer (SCB) has been reported in [7]. This design needs some modifications to be used as shared buffer. Fig 6 shows an implementation of the shared SCB. It should be pointed out that the buffer is used at both ends by different physical channel the channel pointers keep track pf the virtual channels (V.C.) that are used. Since the original SCB [7] has capabilities for shifting up and down, this feature can be used in the new scheme. Thus, the amount of hardware that is needed to implement the new scheme is very similar than having two independent SCBs.

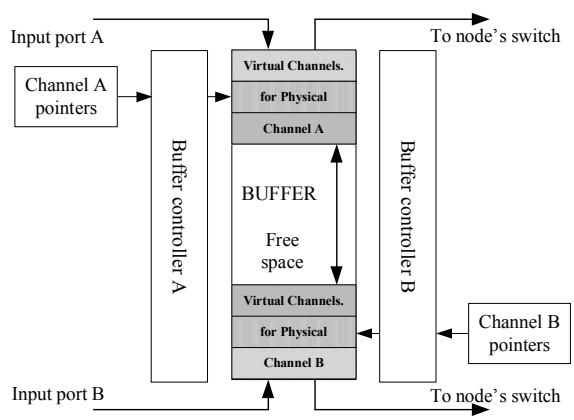

Figure 6. Switches with DAMQ$_{all}$ buffer and DAMQ$_{shared}$ buffer

IV. PERFORMANCE EVALUATION

In this section, we describe the results of simulation experiments conducted to evaluate the performance of the proposed buffer organization scheme. Firstly we describe our methodology and configuration of simulation environment. Then we examine the performance of $DAMQ_{shared}$ in greater detail.

A. Simulation experiments

We have carried out our simulations using flexsim1.2 [6] which is a simulator for flit-level simulation of torus/mesh networks. The architecture we simulate is an 8-ary 2-cube message exchanging system with wrapped around channels as shown in Fig 7.

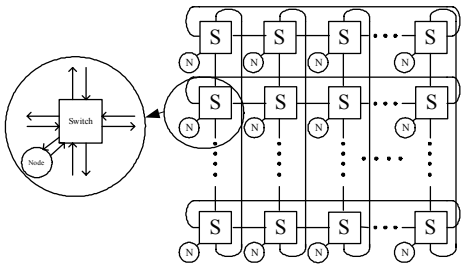

Figure 7. The base system for our simulations

Each switch is attached to one end-node which has one injection channel to the switch and one input channel to receive message from network. Physical channels are duplex channel. Network traffic is uniformly distributed. Every end node generates packets with randomly determined destination and injects them into the network. Other pertinent simulation configuration parameters are listed as follows:

- Routing flit delay is set to 1 cycle.
- Data flit delay is set to 1 cycle.
- Buffers at local end nodes are infinite.
- Packets size is set to 32 flits.
- Virtual channels number for each physical channel is 4.
- Switching technique used is wormhole.

We use adaptive routing protocol to conduct our simulations as several researchers had reported the strong performance brought by adaptive routing [10]. We choose Duato's routing methodology in our experiments because it is a well known and used adaptive routing protocol. We use dimension order path selection function for the Duato's routing protocol. We vary the applied load, buffer size; study their impact on throughput and message latency for the network. Since messages are divided to flits when transmitting, to increase message length has the similar effect on increasing traffic load as to shorten the average injection period for each node. We set the message length at 32 flits and make the network into saturation state by shortening injection period for each node namely by increasing the traffic load rate. Traffic load rate is derived from the following formula [6]:

$$LR \times Ch = N \times (ML / IP) \times Dst$$

Where *LR* is load rate, *Ch* is num. of channels, *N* is num. of nodes, *ML* is message length, *IP* is average injection period, and *Dst* is average routing distance.

The throughput is defined as the number of flits received per node per cycle. The message latency is measured as the average time span for every packet between the moment it was generated and the reception of the whole packet at destination.

B. Simulation results

To increase the number of virtual channels multiplexing a physical channel can improve switch performance, but having too many virtual channels not only incurs expensive hardware expense but also increases the message delay. In our simulation experiments, we set the number of virtual channels of one direction of a duplex physical channel to four, thus the advantages brought by virtual channel mechanism will not be overshadowed by its shortcoming. We evaluate the performance of the switch in terms of the throughput and the message latency which are two most important performance metrics for an interconnection network. We compare our new buffer scheme $DAMQ_{shared}$ to $DAMQ_{all}$ and the traditional statically allocated buffer scheme (SAMQ) used for virtual channels which is reported in [5].

Because of the hardware constrains for network on chip, we examine the network performances for 4 and 8 flits buffer for each virtual channel on SAMQ; fixed 4 flits buffer for each virtual channel on $DAMQ_{all}$. Three different size buffers, 4, 6 and 8 flits buffers for each virtual channel are used on $DAMQ_{shared}$ so that we can examine its performance in detail. The simulation results for throughput and message latency are shown in Table I and II, respectively. It is well known that there is no significant difference for different buffer scheme while the network is not saturated. In our simulation experiments, we keep increasing the traffic load so that we can compare the performance of different buffer schemes until the network is saturated after about 0.4 traffic load is applied. As shown in Fig 8, along with the network saturation process, our $DAMQ_{shared}$ has significant higher throughputs than both $DAMQ_{all}$ and SAMQ when they all use same size of 4 flits buffer and $DAMQ_{all}$ beats SAMQ. $DAMQ_{shared}$ not only achieves the highest throughput when same size buffer is used, but also achieves approximately the same maximum throughput as SAMQ using 8 flits buffer, which double the number of buffer $DAMQ_{shared}$ is used. When we use 6 flits buffer for $DAMQ_{shared}$, it achieves a significantly higher throughput than SAMQ with 8 flits buffer as shown in Fig 8. The max throughput is achieved by $DAMQ_{shared}$, when 8 flits buffer is used for it. In sum, $DAMQ_{shared}$ achieves best performance among the three buffer schemes we tested. The reason, we believe, is $DAMQ_{shared}$ provides more efficient methods for flits to share buffer space than $DAMQ_{all}$ which has already shown advantages over traditional SAMQ scheme. Furthermore, at the same maximum throughput level, $DAMQ_{shared}$ also has a similar latency as SAMQ as shown in Fig 9.

TABLE I. THE PERFORMANCE 8-ARY, 2-CUBE NETWORK COMPOSED OF BLOCK SWITCHES. SHOWN IS THE THROUPUT OBTAINED FROM SIMULATIONS WHERE 4 VIRTUAL CHANNELS USED.

Buffer Type	BS per VC	Applied Traffic Load Rate						
		0.20	0.23	0.29	0.31	0.33	0.35	0.38
SAMQ	4	0.39	0.44	0.53	0.54	0.55	0.55	0.55
	8	0.39	0.44	0.54	0.56	0.58	0.59	0.59
DAMQ $_{all}$	4	0.39	0.44	0.54	0.56	0.57	0.57	0.57
DAMQ$_{shared}$	4	0.39	0.44	0.54	0.57	0.58	0.59	0.59
	6	0.39	0.44	0.55	0.58	0.6	0.61	0.61
	8	0.39	0.44	0.55	0.58	0.6	0.62	0.62

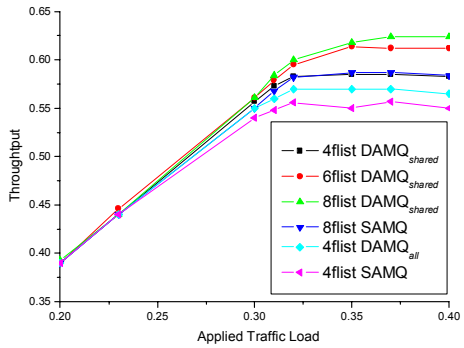

Figure 8. Comparison on throughput between 4/6/8 flit-buffer DAMQ$_{shared}$, 4 flit-buffer DAMQ$_{all}$ and 4/ 8 flit-buffer SAMQ

TABLE II. THE PERFORMANCE 8-ARY, 2-CUBE NETWORK COMPOSED OF BLOCK SWITCHES. SHOWN IS THE MESSAGE LATENCY OBTAINED FROM SIMULATIONS WHERE 4 VIRTUAL CHANNELS USED.

Buffer Type	BS per VC	Applied Traffic Load Rate						
		0.20	0.23	0.29	0.31	0.33	0.35	0.38
SAMQ	4	102	113	140	156	167	174	186
	8	102	112	141	157	175	183	204
DAMQ $_{all}$	4	102	110	142	157	175	184	198
DAMQ$_{shared}$	4	102	113	142	158	178	187	207
	6	101	110	140	157	169	188	216
	8	102	110	139	155	169	185	225

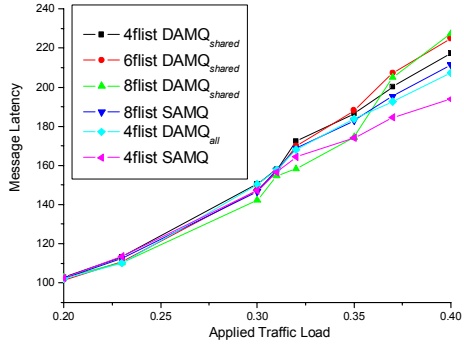

Figure 9. Comparison on Latency between 4/6/8 flit-buffer DAMQ$_{shared}$, 4 flit-buffer DAMQ$_{all}$ and 4/8 flit-buffer SAMQ

V. CONCLUDING REMARKS

In this paper we have presented a novel buffer scheme based on a DAMQ self-compacting buffer. This scheme outperforms existing approaches. In addition the proposed scheme has similar performance using only half of the buffer size used in SAMQ when an adaptive routing protocol Duato's algorithm is used. DAMQ$_{shared}$ provides an excellent scheme to optimize buffer management providing a good throughput when the network has a larger load. Implementing the proposed scheme in hardware would require minor modifications to early implementations of the self-compacting buffer [7].

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, Vol. 35 , No. 1, pp. 70-78, Jan. 2002

[2] R. Sivaram, C. B. Stunkel, and D. K. Panda, "HIPIQS: A High Performance switch architecture using input queueing," *IPPS/SPDP '98*, pp. 134–143, Orlando, FL, March 1998.

[3] Y. Tamir and G. L. Frazier, "Dynamically-allocated multiqueue buffers for VLSI communication switches," *IEEE Transactions on Computers,* vol. 41, no. 2, pp. 725–737, June 1992.

[4] J. Park, B. O'Krafka, S. Vassiliadis, and J. Delgado-Frias, "Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers," *IEEE Supercomputing '94, The Conference on High Performance Computing and Communications,* pp. 713–722,Nov.14-18 1994

[5] W. Dally. "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems,* vol. 3, no. 2, pp. 194–205, 1992.

[6] S. Warnakulasuriya and T.M. Pinkston, "Characterization of Deadlocks in k-ary n-cube Networks," IEEE Trans. Parallel and Distributed Systems, vol. 10, no 9, pp. 904-932, Sept. 1999.

[7] J. G. Delgado-Frias and R. Diaz, "A VLSI Self-Compacting Buffer for DAMQ Communication Switches," *IEEE Eighth Great Lakes Symposium on VLSI,* pp. 128-133, Lafayette, Louisiana, February 1998.

[8] Patrick T. Gaughan and Sudhakar Yalamanchili, "Adaptive Routing Protocols for HvDercub Interconnection Networks," *IEEE Trans. Computer*, vol. 26, no 5, pp. 12-23, May 1993.

[9] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channnels," *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, No. 4, pp. 466-475, April 1993

[10] E. Baydal, P. L´opez and J. Duato, "Increasing the Adaptivity of Routing Algorithms for k-ary n-cubes," in Proc. 10th Euromicro Workshop on Distributed and Network-based Processing, pp. 455-462, Jan. 2002

[11] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," IEEE Trans. Parallel Distributed Syst., vol. 4, no. 12, pp.1320–1331, Dec. 1993.

[12] J. Liu, J. G. Delgado-Frias, "DAMQ Self-Compacting Buffer Schemes for Systems with Network-On-Chip," *In Proc*, 2005 Int. Conf. Comput Design, pp. 97-103, Las Vegas, June 2005.