In Proceedings of the 6th International Symposium on High-Performance Computer Architecture, Toulouse, France, January 10-12, 2000. pp. 73-84.

Flit-Reservation Flow Control

Li-Shiuan Peh
lspeh@cs.stanford.edu

William J. Dally
billd@csl.stanford.edu

Computer Systems Laboratory
Stanford University
Stanford, CA94305

Abstract

This paper presents flit-reservation flow control, in which control flits traverse the network in advance of data flits, reserving buffers and channel bandwidth. Flit-reservation flow control requires control flits to precede data flits, which can be realized through fast on-chip control wires or the pipelining of control flits one or more cycles ahead of data flits. Scheduling ahead of data arrival enables buffers to be held only during actual buffer usage, unlike existing flow control methods. It also eliminates data latency due to routing and arbitration decisions. Simulations with fast control wires show that flit-reservation flow control extends the 63% throughput attained by virtual-channel flow control with 8 flit buffers per input to 77%, an improvement of 20% with equal storage and bandwidth overheads. Its throughput with 6 buffers (77%) approaches that of virtual-channel flow control using 16 buffers (80%), reflecting the significant buffer savings as a result of efficient buffer utilization. Data latency is also reduced by 15.6% as compared to virtual-channel flow control. The improvement in throughput is similarly realized by the pipelining of each control flit a cycle ahead of their data flits, using control and data networks with the same propagation delay of 1 cycle.

1. Introduction

Communication is the limiting factor in many computer systems, network switches and routers, and other digital systems. Buses, which have long been used for communication in such systems, are unable to keep up with their rapidly growing demand for high communication performance. Interconnection networks, historically used in high-end parallel computers, have begun to fill this performance gap, replacing buses in all types of computer systems and in high-end network switches. At the same time, chips have grown in complexity to the point that interconnection networks are beginning to be used for on-chip communication in addition to system-level communication.

Given a particular topology and routing strategy, the efficiency of an interconnection network is largely determined by its flow-control: the method used to allocate resources (buffer space and channel bandwidth) to the flits[1] of packets traversing the network. The shape of the characteristic latency-throughput curve of a network is determined by flow control. A good flow control strategy enables a network to operate at 80% or more of bisection bandwidth with low latency. A poor flow-control strategy, on the other hand, results in a network that saturates at less than 30% of capacity.

This paper introduces flit-reservation flow control in which control flits traverse the network ahead of data flits reserving buffers and channel bandwidth. When the data flits arrive, they are forwarded or buffered according to the reservation. This advance scheduling makes very efficient use of buffers, allowing them to be reused immediately following the departure of a flit. In contrast, the on-the-fly scheduling of existing flow-control methods idles each buffer for a considerable period after each flit departure.

Reserving network resources ahead of the arrival of data flits yields two performance improvements. First, data latency is reduced (by 15.6% for our example on-chip network) because routing and arbitration are performed in advance of data arrival. Second, for a fixed amount of buffer space, saturation throughput is increased because of the more efficient buffer scheduling. For an 8x8 mesh network with eight flit buffers per input, virtual-channel flow control [Dally92] saturates at 63% of bisection bandwidth while flit-reservation flow control extends performance to 77% of available bandwidth, a 20% improvement with equal storage and bandwidth overheads.

Our investigation of flit-reservation flow control is motivated by the design of an on-chip network suitable for use with emerging VLSI technology [DalLac99]. As VLSI technology scales, on-chip wires become slower - giving a wire delay of four clocks per hop in our example network. However, wires on a thick upper metal layer can have a fraction of this delay[2]. Flit-reservation flow control

1. A flow control digit or flit is the smallest unit of flow control.

exploits these fast wires by using them to send relatively small control flits ahead of the wide data flits to make reservations.

Flit-reservation flow control can also be applied to off-chip networks, where there are no fast control wires, by transmitting the control flits of a packet one or more cycles in advance of the data flits. This can be done without delay in many contexts. In a multiprocessor, for example, the control flits for a read reply packet can be sent while the DRAM is being accessed for the data flits. If the destination is not available in advance, the data flits can simply be delayed by a few cycles. This delay is largely recouped by the savings in routing and arbitration latency.

Current flow-control methods are reviewed and compared to flit-reservation flow control in Section 2. Section 3 delves into the details of flit-reservation flow control, explaining how reservations are made and buffers are allocated. Experimental results comparing flit-reservation flow control with virtual-channel flow control are presented in Section 4. Section 5 discusses several design and implementation issues pertaining to flit-reservation flow control and concluding remarks round up the paper in Section 6.

2. Related Work

With store-and-forward flow control[3], each node waits until an entire packet has been received before forwarding any of the packet to the next node. With this approach, a flit is an entire packet. Both channel bandwidth and buffer space are allocated in packet-sized units. With virtual-cut-through flow control [KerKle79], transmission of the packet may begin before the entire packet is received. However, bandwidth and storage are still allocated in packet-sized units. Store-and-forward flow control has long been used by computer networks [Tanenb96] and was adopted by early parallel computers such as the Caltech Cosmic Cube [Seitz85].

Wormhole flow control[4] [DalSei86] improves upon store-and-forward flow control by allocating storage and bandwidth to flits that are smaller than a packet. This allows relatively small flit-buffers to be used in each router, even for large packet sizes. This enabled fast single-chip routers to be built in 1986 technology. For a fixed amount of storage, this also results in lower latency and greater throughput because some flits of a packet are accepted before an entire packet buffer is available.

Wormhole flow control makes inefficient use of channel bandwidth however. While it allocates storage and bandwidth in flit-sized units, wormhole flow control holds a physical channel for the duration of a packet. As a result, when a packet is blocked, all of the physical channels held by that packet are left idle. Other packets queued behind the blocked packet are unable to use the idle physical channel. Hence throughput suffers.

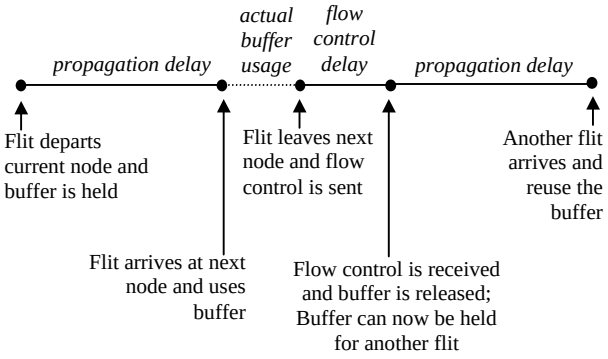


Figure 1. Timeline illustrating the buffer turnaround time of wormhole and virtual-channel flow control. Buffers are held unnecessarily throughout the propagation and flow control delays.

Virtual-channel flow control [Dally92] allows blocked packets to be passed by other packets. This is accomplished by associating several virtual channels, each with a separate flit queue, with each physical channel. Virtual channels arbitrate for physical channel bandwidth on a flit-by-flit basis. When a packet holding a virtual channel gets blocked, other packets can still traverse the physical channel through other virtual channels.

Both wormhole and virtual-channel flow control leave buffers idle between flits, reducing throughput and increasing buffer requirements. Figure 1 shows a timeline of the buffer allocation process used by wormhole and virtual-channel flow control. A buffer is held from the time the data flit departs the current node, to the time the flow control signal returns to inform the current node that the buffer has been released. Only then is the buffer allocated to the next flit. Hence, the turnaround time of a buffer, the idle time between flits, is at least the sum of the propagation delay of a data flit to the next node and the flow control delay back. For flit-reservation flow control, the turnaround time can be zero.

Wave switching [DLSY96] is a hybrid of circuit switching and wormhole flow control, where probes traverse a separate control network, setting up circuits on the data network for subsequent messages. Since the physical data channels are only used by a single message, there is no need for any buffering or flow control for the data flits. As a result, latency and throughput of the data network is enhanced. Like flit-reservation flow control, wave switch-

2. Doubling the width, spacing and thickness of wires doubles their velocity, halving both latency and bandwidth.
3. Sometimes called store-and-forward switching.
4. Sometimes called wormhole routing.

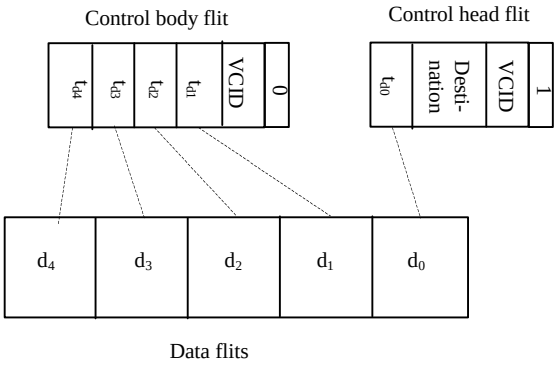


Figure 2. The control head flit contains the packet destination and is responsible for leading the first data flit. It identifies the first data flit by its arrival time (t_{d0}). Subsequent control body flits contain the arrival times of up to N data flits (In this figure, a control flit leads up to 4 data flits, thus N=4). All control flits contain a virtual channel identifier (VCID) which ties control flits of the same packet together.

ing uses a separate control network to reduce latency and extend throughput. While flit-reservation flow control achieves this through advance scheduling, wave switching attains it through removing the encumbrances of flow control mechanisms from the data network and thus trimming the clock cycle. Being governed by circuit switching, however, results in such gains only realizable if the circuit setup time can be amortized over many message deliveries.

Another technique which can similarly extend the latency-throughput envelope is statically-scheduled flow control, such as that used by the MIT Raw Machine [BLAA99], where a compiler schedules the allocation of buffers and channel bandwidth in advance of program execution. This has the advantage of eliminating the routing and arbitration latency and cutting the minimum buffer turnaround time to zero. However, it sacrifices flexibility for this performance. Such an approach cannot support applications with data-dependent communication. This prompted the Raw Machine to also include a separate dynamically-routed network as a fall-back mechanism for applications with dynamic communication patterns. A statically-scheduled network also precludes the use of run-time data migration to improve performance and load balance on an architecture.

Flit-reservation flow control offers many of the advantages of statically-scheduled flow control, while supporting the flexibility of a dynamically routed network. Like the statically-scheduled network, a network using flit-reservation flow control schedules the buffers and channels of the network in advance of the data packet arrival. However, instead of scheduling these resources at compile time, the scheduling decisions are deferred until data packet injection, allowing the schedule to adapt to data-dependent destinations, routes, and packet sizes. With both approaches buffers are used efficiently, with immediate turnaround; and routing and arbitration latency is hidden. With flit-reservation flow control, however, this performance is achieved without sacrificing flexibility.

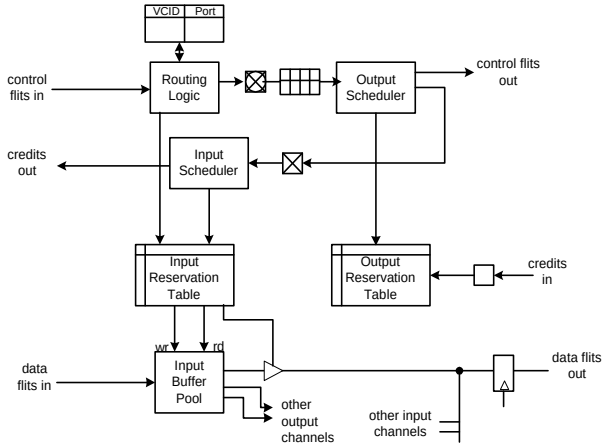


Figure 3. Block diagram of a router governed by flit-reservation flow control.

3. Flit-Reservation Flow Control

Traditionally, a packet consists of a head flit which contains the packet destination and leads subsequent body flits through the network. In a network using flit-reservation flow control, the responsibility of leading data flits falls on the shoulders of control flits. Control flits traverse a separate control network in advance of the data flits and schedule the data network buffer and channel resources for each data flit. When the data flits arrive, they are buffered and forwarded according to this pre-arranged schedule.

As shown in Figure 2, a packet consists of one or more control flits and zero or more data flits. The control head flit contains the packet destination and identifies the first data flit (if any) by specifying its arrival time. Each subsequent control flit specifies the arrival times of up to N additional data flits, where N is the maximum number of data flits led by a control flit. The data flits themselves contain only payload information. They are identified solely by their time of arrival.

A block diagram of a flit-reservation router is shown in Figure 3. For clarity, the figure shows just one input channel and one output channel. The upper part of the figure depicts the control network, while the lower part of the figure shows the data network.

When a control flit arrives at a router it proceeds through the steps of routing, output scheduling, and input

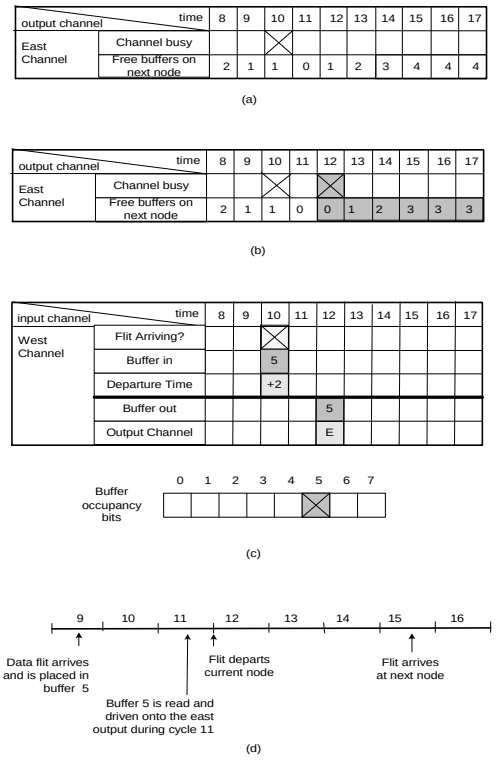


Figure 4. (a) The output reservation table before a control flit schedules a data flit arriving from the West channel at cycle 9 and leaving via the East channel. Since the channel is occupied at cycle 10, the next slot, cycle 11, is checked. The channel is free at cycle 11, but there aren't any free buffers left on the next node. Thus, cycle 11 is also invalid. The data flit is finally scheduled to leave at cycle 12. (b) Here, the output reservation table has been updated to reflect the departure of the data flit at cycle 12. The channel is marked busy; and the number of free buffers on the next node is decremented from cycle 12 onwards. (c) In the input reservation table, buffer 5 is allocated for the data flit when it arrives at cycle 9 and linked to the East output channel at cycle 12. Buffer 5 is also marked as occupied after it is allocated. The shaded regions reflect the updates. (d) Timeline showing the scheduled movements of the data flit.

scheduling. First, the routing logic block uses the destination field of the control head flit to determine the output port to which this packet should be forwarded. This output port selection is stored in a table indexed by virtual channel identifier (VCID). Subsequent control body flits look up their output port in this table using their VCIDs. In parallel with determining the output port selector, the arrival times of each data flit associated with a control flit are recorded in the input reservation table. After completing the routing step, the control flit, annotated with the input port number is forwarded to the output scheduler for the selected output port.

The output scheduler schedules the departures of each data flit associated with the control flit. Its goal is to schedule the data flits to leave as soon as possible for the next hop. This scheduling is done using the output reservation table as shown in Figure 4(a). The output reservation table records the status of each output channel (busy or not) and the number of free flit buffers at the far end of the channel. This status is kept for each clock cycle within a window of time from the present to a scheduling horizon (currently 32 cycles in the future), with circular reuse as time expires.

For a data flit arriving at time t_a, the output scheduler determines the earliest departure time, $t_d > t_a$, when the channel is not busy and there is at least one buffer available at the far end of the line to hold the data flit when it arrives at t_d + propagation delay, t_p. Hence, for all $t >= t_d + t_p$, a buffer must be available on the next hop. The scheduler then reserves these resources by marking the channel busy during cycle t_d and decrementing the buffer count for all $t >= t_d + t_p$.

A scheduling example is shown in Figure 4[5]. The state of the output reservation table before the flit is scheduled is shown in Figure 4(a). Here a data flit is scheduled to arrive at $t_a = 9$ (ten cycles in the future). The output channel is busy during cycle 10 and there are no buffers available during cycle 11[6]. Thus, the first available departure time is $t_d = 12$. Once this assignment is made, the output reservation table is updated as shown by the shaded areas in Figure 4(b). The channel is marked busy during cycle 12 and the buffer count is decremented from cycle 12 to the scheduling horizon.

For each successfully scheduled data flit, its corresponding arrival time field in the control flit is updated with the scheduled arrival time ($t_d + t_p$). At the same time, reservation signals containing t_a and t_d are sent back to the input scheduler associated with the originating input port to orchestrate movements of the data flit. When output scheduling is completed for all associated data flits, the control flit is forwarded over the control output link to the next node.

While the output scheduler reserves the departure times of data flits, the input scheduler orchestrates the movement of a data flit through the router at the scheduled time. When the input scheduler receives reservation signals from the output scheduler, it updates the input reservation table to reflect the flit departure and sends a credit back to the previous node to update the buffer availability in that node's output reservation table. The input reservation table, shown in Figure 4(c), contains a "Buffer In" row specify-

5. For simplicity, the buffer state at t_d is shown to be the buffer state at $t_d + t_p$ throughout this example.
6. Even if the channel were not busy during cycle 10, the flit could not be scheduled that cycle due to the shortage of buffers at cycle 11.

ing the buffer (if any) to which the input port should write each cycle. The table also has a "Buffer Out" row and a "Output Channel" row which indicates which buffer should be connected to which output port on a particular cycle[7].

When a reservation signal arrives at the input scheduling block, the input scheduler updates the table to reflect the flit's departure, by filling the "Departure Time" and "Output Channel" fields. Although this data flit is guaranteed to have a free buffer awaiting its arrival, a specific buffer is not allocated until one cycle before its arrival[8]. Thus at input scheduling time, "Buffer In" and "Buffer Out" fields are filled with a placeholder symbol that encodes the departure time. This is replaced by the actual buffer number just before the arrival of the data flit. At this point, the buffer occupancy bits are consulted, a free buffer is allocated, and the "Buffer In" and "Buffer Out" fields are updated with this buffer number. The buffer logic also detects the situation when a flit is scheduled to depart once it arrives and bypasses the flit directly to the output port.

Continuing our example, upon receiving reservation signals t_a=9 and t_d=12, the input scheduler updates the departure time of the flit and links the flit to the East output channel at time 12. These assignments are shown lightly shaded in Figure 4(c). The input scheduler also sends a credit back to the previous node, causing it to increment its free buffer count from cycle 12 onwards. One cycle before the arrival of the data flit in cycle 9, indicated by the "Flit Arriving?" field, buffer 5 is allocated and the "Buffer In" and "Buffer Out" entries are updated, shown in a darker shade in Figure 4(c). Buffer 5 is also marked as occupied in the buffer occupancy bits.

After a data flit has been scheduled, all its movements are tracked in the input reservation table. Each cycle, the table directs which buffer is to be written with data from the input link, and which buffers are to be driven onto which output links. There are no decisions to be made as all of the work has been done ahead of time by the control flits.

When the data flit scheduled in our example above arrives on the data input on cycle 9, the input reservation table directs it to be written to buffer 5. Then, during the cycle before its scheduled departure time (cycle 11), the table directs the flit to be read from buffer 5 and driven onto the east output. The data flit thus leaves for the next node at cycle 12, and arrives there during cycle 15. The contents of the data flit are never examined. It is stored and forwarded according to the pre-arranged schedule in the input reservation table. If a data flit arrives at a node before its control flit has completed its schedule, it is directed to a free buffer in the buffer pool and kept track of through a logical schedule list.

Control flits schedule the injection of data flits into the network in exactly the same manner described above, with control flits being injected only after they have scheduled the injection times of their data flits. Similarly, control flits schedule the re-assembly of data flits into packets at the destination, since data flits are potentially out of order. The data flits are assigned to sequential re-assembly buffers and the input reservation table schedules the transfer of each data flit from the input link into a re-assembly buffer.

In the absence of contention, a data flit can easily depart the router the cycle after it arrives. This is because all of the decision logic: the routing and arbitration, has been performed in advance by the control flit. This pre-arranged control is responsible for the low data latency of flit-reservation flow control.

Unlike existing flow control methods which hold a buffer from the time the data flit departs from the current node (t_d), flit-reservation flow control marks a buffer as occupied only from the time the data flit arrives at the next node ($t_d + t_p$). As control flits race ahead in flit-reservation flow control, credits are returned in advance and buffers freed in time for other control flits to reserve and use them immediately. This fast recycling of flit buffers accounts for the high throughput and significant buffer savings of flit-reservation flow control.

4. Experimental Results

We simulated flit-reservation and virtual-channel flow control on a flit-level simulator modeling an on-chip 2-dimensional 8-by-8 mesh interconnect. Control and credit signals are routed on wires that are 4 times faster than the data wires[9]. Thus, traversing the control and credit links between nodes takes 1 cycle, while traversing the data link takes 4 cycles. Routing and scheduling latency is 1 cycle. The simulated network uses random arbitration and deterministic dimension-ordered routing.

The simulator generates uniformly distributed traffic across the network to random destinations. Each simulation is run for a warm-up phase of a minimum of 10,000 cycles till average queue lengths have stabilized. Thereafter, 100,000 packets are injected and the simulation is run

7. A higher performance router can be realized by using a multi-ported input buffer addressed by multiple "Buffer Out" rows in the input reservation table. This enables a single input buffer to simultaneously forward data flits to multiple outputs.
8. This circumvents the buffer interchange problem which is discussed in Section 5.
9. The wires of the control network are 4 times thicker than that on the data network. The control network hence has 1/4 of the bandwidth and latency.

TABLE 1. Storage overhead of virtual-channel and flit-reservation flow control. Data flits are f bits wide (f=256 in our example network). In virtual-channel flow control, data flits are padded with the virtual channel identifier and a t-bit wide type field (t=2); In flit-reservation flow control, control flits contain the virtual channel identifier, type field and arrival time stamps for up to d data flits they lead (d=1). b_d refers to the number of data buffers, b_c refers to the number of control buffers, v_d is the number of virtual channels on the data network and v_c is the number of virtual channels on the control network. The reservation tables in flit-reservation flow control have 32 slots, as the scheduling horizon, s, is 32 cycles.

	Virtual-Channel Flow Control				Flit-Reservation Flow Control		
	General	VC8	VC16	VC32	General	FR6	FR13
		b_d=8 v_d=2	b_d=16 v_d=4	b_d=32 v_d=8		b_d=6 v_c=2 b_c=6	b_d=13 v_c=4 b_c=12
Data buffers	$(f + \log_2 v_d + t) \times b_d \times 5$	10360	20800	41760	$f \times b_d \times 5$	7680	16640
Control buffers	-	-	-	-	$(\log_2 v_c + t + (d \times \log_2 s)) \times b_c \times 5$	240	540
Queue pointers	$2 \times \log_2 b_d \times v_d \times 5$	60	160	400	$2 \times \log_2 b_c \times v_c \times 5$	60	160
Output reservation table (Status bits and buffer counts)	$(1+ \log_2 b_d) \times 4 \times v_d$	32	80	192	$(1 + \log_2 b_d) \times s \times 4$	512	640
Input reservation table	-	-	-	-	$[(1 + \log_2 s + 2 + 2 \times \log_2 b_d) \times s + b_c] \times 5$	2270	1980
Bits per node		10452	21040	42352		10762	19960
Flits per input channel		8.17	16.44	33.09		8.40	15.59

till these packets in the sample space have all been received. 5-flit packets are used unless otherwise mentioned. A constant rate source injects packets at a percentage of the capacity of the network and the average latency of the packets is calculated. Latency spans the instant when the first flit of the packet is created, to the time when its last flit is ejected at the destination node, including source queuing time and assuming immediate ejection. This includes the latency incurred by the control flits, as control and data flits are created at the same time, and data flits cannot be ejected till their control flits have scheduled their ejection at the destination node. Across all experiments, we calculated the 95% confidence interval of the average latency measurements and found them to be within 1% error.

To achieve a fair comparison of virtual-channel and flit-reservation flow control, we did a detailed breakdown of the storage and bandwidth overheads incurred by both flow control methods. Experimental configurations where both flow control methods incur approximately the same storage overhead are selected. The extra bandwidth needed by flit-reservation flow control is also accounted for.

The storage overhead of virtual-channel flow control and flit-reservation flow control is shown in Table 1. With virtual-channel flow control, overhead is required to store buffer queue pointers, channel status bits, and next-hop buffer counts. Each data flit is also padded with a virtual channel identifier and a type field distinguishing head, body and tail flits. Flit-reservation flow control, on the other hand, incurs overhead to store the control buffers, their queue pointers, and the input and output reservation tables. The data buffers of flit-reservation flow control contain only payload, as type bits and virtual-channel identifiers are tags on control flits instead. In addition, a control flit contains the arrival times of each data flit it leads. The output reservation table is essentially the equivalent of virtual-channel flow control's channel status bits and buffer counts, archived over the scheduling horizon, while the input reservation table has fields as illustrated in Figure 4(c), together with the buffer occupancy bits.

TABLE 2. Bandwidth overhead incurred by virtual-channel and flit-reservation flow control for each data flit. The packet length is L flits, and the destination field occupies n bits. Other parameters are defined in Table 1.

	Virtual-Channel Flow Control	Flit-Reservation Flow Control
	General	General
Destination	n/L	n/L
VCID	$\log_2 v_d$	$\log_2 v_c$/L [1 + (L-1)/d)]
Arrival times	-	$\log_2 s$
Overhead per data flit (bits)	$n/L + \log_2 v_d$	$n/L + \log_2 v_c/L[1 + (L-1)/d)] + \log_2 s$

Table 1 shows that virtual-channel flow control with 8, 16 and 32 buffers (VC8, VC16, VC32) with 4 buffers in each virtual channel[10] incurs an overhead of 0.17, 0.44, 1.09 flits per input channel. Flit-reservation flow control configurations which incur approximately the same storage

10. Different sizes of virtual channel queues were simulated for each physical channel buffer size. This configuration was found to realize the best performance for virtual-channel flow control.

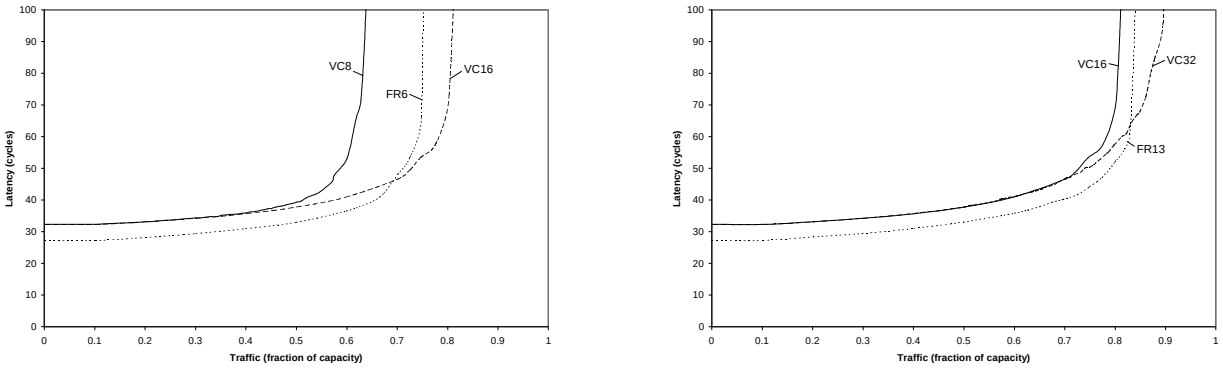


Figure 5. Latency versus offered traffic for virtual-channel (VC) and flit-reservation (FR) flow control with 5-flit packets.

overhead as VC8 and VC16 provide 6 data buffers (FR6) and 13 data buffers (FR13) respectively. In both FR6 and FR13, a control flit leads a single data flit, 2 control flits are injected every cycle and each control virtual channel has 3 buffers. Both configurations have a scheduling horizon of 32 cycles. To compensate for the additional storage which flit-reservation flow control incurs, FR6 has 2 fewer data buffers than VC8, while FR13 has 3 fewer data buffers than VC16.

The bandwidth demands of both virtual-channel and flit-reservation flow control are presented in Table 2. Since FR6 and FR13 use the same number of virtual channels as VC8 and VC16 respectively (i.e. $v_c=v_d$), and 1 control flit leads a single data flit in FR6 and FR13 (d=1), the extra bandwidth overhead incurred by flit-reservation flow control lies in the arrival time field in each control flit. Hence, flit-reservation flow control incurs 5 more bits of bandwidth overhead for a scheduling horizon of 32 cycles, which is 2% for 256-bit data flits.

4.1 Comparison with Virtual-Channel Flow Control

Figure 5 graphs the average latency as a function of offered traffic realized by virtual-channel and flit-reservation flow control. With the same amount of storage, flit-reservation flow control saturates at a higher throughput than virtual-channel flow control. The graphs also reflect a lower base latency for flit-reservation flow control due to the elimination of data routing and arbitration delays.

With 8 buffers per input channel, virtual-channel flow control saturates at 63% capacity. FR6 extends the throughput by 22% to 77% capacity. Biased by the 2% additional bandwidth which FR6 consumes, FR6 improves over VC8 by 20%, taking into account storage and bandwidth overheads. When using 16 buffers per input channel, virtual-channel flow control attains a throughput of 80% of bisection bandwidth, while flit-reservation flow control pushes it further to 85% capacity.

Flit-reservation flow control's efficient use of buffers is reflected in the significantly fewer buffers needed to achieve a certain level of throughput. Flit-reservation flow control with 6 buffers per input channel saturates at a throughput (77% capacity) higher than virtual-channel flow control using 8 buffers per input channel (63% capacity), and close to the throughput obtained by virtual-channel flow control using 16 buffers per input channel (80% capacity). Similarly, flit-reservation flow control using 13 buffers per input channel approaches the saturation throughput (85% capacity) of virtual-channel flow control using 32 buffers per input channel.

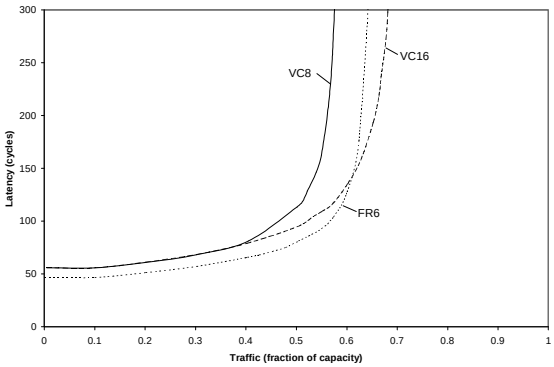
Besides pushing throughput due to immediate buffer turnaround, flit-reservation flow control also eliminates routing and arbitration latency since these decisions are made by control flits in advance[11]. The experiments showed that flit-reservation flow control has a base latency of 27 cycles as compared to virtual-channel flow control's 32 cycles, a savings of 15.6%.

4.2 Effect of Packet Length

Figure 6 shows the latency-throughput curves for virtual-channel flow control and flit-reservation flow control, with 21-flit packets. The base latency is lowered by 9 cycles from virtual-channel flow control's base latency of 55 cycles, a savings of 16.4%. Throughput is also improved beyond the saturation throughput of virtual-channel flow control.

When using 13 buffers per input channel, flit-reservation flow control improves throughput by 13.4% (after taking into account the 2% bandwidth overhead incurred by

11. Both wormhole and virtual-channel flow control can be extended to use a fast control network to hide routing and arbitration latency. Saturation throughput, however, will remain unchanged.



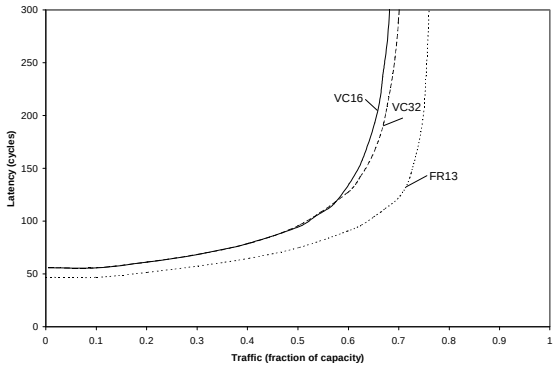


Figure 6. Latency versus offered traffic for virtual-channel (VC) and flit-reservation (FR) flow control with 21-flit packets.

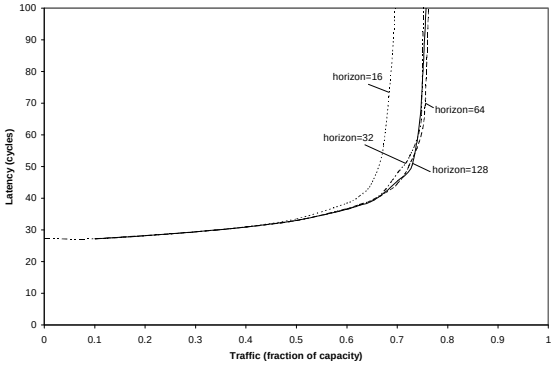


Figure 7. Latency versus offered traffic for flit-reservation flow control with scheduling horizon adjusted from 16 cycles to 128 cycles.

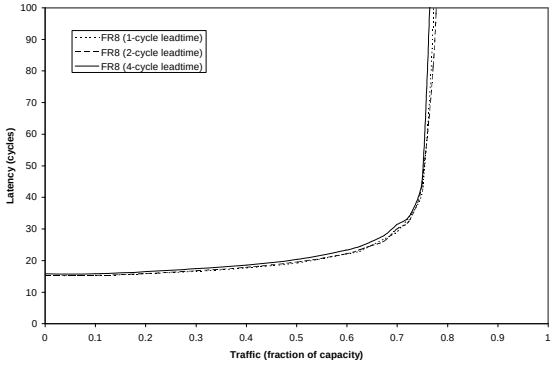


Figure 8. Latency-throughput curve for flit-reservation flow control with control flits leading data flits by 1, 2, and 4 cycles. The control and data networks have the same propagation delay of 1 cycle.

flit-reservation flow control) to 75% capacity, beyond the saturation throughput of virtual-channel flow control (65% capacity) using 32 buffers, reflecting the significant buffer savings enjoyed by flit-reservation flow control.

However, with 6 buffers per input channel, flit-reservation flow control only pushes the throughput envelope slightly from virtual-channel flow control's throughput at 55% capacity to 60% capacity, an improvement of 7% after considering bandwidth overhead. Thus, the effectiveness of flit-reservation flow control is tempered when the buffer pool is small relative to the length of packets. This is because when a packet is blocked, all its data flits are stalled and hold buffers. The longer a packet, the more congestion in the buffer pool. Hence, when the buffer pool size is small relative to the packet length, active packets are often unable to pass blocked packets.

We tracked a specific buffer pool of a router in the middle of the mesh interconnect using flit-reservation flow control with 6 buffers in each buffer pool, and noted that near saturation, the buffer pool is full 40% of the time, as compared to virtual-channel flow control which saturates when the buffer pool is only full <5% of the time. Thus, although flit-reservation flow control uses the buffer pool more effectively, it is unable to turnaround buffers when many of them are actually held due to blocking.

4.3 Effect of Scheduling Horizon

The scheduling horizon determines how far ahead a control flit can make reservations for its data flits. Hence, a scheduling horizon of 32 cycles implies that at time t, the latest departure time that a control flit can reserve for its data flits is 32 cycles in the future, i.e. time t+32. Figure 7 contains the latency-throughput curves of flit-reservation flow control (experimental configuration FR6) with the scheduling horizon adjusted from 16 cycles to 128 cycles.

The figure shows that throughput is relatively insensitive to scheduling horizon. With a horizon of 16 cycles, throughput is within 10% of optimum and there is little improvement in increasing the horizon beyond 32 cycles.

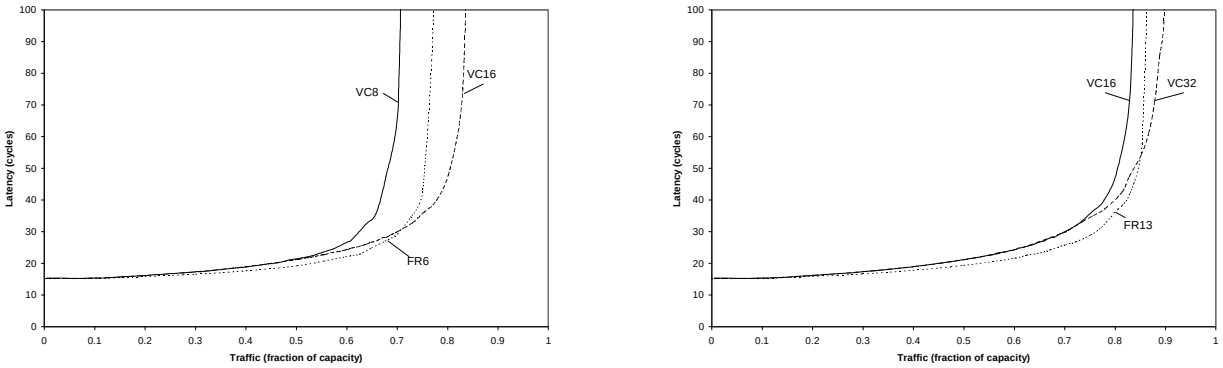


Figure 9. Latency-throughput curve for flit-reservation flow control with leading control instead of fast control. Control flits are injected a cycle before their data flits, on a control network with the same propagation delay of 1 cycle as the data network.

A longer scheduling horizon increases the probability that a control flit will be able to schedule all of its data flits and depart before the arrival of the data flit. However, a longer scheduling horizon can only be exploited if control flits lead data flits significantly. The optimum scheduling horizon thus depend on the relative capacities of the control and data networks.

4.4 Fast control versus Leading control

Flit-reservation flow control requires that control flits precede data flits. This can be accomplished either with fast control wires or by injecting control flits one or more cycles ahead of data flits on a control network with the same propagation delay as the data network.

To explore the performance of flit-reservation flow control with leading control, we modified our simulation to inject control flits N cycles ahead of their associated data flits, with control, data and credit signals traversing wires with the *same* propagation delay of 1 cycle between nodes. Virtual-channel flow control is also simulated on this network where data and credit propagation delay between nodes is 1 cycle. The latency measurements of flit-reservation flow control include the N cycles in which data flits are deferred behind the control flits.

Figure 8 shows the latency-throughput curves of flit-reservation flow control with each control flit injected 1, 2 and 4 cycles ahead of its associated data flits, using the FR6 configuration with 6 data buffers per input. The experiments show that the throughput achieved by flit-reservation flow control is independent of lead time. This is because with a small buffer pool, data flits are frequently stalled due to high contention on the data network. The control flits, on the other hand, traverse the control network with little contention[12] and hence arrive at each hop well ahead of their corresponding data flits. Reservations can thus be scheduled in advance, and buffers reused efficiently to extend throughput. We tracked the average number of cycles between the arrival of a control flit at the destination and the arrival of its first associated data flit, and found that for flit-reservation flow control with a 1-cycle lead time, control flits arrive 14 cycles in advance of data flits when the offered traffic is 77% capacity. This is comparable to the lead of 15 cycles that control flits with a 4-cycle lead time attain at 77% capacity.

The data also shows that delaying the data flits for up to 4 cycles has little effect on overall latency. This delay is more than made up by the elimination of routing and arbitration delay. On the other hand, when control flits are only injected one cycle ahead of data flits, data flits frequently catch up with their control flit and advance scheduling cannot be carried out.

Figure 9 compares flit-reservation flow control with a 1-cycle leading control to virtual-channel flow control on 5-flit packets. The performance improvement is the same as for flit-reservation flow control with a fast control network. While there is no reduction in base latency, flit-reservation flow control reduces latency under moderate to high loads. At 50% capacity, for example, FR6 has a latency of 19 cycles while VC8 and VC16 experienced an average latency of 21 cycles. This savings in average latency is as a result of control flits forging further ahead of data flits when data flits are stalled in a congested data network. Control flit can thus schedule data flits ahead of time, and eliminate routing and arbitration latency.

The base latency experienced by flit-reservation flow control (15 cycles) is the same as that obtained by virtual-channel flow control, as the 1-cycle lag which the data flits

12. In our experimental configurations, each control flit leads a single data flit. Hence, the load on the data and control networks are the same. However, two of these narrow control flits are injected and processed per cycle. Thus, even when the data network is near saturation, the control network is still seeing minimal contention.

TABLE 3. Summary of experimental results.

		FR6	FR13	VC8	VC16	VC32
Fast Control						
5-flit packets	Base latency (cycles)	27	27	32	32	32
	Latency at 50% capacity (cycles)	33	33	39	38	38
	Throughput (% capacity)	77%	85%	63%	80%	85%
21-flit packets	Base latency (cycles)	46	46	55	55	55
	Latency at 50% capacity (cycles)	81	75	113	95	97
	Throughput (% capacity)	60%	75%	55%	65%	65%
Leading Control						
5-flit packets	Base latency (cycles)	15	15	15	15	15
	Latency at 50% capacity (cycles)	19	19	21	21	21
	Throughput (% capacity)	75%	83%	65%	80%	85%

experience is equivalent to the 1-cycle routing and arbitration latency incurred by virtual-channel flow control. If the destination is available ahead of the packet data, the control flits can be injected ahead of the data without stalling the data flits. In this case, flit-reservation flow control with leading control realizes the same latency reduction as with fast control. For this example network with 1-cycle propagation delays, flit-reservation flow control with the control flit leading by at least 10 cycles cuts the base data latency from 15 cycles to 6 cycles.

Table 3 summarizes the experimental results reported in this section, comparing the base latencies and throughput experienced by virtual-channel flow control and flit-reservation flow control.

5. Discussion

All-or-nothing versus per-flit scheduling. When control flits arrive at the output scheduler of a flit-reservation router, its data flits are scheduled and successful reservations are fed back to the input scheduler flit-by-flit. Each successfully scheduled data flit can hence move on to the next hop, regardless of whether all the data flits led by this control flit has been successfully scheduled. An alternative to this is all-or-nothing scheduling, whereby data flits are only forwarded to the next hop if the control flit succeeds in scheduling all its data flits.

The advantage of all-or-nothing scheduling is that no schedule list needs to be maintained, as data flits never arrive before control flits. However, all-or-nothing scheduling results in data flits being frequently stalled in the buffer pool due to insufficient free buffers on the next node to accommodate all data flits led by the control flit. Per-flit scheduling, on the other hand, allows scheduled data flits to move ahead to the next hop, thus freeing the buffers they occupy on the current node. Other data flits bound for this node can then reuse these buffers. As a result, per-flit scheduling attains higher throughput than all-or-nothing scheduling.

Excess capacity on control network. With either fast or leading control, flit-reservation flow control relies on the excess capacity on the control network relative to the load on the data network. When the data network is experiencing high contention, control flits must continue to be able to race ahead through the lightly-loaded control network and schedule reservations in advance, reusing buffers and thus extending throughput. Excess capacity on the control network can be realized through loading the control network with fewer control flits relative to data flits. For instance, if we have 1 control flit leading 4 data flits, for a 5-flit packet, there are 2 control flits for the 5 data flits, resulting in the load on the control network being 40% that of the data network. Another way to realize excess capacity is to increase the saturation throughput of the control network, either by increasing the number of control virtual channels, the number of control buffers, or the number of control flits injected per cycle. Without this excess capacity, when data flits are stalled, control flits will also be facing high contention. Advance reservations will thus be unrealizable and buffers will not be quickly recycled to improve throughput.

Single wide control flit versus multiple narrow-width control flits. Instead of injecting multiple narrow-width control flits as in our example network, one wide control flit can be injected per cycle. The wide control flit leads several data flits. The advantage of a control flit leading a single data flit is that data flits will never arrive before control flits and a schedule list need not be maintained. However, with multiple data flits led by a control flit, bandwidth overhead is lower since only control flits need a virtual channel identifier.

Buffer pool versus distinct buffer queues. Instead of the distinct queues of buffers used in virtual-channel flow control, flit-reservation flow control uses a buffer pool for each input channel. This is because there is no differentia-

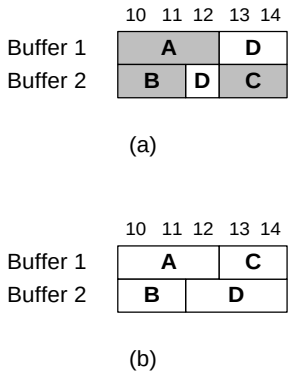


Figure 10. (a) An example where buffer transfers are required due to buffers being allocated at reservation time. Flits A, B and C were allocated buffers before the reservation for flit D arrives. As a result, flit D needs to be transferred from buffer 2 to buffer 1 in cycle 13. (b) Here, buffer allocations are performed just before flit arrivals. Buffer 2 is allocated to flit D since it is the only one empty when it arrives at cycle 12, and flit C will be allocated buffer 1 when it arrives at cycle 13. No buffer transfers are needed.

tion between data flits of different packets on the data network layer. Each data flit is treated similarly, and contains no tag to distinguish it from any other data flit.

It should be noted though that the buffer pool does not account for the improved buffer utilization of flit-reservation flow control. We simulated virtual-channel flow control with a shared buffer pool among its virtual channels [TamFra92], but saw no improvement in network throughput.

Buffer allocation at scheduling time versus just before arrival. Our implementation of flit-reservation flow control reserves a buffer when an input reservation is made but does not assign a particular buffer until the cycle before the flit arrives. If the buffer was assigned at reservation time we could encounter situations where a single buffer is not available for the entire residency of a flit and thus the flit would need to be transferred from one buffer to another during its stay at a node. Waiting until flit arrival to assign buffers eliminates the need for buffer transfers.

Figure 10(a) illustrates a case where a buffer transfer is needed. Data flits A, B and C have buffers reserved and allocated. Buffer 1 has been allocated to flit A which will leave in cycle 12. Data flit B holds buffer 2 and will depart in cycle 11. Thereafter, buffer 2 is allocated to data flit C which will arrive in cycle 12. At this point, the output scheduler notes that there is a free buffer from cycle 12 onwards and reserves it for data flit D which leaves in cycle 14. Unfortunately, as buffers were allocated at reservation time without knowledge of future reservations, flit D will need to reside in buffer 2 for cycle 12 and then be transferred to buffer 1. If buffer allocations are deferred till flit arrival, this situation will not arise, as shown in Figure 10(b).

Synchronization issues. Because data flits are identified solely by their arrival times, it is critical that the routers at the two ends of the link have a common time reference. If the routers operate mesochronously[13], this is not an issue as the timing of the two routers remains synchronized with a fixed offset. With a plesiochronous link, however, it is necessary to make the clock domain crossing in the middle of the input buffer. The input side of the input scheduler and buffer operates in the clock domain of the transmitting router while the output side operates in the local clock domain. In this case, buffers must be held for one extra cycle before releasing them to avoid buffer conflicts when the transmit clock slips a cycle with respect to the local clock.

Error recovery. As with any flow control method involving control flits and state variables, the corruption of control information can make it impossible to deliver data flits. For example, if the time fields of a control flit are corrupted, it becomes impossible to identify the data flits. Similarly if the contents of the input or output reservation tables become corrupted, one or more data flits buffered on the node may be lost.

To minimize the probability of data loss, control flits may be protected by an error detection code and retransmitted in the event of an error. Also, the internal tables may be protected with parity to detect corruption of internal state. When a table error is detected, the affected data flits are dropped. This will be detected on the next hop when an idle pattern is received instead of the affected data flit and the collective state of the scheduling tables will return to a consistent state with no lost buffers or stalled links.

Deadlock issues. Past research into deadlock avoidance for current flow control techniques such as store-and-forward, virtual cut-through [Duato96], wormhole [Duato95] and virtual-channel [Dally87] flow control have identified various dependencies that exist and formally detailed conditions for deadlock avoidance. Much of this theory applies directly to flit-reservation flow control. However, it must be extended to account for the dependencies that may exist in both directions between control flits that travel on virtual channels and data flits that share a single buffer pool.

13. See [DalPou98] pp. 473-475 for a description of mesochronous and plesiochronous synchronization.

6. Conclusion

In this paper, we have introduced flit-reservation flow control, in which control flits race ahead and reserve bandwidth and buffers for data flits ahead of time. Unlike existing flow control methods that hold buffers from the time a data flit departs a node till the time the credit is received, the advance scheduling done in flit-reservation flow control enables a buffer to be held only during actual buffer usage. As a result, buffers can be immediately reused. The advance reservations also reduces data latency, since data flits can pass through nodes without waiting for routing and arbitration decisions to be made.

Simulations comparing flit-reservation flow control with virtual channels show that flit-reservation flow control is able to extend the throughput achieved by virtual-channel flow control. Experiments also show the lower base latency achieved by flit-reservation flow control.

Flit-reservation flow control requires control flits to traverse the network ahead of data flits, making reservations in advance. This can be realized through fast on-chip control wires, or the pipelining of control flits one or more cycles ahead of data flits on a control network with the same propagation delay. Experiments show that when control flits are injected 1 cycle in advance of data flits on a control network with the same propagation delay as the data network, flit-reservation flow control can similarly improve throughput over that of virtual-channel flow control and realize significant buffer savings.

Immediate buffer reuse and elimination of routing and arbitration latency can also be achieved through the use of statically-scheduled flow control, where a compiler schedules the allocation of buffers and channel bandwidth prior to program execution. However, flexibility is sacrificed, as a statically-scheduled network is unable to support dynamic data-dependent communication patterns. By scheduling buffers and channel bandwidth dynamically at packet injection time instead of statically at compile time, flit-reservation flow control is able to offer many of the advantages of statically-scheduled flow control, while supporting the flexibility of a dynamically-routed network.

As technology scales, buffer memory and network latency become ever more critical in communication networks. Flit-reservation flow control is able to achieve savings in both critical resources without sacrificing network performance or flexibility.

Acknowledgements

We wish to thank the anonymous reviewers for their insightful suggestions and comments which contributed to a better exposition of this paper. This work was sponsored in part by Defence Advanced Research Projects Agency (DARPA) under contract number MDA904-98-C-A933.

References

[BLAA99] R. Barua, W. Lee, S. Amarasinghe, A. Agarwal, "Maps: A Compiler-Managed Memory System for Raw Machines", *In Proceedings of the Twenty-Sixth International Symposium on Computer Architecture*, Atlanta, June 1999.

[DalSei86] W. J. Dally and C. L. Seitz, "The Torus Routing Chip", *Distributed Computing*, vol.1, pp. 187-196, 1986.

[Dally87] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547-553, May 1987.

[Dally92] W. J. Dally, "Virtual-Channel Flow Control", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, March 1992.

[DalPou98] W. J. Dally and J. W. Poulton, "Digital Systems Engineering", *Cambridge University Press*, 1998.

[DalLac99] W. J. Dally and S. Lacy, "VLSI Architecture: Past, Present, and Future", *In Proceedings of Advanced Research in VLSI*, Atlanta, March 1999.

[Duato95] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055-1067, October 1995.

[Duato96] J. Duato, "A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 8, pp. 841-854, August 1996.

[DLSY96] J. Duato, P. Lopez, F. Silla and S. Yalamanchili, "A high performance router architecture for interconnection networks", *In Proceedings of the 1996 International Conference on Parallel Processing*, Bloomington, August 1996.

[DYN97] J. Duato, S. Yalamanchili, L. Ni, "Interconnections Network - An Engineering Approach", *IEEE Computer Society Press*, 1997.

[KerKle79] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique", *Computer Networks*, vol. 3, pp. 267-286, 1979.

[Seitz85] C. L. Seitz, "The Cosmic Cube", *Communications of the ACM*, vol. 28, no. 1, pp. 22-33, January 1985.

[TamFra92] Y. Tamir and G. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches", *IEEE Transactions on Computers*, Vol. 41, No. 6, June 1992.

[Tanenb96] A. Tanenbaum, "Computer Networks", 3rd Edition, *Prentice Hall*, 1996