Buffer Allocation Approaches for Virtual Channel Flow Control

Neeraj Parik, Ozen Deniz, Paul Kim, Zheng Li
Department of Electrical Engineering
Stanford University, CA

Abstract

Buffers are one of the major resources used by the routers in virtual channel flow control. It is reported that they consume 60% of routers' power and silicon [5]. Architects of interconnection networks must use the buffers efficiently while assigning buffers to VCs from the fixed budget. This paper analyzes various buffer allocation approaches. All these approaches have constraint on buffer resources at various levels either at network or node, and aim at increasing throughput, minimizing latency using virtual channel flow control. We also briefly mention the buffer allocation approaches that use the freedom provided by virtual channels to meet the application requirement (prioritizing traffic or multimedia traffic). We discuss a unique flow control (flit reservation flow control) method aimed at reducing required buffering and still achieve significantly higher throughput.

Paper organization

We start by analyzing the structure, benefits and performance metrics for the VC Flow Control mechanisms defined by [2]. The detailed throughput analysis for different networks and traffic rates is further provided by [7]. We then look at a different view to formulate the buffer allocation problem at network level provided by [8, 9]. The most efficient buffer utilization could come only from the approach for dynamic buffer allocation to VCs based on network demands provided by [5]. Finally we present a discussion on [6], which is a unique flow control method that reduces the buffering requirement per VC by reducing the credit turn around time to zero. We conclude by presenting our comments and some recent trends in interconnection networks.

Introduction

Channels are expensive resources and packets traversing between two points could block the channel due to lack of resources at the receiver or intermediate points. This problem is unique to networks that use wormhole routing (i.e. attempt to save silicon-space). Wormhole routing is a technique in which a complete buffer is not allocated to the packet before forwarding. The buffer allocation and flow control is done at the flit level. The wormhole routing technique thus reduces the latency of packets and requires less buffering in the network. Even with the wormhole routing technique, it is a common situation for the unrelated traffic to interact in a blocking manner causing lower throughput of the network. In wormhole routing, congestion on the downstream link soon propagates upstream thus blocking the network completely [1]. The link by link cumulative effect of such blocking or congestion spreading can severely affect the network throughput [2]. The problem is unique to networks that use wormhole routing. Traditional packet switch networks that perform flow control at the packet level never block partially transmitted packets. The previously mentioned problem of the wormhole routing method can be solved by adding virtual channels. Virtual channels decouple the allocation of buffer space from channel bandwidth. This allows portions of the packet (flow control units of packets called flits) to compete for the bandwidth of a single physical channel. If a packet is blocked on a physical channel then another packet could bypass it using the same physical channel. Virtual channel design must aim at utilizing the channels to the highest possible extent and also must target silicon space usage.

Virtual channels (VC) were initially used to avoid deadlock in the torus routing chip. A cyclic network with resource dependencies can be made acyclic by routing through appropriate VCs. The output queuing in switches provides only single stage resource decoupling. With output queuing there is still a single buffer associated with the output; a long packet that will not fit in a single node could still flow into the input thus blocking it. Tamir showed that output queued switches could be looked at as split input queued switches. Another use of virtual channels is to guarantee bandwidth to circuits as with iWARP[2]. VCs can be used in both input and output queue switches. In virtual channel flow control, buffers are allocated to each VC. These buffers can be allocated independently or by an on-demand basis. A packet is assigned a VC and owns that VC until it is completely transferred. The packet makes progress using virtual channel flow control performed at the flit level.

However, the advantages of VCs come with overhead; the overhead has various components both in logic required and communication (i.e. amount of information exchanged between nodes). This communication overhead is the VC identification field in the flits; which adds extra bits in the flit contents. The credit information between the nodes also needs to be exchanged on a per VC basis. Also the nodes must maintain per VC state information. The transmitting node keeps state information like count of free buffers, VC assignment and priority of packet. The receiving node also keeps VC assignment, input and output pointers for each lane buffer and state of the channel (waiting, active, idle).

The proposed solution in the paper discusses overhead in terms of fixed segmentation of the buffer (i.e. a given total buffer space budget on a node is equally segmented among all VCs). One must consider dynamic allocation of these buffers to different VCs. This would increase buffer utilization efficiency as well, since not all VCs need the same amount of buffering [5]. Having dynamic VCs provides the basic framework for implementing different schemes. Dynamic buffer allocation to VCs is more complex and thus requires more overhead in terms of control-logic complexity. These dynamic schemes aim at maximizing buffer efficiency.

Adding VCs to the inputs and outputs also increases the complexity of switch design [2]. The buffer allocation scheme at the inputs and outputs could affect the switch performance as well. The problem in the switch is how to treat the VCs at the input and outputs (if the switch treats these VCs as separate channels or multiplexed channels). Treating the VCs as separate physical channels (also called de-multiplexed) increases the complexity of the switch by an order of magnitude. Treating VCs as multiplexed channels leads to complexity in managing flow control and arbitration, since one of the possible VCs must be selected from among a set of eligible VCs at both the inputs and outputs while minimizing the blocking probability. A point in between the two extremes is to have partially de-muxed inputs and a multiplexed output. This gives reasonable performance and flow control complexity. Multiplexing at any level would reduce performance (i.e. can cause un-intentional blocking) but having partially de-multiplexed inputs reduces the probability of un-intentional blocking in the switch allocation stage since more competing inputs are available. The arbitration must take into account the flow control information for that output VC. Further analysis of the complexity of switch allocation and virtual channel allocation stages in a switch micro-architecture for dynamic buffer allocation to VCs shows that it also reduces the complexity of the switch allocation stage and overall area [5].

[2] Presents simulation results for k-ary n-fly and k-ary n-cube network topologies. All the simulations in [2] were performed with a fixed buffer resource budget per node. Adding VCs reduced the amount of buffering proportionally. The simulation results show that VCs increase throughput of the network and at the same time decreases the latency of the network. In general the throughput of the wormhole network reduces with number of stages, but the throughput of the network with virtual channels is also independent of the number of stages. Similar but slightly different results on throughput analysis for various traffic patterns were found by [7]. We analyze results by [7] in the next section of the paper. Lower latency is observed up to a certain amount of virtualization, but excessive virtualization has a negative effect on latency due to higher buffer utilization and bubbling effects in the pipeline. The bubbling happens when buffering per VC falls below the credit round trip time equivalent buffer space. [2] Also implemented a deadline scheduling (by age) for packets and observed lower latencies. [3, 4] Used this (deadline scheduling by [2]) result in real applications and are analyzed after [7]. [4] Provides more detailed results of priority scheduling but didn't use any special buffer allocation schemes as with [2]. [2] Was the first of its kind of paper on VC flow control and this paper defined parameters to access the meritocracy of the Virtual Channel scheme. Further in this paper, we analyze different approaches to buffer allocation schemes for virtual channel flow control and use the [2] defined metrics to access its efficiency.

Virtualization with per node buffer budget

[7] Presents simulation results describing the effects of the number of VCs on the throughput of the system. The study defines the problem that a network-node with fixed buffer space can either be used to create deep buffers per virtual channel or can be divided among different buffers thus creating shallow buffers. [7] Studies the effect of the buffer depth of the VCs and the number of VCs in each dimension on the performance of wormhole

switched mesh, torus and hypercube networks. These networks are studied under dimension-order routing with the total buffer size associated to each physical channel constant. In the Matrix-transpose traffic, the source node (n[1],n[2],n[3]..n[m] sends messages to the destination node (n[m/2],n[m/2+1],….n[m],n[1],n[2],n[3].n[m/2-1]). Given that a network uses Dimension Order Routing, all traffic in the Matrix Transpose traffic would cross the bisection of the network. Hot-Spot traffic pattern is where messages are sent to a specific node with a certain probability and are otherwise uniformly distributed. A source node generates a random number. If this number is greater than a set predefined threshold, the message is sent to the hotspot node. Otherwise, the messages are sent to the other nodes with uniform distribution. The detailed information about the simulation environment and methodology can be found in [7]. The results of the simulation can be summarized in table 1.

Distribution / Load	Uniform	Matrix Transpose	Hot Spot
Low load	Buffer Depth	Buffer Depth	Buffer Depth
High load	Number of VC Buffer	Buffer Depth	Buffer Depth

Table 1: The important metric for the traffic pattern and traffic rate

The above table in words is that for low traffic rate the depth of the buffer is the most important metric for all traffic patterns. For high traffic rate, the best metric depends on the traffic pattern. Since the buffer size per node is fixed, increasing the number of VCs reduces the buffer size associated with VCs. As a result, a message will be distributed over a large number of nodes. High number of VCs increases channel efficiency. However, the distribution of the message over various network-nodes can increase the contention in the network with a fixed-number of VCs per node. As can be seen in the table above, for low traffic rates, the depth of the buffer associated to the VC is more important than the number of VCs and is independent of the traffic pattern and topology. At low traffic rates not many VCs are required, but increasing the number of VCs reduces the buffer size allocated to these VCs. If we increase the number of virtual channels further, the advantages of increasing the virtual channel number will not be exploited. Meanwhile, the buffer size allocated to these virtual channels will decrease and the contention will degrade the performance. For high traffic rates, the trade-off between these two metrics depends on the traffic distribution pattern. Under uniform traffic, as traffic load increases, we see the advantage of increasing the number of VCs. In this case, a large number of messages will be handled by the network simultaneously, so VCs reduce the contention of the network. Excess number of VCs causes an adverse effect on performance.

In the Matrix Transpose traffic pattern, a source node sends a message to the same destination node and these messages follow the same path in dimension-order routing. For this traffic pattern the buffer depth is more important than the number of VCs in both lower and higher traffic rate. Because, the path for all source-destination nodes is fixed in dimension-order routing and the traffic on different dimensions is independent of that on other dimensions. For the Hot-Spot traffic pattern, in a low traffic rate, the important metric is the buffer depth of the VC like the uniform and Matrix Transpose traffic. In a high traffic rate, buffer length is again more important than the number of VCs as with Matrix Transpose traffic. This can be explained as in higher traffic rate, the network immediately saturates and the number of VCs does not have much effect in the performance of the network.

[7] identifies that there is a trade off between increasing the number of VCs and the buffer length associated with these VCs given a fixed buffer budget. There is no fixed limit for the number of VCs for any topology or traffic pattern. One must simulate the network and the corresponding traffic pattern for better performance. The results of [7] seem to be affected by the choice of simulation environment, flow control methodology and routing of the system. Any traffic pattern could match the uniform random distribution by using global oblivious routing; in that case for high traffic it is the number of VCs which will become an important metric and not the buffer depth. Also [7] used a fixed number of VCs for each node in the same dimension throughout the network for all topologies. Traffic in asymmetric topologies, such as mesh, is often high in the center of the network and low on the edges. So to use the budget effectively one must typically use a less number of VCs on the edges than in the center [8]. [7] also reports degraded performance on the use of excessive virtualization which could be an artifact of the flow control scheme used in the network that has a higher credit round trip latency [1]. However, the scheme described by [6] could reduce the

minimum buffering requirement required to keep the channel busy by reducing the credit turnaround time. Another common phenomenon at low loads is excessive bubbling in the data path [4], due to credit information propagating across the network. This effect can be reduced by increasing the buffer depth per VC since credits now need to propagate through a fewer number of nodes. [7] doesn't identify any such observation but rather mentions contention as the source of network performance degradation at low traffic rates.

Supporting application requirements in a buffer constrained network using VCs

[2] States that VCs also provide freedom in allocating buffer resources to packets in the network. This freedom can be used to allocate buffer resources in random, round-robin or prioritized ways. [4,3] Doesn't use any special buffer allocation mechanism either to prioritize traffic or to reduce jitter to support multimedia applications. [4] Discusses a scheme to prioritize delivery of certain packets in networks using VCs. The scheme of [4] provides more bandwidth per round robin cycle to the flits from higher priority. [4] Does reduce the latency of the higher priority traffic. [4] Uses a simple buffer allocation mechanism in which each virtual channel is assigned a fixed number of flit buffers. The scheme relies on the fast siphoning of higher priority buffers and thus fast reverses the flow of credits to prioritize service. Even with prioritization in place, the VC based network cannot deliver traffic with low-jitter and deterministic data rate. [3] Describes a hybrid network that supports both circuit switching and virtual channels. The network uses admission control for multimedia streams that request a certain data rate. The streams are then allocated a fixed number of slots per round robin cycle to achieve the requested data rate. The network node implements a central pool of buffers. This pool of buffers is allocated to both the streams using circuit switching as well as to the packets using virtual channel wormhole flow control. The arriving control packets, which are basically used for circuit establishment doesn't use any buffers since virtual-cut-through is used to forward these control packets. [3] uses a central-RAM based buffer organization due to a high number of VCs. [3] also implements a certain amount of buffering in registers at the inputs and outputs to hide the latency of accessing RAM during the read and write operations. [3] uses a register based buffer organization instead, since the latency of accessing RAM could be the bottleneck when packets or the number of buffer resources allocated to a packet at the node are only a few flits in length. [4] Presents simulation results which shows a 35% improvement in the delivery of priority messages (20% of packets) with minimal effect on the average latency of other traffic. The improvement in latency of higher priority traffic is higher for longer messages. Longer priority messages starts affecting the average latency of other types of traffic.

VC allocation with a network level buffer budget

[8] Addresses a unique problem for on-chip interconnection networks, where channel bandwidth is comparatively abundant, but the memory is relatively expensive. Besides the tight buffer budget, on-chip networks are always application oriented, which gives us some knowledge about the expected traffic pattern. This could be used to optimize for the number of VCs per-node on-chip. Since not all the nodes in the network are equally congested, the uniform-allocation of VCs to each node is not the optimal solution. For example, in the case of mesh networks, the edges carry less traffic than the center. [8] Assumes that Head of Line (HoL) blocking is the most common cause of network performance degradation and this could be improved by adding VCs to the channel. [8] Addresses the buffer allocation problem at the network level. It allocates VCs (which consume buffer resources) to different nodes in the network. [8] Allocates fixed size buffers to each channel, so the problem is now reduced to how to allocate a fixed VC budget to maximize the network performance.

[8] Describes a methodology to allocate the VCs across the network. After preparing the simulation model for the topology, routing algorithm and traffic pattern, [8] starts the process of allocating VCs across the network with one VC per channel. Then it uses a greedy algorithm to allocate the VC budget. The algorithm steps are described below:
1. A traffic analyzing algorithm (described below) is used to locate the most congested channel with multiple active flows. Only the congested channels with multiple active flows are applicable to be granted VCs.
2. Add a VC to this channel.
3. Go to 1, until the VC budget is used out.

[8] Uses an algorithm based on a probabilistic model to analyze the traffic and estimate the congestion on each channel. The packet injection

per node is derived from the traffic pattern. This is used to simulate the observed traffic model at each channel of the network. With a known traffic model at each channel, the expected contention probability could be calculated.

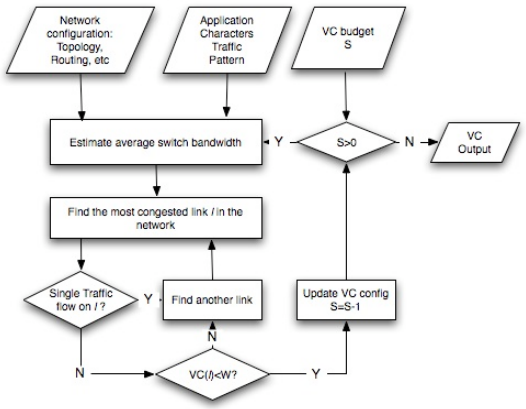

Figure 1: VC planning flow chart in [8]

[8] uses several synthesis and real traffic patterns to evaluate the result, including the Hot-Spot traffic described in [7] above. [8] Simulates a 4x4 mesh network and claims to achieve a 12.1% increase in throughput with one extra VC only. [8] Achieves the same performance as with a uniform 2 VCs across all nodes while allocating only half the number of VCs (hence buffers). [8] Also achieves lower latencies. Results with a real application are also mentioned in [1] .

[8] Assumes that the traffic pattern is well known. This is not the case for most interactive applications. This reduces its applicability only to a narrow set of interconnection networks. It also assumes that the designed network is scalable. However, a different set of problems like on-chip wire delays affects the network performance [5]. The cost of building the network suggested by [8] is high due to the different number of VCs across the network. Each chip or node is unique in structure and inception. [8] uses the greedy algorithm to allocate VCs rather than having a global picture which could give sub-optimal results.

As already stated above, [8] assumes most of the network performance degrades due to HoL blocking. The results showed by [8] use only a deterministic algorithm in its simulation. It may be possible to get better performance for the same budget with adaptive routing. However, the adaptive algorithms are not easy to model in simulation due to its real-time nature. Correct modeling of an adaptive algorithm will also require a simulation model to match on-chip latencies. [8] Allocates VCs of fixed-depth to the channel, which is not the most efficient way to utilize the expensive buffer resources. [9] Shows what results from allocating buffers at the flit level rather than adding VCs to congested links (m flits each). The same network performance could be achieved at 85% less resources in the baseline wormhole flow control. This could be explained by the fact that HoL wastes buffer resources. A fixed length buffer such as a VC is allocated to a packet on each node. If this packet is blocked elsewhere then the unoccupied buffers cannot be recycled to other VCs for the static VC allocation cases. This leads to a waste of buffer resources. [9] Also states that having granularity of a flit to allocate resources to a VC will lead to the most efficient utilization of buffer resources. This idea is used in [5], which discusses an on-chip regulator with flit level granularity for allocating resources to a VC. The most efficient way to formulate this problem is, given a fixed number of buffer resources across the network, study the traffic pattern for a given topology and routing algorithm and allocate the buffers non-uniformly across the chip. These non-uniform buffers can then be used for a dynamic number of VCs. Dynamic VC allocation is especially useful for networks with bursty traffic sources as the same buffer resources can then be used by the traffic sources as the flows are established and torn down [9].

Dynamic VC Allocation with per node buffer budget

ViChaR [5] claims that buffer budgets in an on chip network can best be utilized using dynamic virtual channels. [5] proposes a scheme called dynamic virtual channel regulator (ViChaR). In this scheme VCs are allocated dynamically and are assigned buffer resources according to network traffic. ViChaR maximized throughput and buffer utilization by dispensing a variable number of VCs on demand. [5] also claims that Network-on-Chip power consumption will eventually surpass logic and memory power, so an optimal Network-on-Chip with the least number of buffers that meets the demands of traffic must be designed. ViChaR is based on results observed by [7] that network performs best with a low number of VCs, each having deep buffers on low-loads. [7] also recommends increasing virtual channels for high traffic with optimal buffer depth (~credit round trip latency buffering). In static VC allocation schemes

like [8, 9], size and the number of VCs are design time decisions and thus doesn't perform optimally for all traffic conditions [7]. ViChaR claims that there is no overhead in logic complexity due to this unique dynamic buffer allocation scheme. This is due to trade offs in VC allocation and switch allocation stages of the canonical switch architecture. ViChaR claims ~4% reduction in logic area due to this trade off in microarchitecture. It also reports a 25% increase in performance with the same amount of buffering. ViChaR uses registers (flip-flops) for buffering purposes to avoid the latency penalties incurred when buffers in SDRAM / DDRAM are accessed.

ViChaR uses the work by DAMQ (Dynamically Allocated Multi-Queues) and Fully Connected Circular buffer (FC-CB) as the base to start the analysis. DAMQ uses a fixed number (four) of queues at the input ports and link-list based read and write pointers. The fixed number of queues could still cause HoL (Head of Line) blocking and hence buffer under utilization. The link-list based structure is logically very complex and has a three cycle latency in flit arrival to departure. FC-CB is also a scheme with a fixed number of VCs and is logically complicated; in this scheme the switch fabric bloats up the design to cubic complexity. FC-CB had higher dynamic power consumption also. ViChaR also assumes that HoL (Head of Line) blocking is the main cause of buffer underutilization (thus affecting performance) as observed by [8].

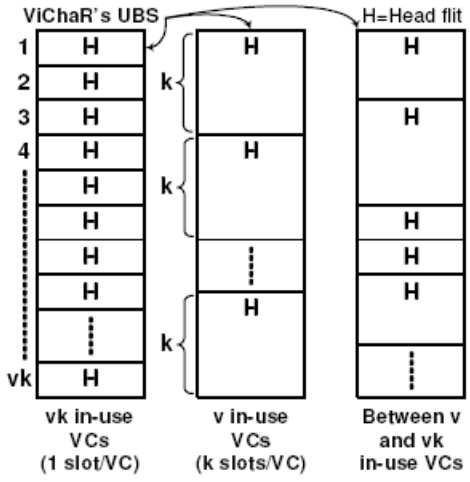

Figure 2: Possible configurations in ViChaR

The figure 2 above shows the buffer allocation in ViChaR at different traffic conditions. ViChaR uses a Unified Buffer Structure (UBS) at each input port to share the same buffer among all the VCs associated with this port. It implements a UCL (Unified Control Logic) to dispense VCs and associate buffers to these VCs dynamically. The ViChaR consists of 5 logic blocks per port namely the VC dispenser, slot (flit-space) availability tracker, arriving / departing flit pointer logic, VC availability tracker, VC dispenser and VC control table. The two of these blocks, namely arriving /departing flit pointer logic, and slot availability tracker can be thought of as attached to the input side and the other three blocks are closely related to the output port. As mentioned previously, ViChaR uses a register based buffering for the flits. The MUXes and DEMUXes in the ViChaR are not made extra large, but instead ViChaR uses the same MUXing and DEMUXing scheme as a generic router. ViChaR's "VC dispenser" assigns a new VC to each incoming packet to avoid HoL (Head of Line) blocking. The slots assigned by the "slot availability tracker" to the incoming flits of each packet in the UBS may not be contiguous. These slots assigned to a single packet are tracked by the "VC control table". Each output port has its own "VC control table" which tabulates all the packets vying for that output port. "VC control table" also lists all the slots occupying the flits of that packet in the UBS. VC availability tracker keeps information on the state of VCs to free up and re-allocate VCs. The flow control scheme used by the ViChaR is on-off style flow control, in this scheme the incoming packet stream is throttled, when slot indications from the "slot availability tracker" are exhausted.

Simulations with Uniform random, tornado traffic were performed. The network was also simulated with normal-random and self-similar traffic. [5] Claims that ViChaR network with escape channels using deterministic order routing is deadlock free. The uniform random traffic pattern on a 8x8 mesh achieved results similar to [8, 9]. [5] Also claims no increase in latency due to higher buffer utilization where as [2] explained the observed higher latency at excessive virtualization due to higher buffer utilization. [5] Also showed better throughput and latency results than DAMQ or FC-CB designs using similar buffer resources.

In the following discussion, we assume a v*k flit buffer is used by UBS where v will be the number of active VC's with k flits each for the equal distribution of buffer resources case. P is the number of input ports and output ports at the switch. It is important to see that the VC allocation and SA allocation stages are handled differently in the ViChaR [5]. Each input port can have up to v*k

VCs arbitrating for P output ports (using table based dynamic VC allocation). Similarly the SA allocation is vk: 1 VC arbitrating for P output ports. Hence, the ViChaR uses different VC allocation and SA allocation architectures. The table below compares the VC allocation and SA allocation to a generic router.

	VA stage-1	VA stage-2	SA stage-1	SA stage-2
Generic Router	Total of Pv v:1 arbiters	Total of Pv v:1 arbiters	Total of P v:1 arbiters	Total of P P:1 arbiters
ViChaR	Total of P*P vk:1 arbiters	Total of P P:1 arbiters	Total of P vk:1 arbiters	Total of P P:1 arbiters

Table 2: Resource usage by the ViChaR and generic router

From the table we see that, if we have a high value of k (i.e. average buffer space per VC per input port in the UBS) then overhead in logic of the ViChaR will increase significantly; which means that for large buffer structures (large size of UBS) the overhead is significant. So in order to reduce k (i.e. per VC buffering requirement), a scheme like flit-reservation flow control as proposed in [6] may be used. This could further improve of buffer space savings thus provides attractive trade-offs with switch logic complexity. It is important to note that ViChaR uses register based buffering when reporting significant performance benefits. The scheme proposed in [4] uses SDRAM based buffering for mid to large structures. One simple view of the dynamic VC allocation mechanism is that it achieves efficient buffer utilization by not allocating space to blocked packets and preserves that space to be used by active packets. This dynamic structure makes supporting various applications governed traffic handling schemes (e.g. round robin), as used by [3, 4] difficult. The proposed ViChaR architecture could easily enter in a situation where all the active VCs are competing for a blocked output port (which could be down due to a fault) thus stalling the network completely. Thus the industrial ViChaR network must be made fault tolerant. Overall, ViChaR is an excellent architecture with huge benefits.

Reducing the per node buffering requirement by using flit reservation flow control

[6] Proposes a unique flow control method called flit reservation flow control (FRFC). This method aims at reducing the credit turn around time to zero. Reducing the credit turn around time to zero means that amount of buffering per VC required to keep a particular VC busy is reduced. FRFC decreases the buffer utilization of the network as buffers in the network are usually waiting for credits to return to be utilized again. In FRFC, control flits traverse the network in advance of data flits, reserving buffers ahead of time. The control flits and data flits can either be on separate networks or the same network. This data path scheduling ahead of data arrival allows buffers to be held only when the buffers are in actual use, unlike other flow control methods.

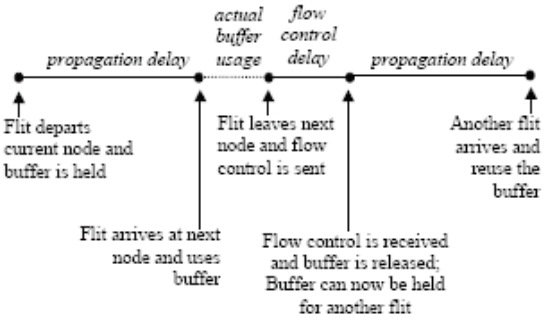

Figure 3: Timeline showing the buffer turnaround time for wormhole and virtual-channel flow control [6]

To understand the credit turn-around time we have to look at the pipelined view of the router. The pipelined view of router breaks the process of forwarding data flits into several small steps. In traditional flow control methods like wormhole routing, a buffer is first allocated (when the SA stage finishes) and held from the time the data flit departs the current node, to when it is processed by the next node, to the time the flow control signal returns to inform the current node that the buffer can be released. Only after this turn-around time the buffer can be reused. So the credit turn around time of the buffer is at least the sum of the propagation delay of the data flit in the forward direction and the flow control back. In FRFC, this turnaround time is essentially zero.

The method in [6] also streamlines the flow of credits. Flit reservation flow control falls back to normal wormhole routing performance in the case of heavy congestion since the buffer scheduling in advance may not succeed. The control flits carry information about the data packet destination and the offset in terms of cycles for the corresponding data flit that arrives. Once the control flits are received and the routing logic determines the output

port for this control flit. The control flit is then forwarded to the corresponding output port. The output port maintains an output reservation table, which contains cycle by cycle (up to certain number of cycles in future) information about the physical channel busy (attached to this output port) and a count of free buffers on the next node. The control flit is then forwarded to the output port, where it has made reservations in future time slots for all the data flits associated with this control packet. The data offset in the forwarded control flit must be updated as well.

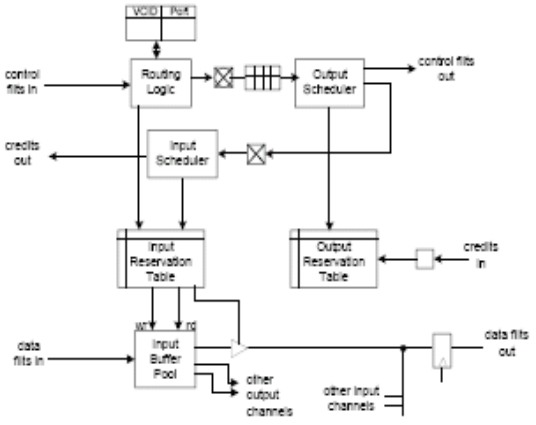

Figure 4: Block diagram of a flit-reservation flow control router [6]

After forwarding the control packet, the output reservation information is passed to the corresponding input port for this packet flow and updates the input reservation table. The input reservation table maintains cycle by cycle information of the data flits arriving, the corresponding output port and the offset of departure from arrival. The input scheduler uses the information from the output scheduler about the departure time of the flits and the output port number to populate its input reservation table. The input scheduler then generates a credit flit to the transmitting node informing it of the availability of the buffer. The transmitter then knows about the exact cycle numbers for which the buffer will be held in the receiving node. The control flits can be of two types namely the head flits and the body/tail flits. The body flits contain arrival time information of multiple successive data flits.

[6] Presents results in which this scheme approaches the throughput (~77%) of Virtual channel flow control with only 6 buffers as compared to 16. For an 8x8 mesh network with eight flit buffers per input, virtual-channel flow control saturates at 63% of the bisection bandwidth while FRFC achieves a performance of 77% of bandwidth, resulting in a 20% improvement with equal buffer storage. This is due to higher buffer utilization thus lower buffer space for the same throughput and performance. One of the benefits pointed out about flit reservation flow control is that for a fixed amount of buffer space, saturation throughput is increased because of more efficient buffer scheduling. Another benefit of FRFC is that it providing similar advantages of the statically-scheduled flow control method, where the flow of data is compiled before runtime. FRFC provides an added benefit in that it supports the flexibility of a dynamically routed network. Instead of scheduling resources at compile time, the scheduling decisions are made when data packets are injected, allowing the schedule to adapt to different traffic patterns. Both methods use buffers efficiently, with immediate turnaround and routing and arbitration latency hidden, but FRFC doesn't sacrifice flexibility.

FRFC seems a bit complicated and logic intensive. Even though it saves buffers, this savings could be offset by the extra logic needed to implement the design. There could also be a lot of communication overhead with the control flits which could make the network saturate at lower throughputs for uniform random or well–behaved traffic patterns. FRFC also requires synchronization between nodes which might make it difficult to implement for chip-to-chip networks (however, a mechanism to maintain synchronized clocks across chips is described by IEEE-1588). Since FRFC uses time offsets, errors in this time field could be difficult to recover from. One possible ways to recover is to have the time field round off at some high threshold. This threshold must be big enough to ensure the maximum lifetime of packets in the network. This would be similar to a scheme used by TCP in WAN. However, this would further increase the overhead associated with the flits. Another drawback of FRFC is from its architectural aspect that uses a shared pool of buffers. If the buffer pool is small compared to packet lengths and a packet is blocked, future data flits could be stalled waiting for a buffer. The situation becomes worse with increased packet lengths.

Conclusion

Virtual channel flow control is one of the most popular flow control methods used in interconnection networks due to its various benefits: like efficient buffer utilization, high throughput, low latency and network throughput independent of the number of stages. In this paper, we discussed the importance of buffer allocation for virtual channel flow control. Various different buffer allocation approaches were discussed that are applicable at the node-level and network level. A node level approach was provided by [7]. A network level approach was discussed by [8]. The common consensus is important to maximize the buffer utilization. For higher buffer utilization one should consider using schemes like flit-reservation flow control, this scheme completely avoids the idling in buffer utilization due to the credit turnaround time. One of the most important mechanisms for buffer allocation is dynamic-buffer allocation with a fixed budget resource per node. This mechanism automatically adapts to deep buffers per VC with a low number of active VCs for low loads. The number of active VCs per node (bounded by an upper limit when excessive virtualization can hinder throughput due to bubbling) are increased as the load of the network increases.

As per-channel on-board bandwidth is increasing and chip to chip communication is moving to high speed serial links, the virtual channel flow control mechanism is not being used. This is due to excessive latencies being incurred by these high speed serial links. The cause of this increase in latency is combined with the effect of serialization latency but the error encoding and transmission control mechanisms used on these links improves the high bit error rate. This excessive latency increases the amount of buffering required per VC to keep a channel busy to an extent that it exceeds the packet length supported by most systems thus making the virtual cut through mechanisms more appropriate. We see more and more virtual-cut-through in use for on-chip communication links like PCIe, RapidIO and Infiniband.

However, due to rapidly shrinking feature size and increasing on-chip wire delays, the Network-on-chips (NoCs) are becoming quite popular. Traditional on-chip wiring methods exceed the on-chip power and delay budgets and have seen the onset of NoCs. The number of resources used by the router are major cost and power consumption determiners. Routers consume 60% of the chip's buffers and power, making them a major power consumer in NoCs. These NoCs will be VC-based networks with fixed resource budgets. This paper discussed various approaches which can be used by these emerging NoCs.

Bibliography

[1] William J. Dally and Brian Towles, Principles and Practices of Interconnection Networks, ISBN: 0-12-200751-4, Morgan Kaufmann, 2003.

[2] W. J. Dally, Virtual-Channel Flow Control, IEEE Transactions on Parallel and Distributed Systems, Volume 3, Number 2, March 1992, pp 194-205.

[3] Jose Duato1, Sudhakar Yalamanchili2, M. Blanca Caminero3, Damon Love2, Francisco J. Quiles3, "MMR: A High-Performance Multimedia Router - Architecture and Design Trade-Offs"

[4] Abdel-Halim Smai, Dhabaleswar K. Panda, Lars-Erik Thorelli, "Prioritized Demand Multiplexing (PDM): A Low-Latency Virtual Channel Flow Control Framework for Prioritized Traffic"

[5] Chrysostomos A. Nicopoulos, Dongkook Park, Jongman Kim, N. Vijaykrishnan, Mazin S. Yousif, Chita R. Das ViChaR: "A Dynamic Virtual Channel Regulator for Network-on-Chip Routers"

[6] L. Peh, W. J. Dally, "Flit Reservation Flow Control", 1999

[7] Rezazad, M.; Sarbazi-azad, H., "The effect of virtual channel organization on the performance of interconnection networks" Parallel and Distributed Processing Symposium, 2005. Proceedings of 19th IEEE International, 4-8 April 2005, Page(s):8 pp

[8] Ting-Chun Huang, Umit Y. Ogras, Radu Marculescu, "Virtual Channels Planning for Networks-on-Chip"

[9] J. Hu, R. Marculescu, "Application-Specific Buffer Space Allocation for Networks-on-Chip Router Design"