Bubble Sharing: Area and Energy Efficient Adaptive Routers using Centralized Buffers

Syed Minhaj Hassan
School of Electrical and Computer Engineering
Georgia Institute of Technology
Email: minhaj@gatech.edu

Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology
Email: sudha@ece.gatech.edu

***Abstract*—Edge buffers along with multiple virtual channels have traditionally been used to provide deadlock freedom guarantees in on-chip networks. The problem with such schemes is their high buffer space requirement which consumes significant power and area. In this work, we propose *bubble sharing* flow control to provide deadlock freedom with small, shared central buffers, eliminating edge buffers, improving buffer utilization, and decreasing router buffer requirements. The key insight involves sharing of the flit-size bubbles (free buffers) among cyclic network paths via central buffers in the router, reducing the overall router buffering space requirement. This technique effectively reconciles the trade-off between high radix and buffer space, encouraging the use of low hop count, high-radix topologies, with both deterministic and adaptive routing. Comparisons show improvement in average packet latency by 31% as compared to traditional 2VC edge buffer routers with 33% reduction in area for an 8x8 generalized hypercube topology.**

I. Introduction

The state of the practice for baseline network-on-chip (NoC) routers has been the use of edge buffers, whose buffer capacity requirements are proportional to the router radix and link length (to fully utilize the link in the presence of flow control delays). These buffer requirements are commonly increased through the use of virtual channels (VCs) to ensure deadlock-free routing, and further increased multiplicatively with the number of message classes, to avoid protocol deadlock [3] [7]. This results in substantial area and power devoted to buffers, up to several hundred KBs of storage for a 32-64 node NoC. These overheads mitigate the advantages of NoCs, specially with high radix routers (which have low hop count and utilize the increased wiring density of NoCs more effectively). We argue that the desirable design point for NoCs is the one with high radix, low buffer space, and supports both adaptive and deterministic routing.

This paper proposes a router architecture that effectively reconciles this trade-off between radix and buffer space, using a novel combination of flow control and buffering strategies. The key idea is to use shared central buffers coupled with novel wormhole-based extensions to bubble flow control. Techniques for ensuring deadlock freedom have evolved from early channel dependency-based techniques to the more recent buffer management approaches, wherein progress is ensured by guaranteeing the availability of free buffer space (bubbles) in every cyclic packet path. The key insight in this paper involves the sharing of the flit-size bubbles, (free buffers), among cyclic network paths via central buffers in the router, hence called *bubble sharing* flow control. These extensions minimize the amount of buffer space that has to be reserved to ensure deadlock freedom, permitting the use of small, low-cost central buffers without the need for edge buffers, increasing area and energy efficiency. The buffer size is independent of radix, but grows slowly with the number of escape paths. The result is a large reduction in the buffer space for adaptive or deterministically routed high radix NoC routers, harnessing the benefits of high radix, while minimizing traditional high buffer space overheads. For example, in an 8x8 generalized hypercube, our adaptive bubble shared router decreases the average packet latency by 31% over a traditional 2VC router with 33% reduction in area.

This paper makes the following contributions.

- Introduces *Bubble Sharing*, a flow control technique that extends the worm-bubble flow-control [1] scheme to centralized buffer routers, reducing its buffer space.
- Proposes *Adaptive Bubble Sharing* that enable adaptive routing with bubble sharing flow control for wormhole switched networks.
- Evaluates a centralized buffer router architecture that implements the bubble sharing scheme, and compares its power and performance against competing approaches.

The remainder of the paper is organized as follows. Section II explains the importance of reduction in the buffer space and the recent proposals that enable it. Section III describes the bubble sharing scheme and its adaptive version. Finally, section IV compares the power and performance of our scheme with different state of the art low latency routers.

II. Background & Motivation

A. Need for Buffer Space Reduction

Traditional virtual channel based routers use edge buffers for deadlock freedom and performance optimization. This puts increased pressure on buffering requirement of an on-chip network. The problem with large buffer space is high area and static power that dominates all other components in traditional networks. Buffer storage in these networks increases with an increase in the number of ports, the number of virtual channels, the number of message classes, and the link width. Furthermore, to keep the links fully utilized, each buffer has to be large enough to hide the credit round trip delay, which becomes higher with longer links. Thus, a 5 ported router with 2VCs, used in a 2 dimensional torus, with 16 byte wide links, and 5-flit input and 2-flit output buffer requires $(16*5*(5*2+2) = 960) \sim 1K$ bytes of storage per message class. With 3 message classes (as required in MEOSI directory protocol) and an 8x8 network, the buffer space required by the network approaches $1*3*64 = 192KB$. In addition, packet

sized buffering is required in the network interface for both injection and ejection that further increases the buffer area.

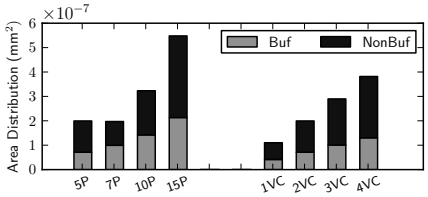

Fig. 1. Area distribution with different ports & VCs

Figure 1 gives the breakdown of buffer vs. non-buffer area, (calculated from Orion2.0 [16]), with different ports and VCs. The four bars on the left gives the area by changing the number of ports. Input buffer size, output buffer size, and link width is 5 flits, 1 flit, and 128 bits, respectively. VCs and message classes are fixed at 2 and 1 respectively. With small number of ports, e.g., in 2D torus, buffers constitute more than 60% of the router area. By increasing the number of ports to 10, (as required by a 4x4x4 flattened butterfly), the buffer area increases by 2x. Note that this does not account for extra buffering required to keep the pipeline busy with longer links. The bars on the right sets the number of ports to 5 but increases the virtual channel count. Again, even with 4 VCs, buffers constitute almost 55% of the total router area.

It is evident that reduction in buffer space with minimum cost in performance is desirable. In this work, we propose using shared central buffers coupled with novel variations of bubble flow control to achieve this goal.

B. Related Work

Many recent works have addressed the buffer space reduction problem [15] [12]. The main focus has been to either share the buffers among the VCs, or reduce the number of entries per input buffer. The extreme case includes various versions of buffer-less flow control that removes the input buffers altogether, by the use of deflection routing [10] [6]. The problem with buffer-less routers is their low saturation throughput and out of order delivery of packets. Out-of-order delivery requires greater buffering at the network interface, negating much of the advantages. Some other techniques include flit-reservation flow control [14] and whole packet forwarding [8] that focus on the efficient utilization of the input buffers. Recently, the use of shared central buffers have been proposed to keep the buffering space largely independent of radix [5]. We have extended the ideas used in their scheme to flit level, for wormhole networks, using bubble flow control [11].

We will next discuss the key ideas behind centralized buffer routers and bubble flow control, that target reduction in the number of input buffers and VCs, respectively.

1) Bubble Flow Control and its Variant: Bubble Flow Control (BFC) has been proposed [11] to avoid deadlocks in a packet based ring without the use of VCs. The basic idea is to ensure that at least one packet sized bubble (empty buffer space) is kept in the ring all the time, even after a new packet is injected into the ring. This is ensured locally by allowing injection in any ring only when an empty space of 2 packets is available at the input port. The problem with this local bubble flow control (LBFC) is that it keeps too many empty packets in the ring. Critical Bubble Scheme (CBS) [2] was proposed to ensure that only one global empty packet is required in each ring, by marking one packet sized bubble in the ring as critical. Injection into the critical bubble is prohibited, keeping it preserved all the time. Both LBFC and CBS, however, only works for packet based networks. A wormhole-based version of CBS called worm-bubble flow control (WBFC) was presented recently [1]. The idea is to mark flit size bubbles in the input buffer as critical before injection, and keep a count of the marked bubbles. Once an injection port has marked enough bubbles to hold a complete packet, a new packet is allowed to enter the ring. In this way, the original critical bubbles of the ring, (inserted at initialization), will always be maintained. We extended the WBFC scheme to routers with shared central buffers, called *Bubble Sharing*. Both WBFC & Bubble Sharing schemes are further explained with an example in Section III-A.

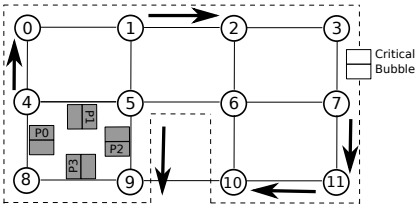

Fig. 2. Bubble Coloring Scheme (BCS)

[17] presents bubble coloring scheme (BCS), a method to perform adaptive routing using bubble flow control in packet based networks. The basic idea is to maintain a virtual ring with a critical bubble that connects all the routers of the network. This ring will always be kept deadlock free, and can be used as an escape path for adaptive routing. Consider the example of a mesh shown in Fig 2. The dotted line represents a fully connected virtual ring utilizing some channels of the network. A critical bubble is maintained in the ring using the injection/ejection rules of CBS. This bubble will allow packets in the ring to always make forward progress. Packets that are not in the ring can always contest for injection into the ring. e.g. in Fig 2, four packets are waiting on each other in a cycle. However, packet 0 is also contesting for the north port, which is part of the virtual ring. The critical bubble present in the ring will move backwards, allowing packet 0 to escape the cycle. This scheme was proposed for packet based edge buffer networks.We have used the basic idea of providing an escape path using a virtual ring to design our adaptive bubble sharing scheme for wormhole based centralized buffer routers. Section III-B provides the details.

2) Centralized Buffer Routers for High Radix Networks: [5] proposes centralized buffer routers (CBR). The basic idea is to remove the edge buffers per port, and use a central buffer to be shared among all ports. The router micro-architecture, with modifications for the current scheme presented as shaded regions, is given in Fig 5. The major components of the router are the central buffer (CB) and the crossbar, with single flit input and output staging buffers. The central buffer is bypassed

in case there is no conflict at the output port. Packets that enter the central buffer are prioritized over packets at the input buffer that arrive later. This is done by three different allocation stages that work in parallel (i.e., IBSA, CBSA, and CBA). The authors argue that because of the bypass path, only the packets that conflict at the output ports will enter the central buffer. This means that the probability of conflicts at the input port of the central buffer is low. However, if multiple packets do collide, their entry into the central buffer will be serialized. They note that the waiting time of this serial entry is very small (equal to pkt_size). This allows them to keep the number of input ports of the central buffer as one, keeping its area small. This holds true for the output port of the central buffer as well.

The scheme uses elastic buffer links [9] instead of credit based flow control. This allows downstream packets to not wait on the credit round trip latency, avoiding holes among subsequent flits, even with single flit staging buffers at the input. Moreover, the control and data information in the links are split to make the bypass path latency single cycle. The design uses a variant of localized bubble flow control. The idea is to inject in a ring only if there is enough space in the central buffer to drain the complete packet, and still leaves at least one flit-sized bubble in the ring. This scheme was typically proposed for large radix routers, which require substantial buffering at the inputs. The shared central buffer reduces this requirement proportional to packet size and network dimension, i.e., 2*dim*PktLength+1 [5]. Our Bubble Sharing scheme took the basic CBR design and extended the WBFC idea to be used with central buffers. This reduces the buffering requirement to flit or worm-bubble level.

III. Bubble Sharing Flow Control

A. Bubble Sharing with Central Buffers

This section describes our bubble sharing scheme. We first explain the key ideas of WBFC using an example and points out the modifications required to adapt it to CBRs. We organized the central buffer as dynamically allocated multi-queue (DAMQ) [15] with shared pool of small worm-bubble sized slots. Each slot has a space of 2-3 flits and assigned completely to an output port; that is, once a slot is assigned to output *x*, all entries will be consumed by packets going to output *x*, until the slot is unassigned. Each slot act as a worm bubble, multiple of which can be taken by each ring.

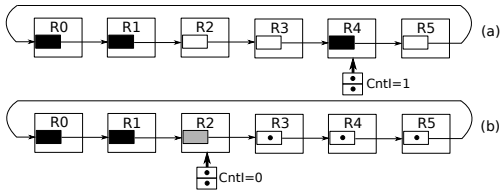

Fig. 3. Worm Bubble Flow Control (WBFC)

Black & White Bubbles: Consider the example in Figure 3 with edge buffer routers. Suppose each bubble indicates an empty input buffer in the downstream router and is denoted by *worm-bubble* or simply *bubble*. Black means a marked *worm-bubble* and white means an unmarked *worm-bubble*. Let *PktS_WB = M/x* denote packet size in terms of worm-bubbles, where M = packet-size and x = worm-bubble size.

Injection: Before insertion into a ring x, a new packet first marks the *worm-bubble* of the corresponding ring as black. This is shown in Fig 3a), where a packet at R4 is trying to get injected. It marks the corresponding bubble as black, and also maintains a count of the marked bubbles, shown by *CntI = 1*. Forward movement of flits displaces the bubble along with their color backwards. Hence the black bubble at R4 will be pushed backwards to R3, leading to the reappearance of a white bubble at R4. The packet trying to get injected can again mark this bubble as black, further incrementing its count. Now consider the case when *PktS_WB-1* worm-bubbles have already been marked for the ring, (i.e., $CntI \geq PktS\_WB - 1$), and it encounters an empty white bubble. If the packet is injected now, it will occupy the space of bubbles marked by itself and the current white bubble. Hence, guaranteeing that any black worm-bubbles injected in the ring during initialization, will remain intact. This is the key idea in WBFC that has been used in our Bubble Sharing scheme. Central buffers, as explained earlier, provide a shared pool of these bubbles, instead of having 1 bubble for each ring, per router. Hence, instead of reserving a black bubble every cycle, for a particular ring, multiple black bubbles can be allocated simultaneously. A count called $WhiteBubbleCnt$, is maintained to keep track of the unoccupied white bubbles. Hence, if $WhiteBubbleCnt + CntI \geq PktS\_WB$ and there is an empty white bubble, injection can happen directly. It should be noted that during injection, $CntI$ is passed to the head flit of the packet as done in the original WBFC. The rules for injection are given by label 4 & 6 in algorithm 1.

Transit: In the original WBFC, the marked bubbles are unmarked when the packet is moving through the ring, decrementing $CntH$. In case of Bubble Sharing, multiple black bubbles for a particular ring are unmarked simultaneously at a particular router. This quickly reduces the amount of black bubbles in the ring, leading to significant reduction in injection delays as compared to WBFC. The rule is given by label 11 & 12 in algorithm 1.

Ejection: If the packet does not encounter an equal amount of marked bubbles, the remaining count is passed to the ejecting router, which means that this router has already injected few black bubbles into the corresponding ring, This rule, (label 10 in algorithm 1), remains the same in both WBFC and Bubble Sharing.

Grey Bubble: If multiple ports are marking bubbles in the ring simultaneously, it is possible that all the bubbles in the ring are marked without any packet being injected. This can lead to starvation. A grey bubble was introduced in original WBFC that allows packets to be injected even if the input port has not marked enough bubbles. An example of this is shown in Fig 3b), which illustrates a case when all bubbles are either occupied, or have been marked as black except for one grey bubble. Since the packet encounters a grey worm-bubble, it will be injected. The details of why grey bubbles work can be found in [1]. Bubble Sharing scheme uses the grey

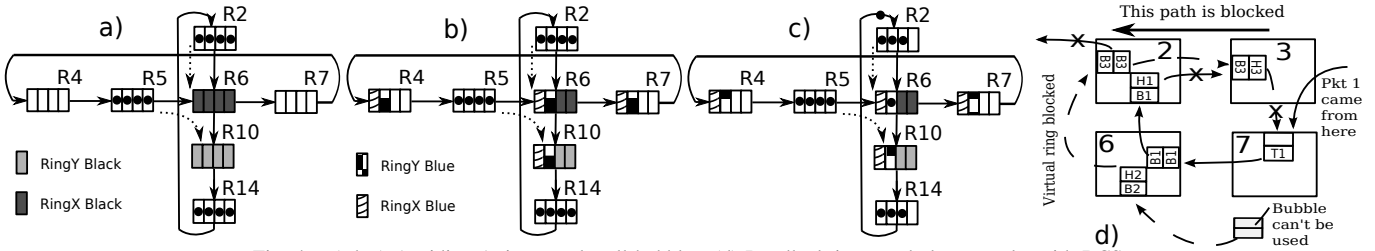

Fig. 4. (a,b,c) Avoiding 1 ring to take all bubbles. (d) Deadlock in wormhole networks with BCS

bubble rules without any modifications (label 5, 7, 8). It should be noted that the grey bubble rules require $PktS\_WB$ black and one grey bubble at initialization (label 1, algorithm 1). Furthermore, progressive movement of grey worm-bubble is necessary to ensure progress. Details can be found in [1].

Init: **for** *each ring* **do**
1 Insert $PktS\_WB$ BLACK & 1 GREY bubble;
2 For each router, assign 1 bubble as BLUE;
3 WhiteBubbleCnt = non-assigned worm-bubbles;
end
Injection:
4 **if** $WhiteBubbleCnt > 0$ *and* $CntI + WhiteBubbleCnt \geq PktS\_WB$ **then**
assignColor(BLACK); // till CntI == PktS_WB-1;
HF.CntH = CntI; CntI = 0;
5 **else if** $isColor(GREY)$ *and* $CntI \geq 0$ **then**
HF.color = GREY; color = WHITE;
HF.CntH = CntI; CntI = 0;
6 **else if** $WhiteBubbleCnt > 0$ *and* $CntI < PktS\_WB - 1$ **then**
assignColor(BLACK); // till CntI == PktS_WB-1;
don't Inject;
end
Ejection:
7 **if** $HF.color == GREY$ **then**
8 **if** $WhiteBubbleCnt > 0$ **then**
assignColor(GREY);
9 **else**
turnBlueToGrey;
end
end
10 CntI=HF.CntH; HF.CntH = 0; HF.color=WHITE;
Transit:
11 **if** $isColor(BLACK)$ *and* $HF.CntH > 0$ **then**
removeColor(BLACK); // as much as possible
12 **else if** $isColor(BLACK)$ *and* $HF.CntH == 0$ **then**
bkwdDispl(BLACK);
end
CntI_Logic:
13 **if** $CntI > PktS\_WB - 1$ **then**
bkwdDispl(CNTI); CntI–;
end

Algorithm 1: Bubble Sharing Flow Control

Blue Bubbles: It is possible that one ring takes all the white bubbles and starves others. Moreover, the movement of this ring can be dependent on movement of other rings, for example in xy routing, movement of rings in the x-dimension are dependent upon successful movement of rings in the y-dimension. If the ring x takes all bubbles at a particular router, ring y may not be able to move through that router, which will stop ring x from moving as well, causing deadlock. The example in Fig 4a) explains the situation. A packet in R2 in ringY wants to move through R6, but all bubbles of R6 are taken by ringX. Thus ringY cannot move, although it has sufficient bubbles in R10. Since ringY cannot move, a packet at R5 in ringX cannot make progress as well.

The problem can be solved by introducing a blue bubble for each router in each ring. This bubble acts as a normal white bubble assigned to that ring, but as a black bubble assigned to a different ring for all other rings. This ensures that at least 1 worm-bubble slot is kept in each router for each ring, thus allowing that ring to always make progress. In the previous example, Fig 4b) shows blue bubbles for both rings. The blue bubble of ringY will allow flits waiting at R2 to move forward into R6 consuming the blue bubble (Fig 4c). However, as soon as the flit leaves the router, the blue bubble for that ring is reclaimed, ensuring forward movement all the time. It should be noted that to guarantee progressive movement of a grey bubble, the blue bubble should be converted to grey bubble for that ring, in case no other white or black bubble for that ring is left in that router. This also holds true in the case of ejection. The ejection rule is slightly changed to encounter the blue bubble as shown by label 9 in algorithm 1.

Starvation Concern: As explained earlier, if a packet does not encounter enough black bubbles during transit, it passes the remaining count of its marked black bubbles (i.e., $CntH$) to the ejection port. However, for traffic patterns with 1-1 communication, it is very likely that this counter keeps incrementing at a particular ejection point. This means that one router has injected most of the black bubbles in that ring. This condition can lead to starvation of other routers in the ring, which may never inject new black bubbles.

We solved this problem by having a separate backward displacement signal for *CntI*. Every time *CntI* becomes greater than the packet size in terms of worm bubbles, *CntI* is displaced backwards, evening out the black bubbles injected by different routers in the ring (rule 13). Backward displacement of *CntI* means that a router, which has injected too many black bubbles in the ring, is giving those bubbles to its neighboring routers to decrease their injection delay.

B. Adaptive Bubble Sharing

Deadlock avoidance with adaptive routing requires three things to happen: 1) There must always be an escape path from any source to any destination. 2) Packets ejecting the escape path must be consumed. 3) All packets are guaranteed to contest for injection into the escape path.

Satisfying Condition 1: The first condition is satisfied by having a virtual ring with guaranteed bubbles, as used in BCS. For networks with centralized buffers, bubble sharing can be used instead of a packet-based critical bubble. A problem, however, is that the virtual ring in BCS allows the use of 180 degree turns. With wormhole networks, this can lead to deadlocks within a packet, that is, a packet going towards its minimal direction takes a 180 degree turn to

enter the escape ring, and then takes another 180 degree turn towards its minimal direction, deadlocking itself. We avoided this by having two separate virtual rings going in opposite directions, and prohibit 180 degree turns. Since both escape paths are deadlock free, prohibiting one does not break deadlock avoidance guarantees provided by the other.

Satisfying Condition 2: The second condition is similar to what is generally called the consumption assumption. However, since packets can leave the virtual ring without reaching their destinations, it is possible that the head flit leaves the ring and gets stuck in the non-ring channels of the network, leaving body and tail flits in the virtual ring. This situation is explained in Fig 4d) for edge buffer wormhole networks, in which the head H1 of a packet P1 is contesting for the escape path in the virtual ring. However, another packet P3 has occupied it, but cannot make forward progress because the tail T1 of packet P1 is stuck in its path in the virtual ring. The bubbles present somewhere else in the virtual ring cannot be used to remove the deadlock. This condition does not occur in original BCS because, in packet-based networks, when a packet leaves the ring, it always drains, i.e., there are no body and tail flits to get stuck. With central buffers, we can utilize the above mentioned fact by checking a space of *PktS_WB* in the central buffer before ejecting a packet from the ring, ensuring that it will drain completely. Hence, ejection out of the virtual ring is only allowed if $WhiteBubbleCnt > PktS\_WB - 1$

Satisfying Condition 3: The third condition is satisfied in edge buffer routers by allowing head flits to leave only when the input buffer of the downstream router is empty *(credit=input_buffer_size)*. This cannot be used with centralized buffer routers due to the presence of EB links with no credit based flow control. This means that it is possible that all head flits wait behind the tail flits in a cycle, and are not even allowed to contest for the escape path.

The solution requires a guarantee that once a head flit traveling outside the virtual ring leaves the allocation stage, it will reach the head of the downstream input buffer, contesting for the escape path (if required). A simple way to guarantee this is to allow movement only when there is a space of one packet left among the white slots in the central buffer. This will ensure that the current packet drains completely, and the subsequent packet's head reaches the top of the input buffer. However, this will put a significant injection bottleneck in the non-ring channels, especially because the virtual ring will be eager to occupy any available white bubbles. We reserve a pool of bubbles, (let's call them yellow bubbles), specifically for the non-ring channels, and prohibit the virtual ring to take these bubbles. Channels not in the ring are allowed to occupy from the pool of yellow and white bubbles, while channels within the ring can only take white or their own black, blue or grey bubbles. Injection in the non-ring channels is allowed as long as $WhiteBubbleCnt + YellowBubbleCnt > PktS\_WB - 1$. This condition is enough for drainage of packets ejecting the ring as well (condition 2), since they are also injecting into the non-ring channels. The introduction of yellow bubbles will allow packets to take minimal non-ring channels more often, having a significant impact on low load latency.

C. Modifications to Centralized Buffer Router

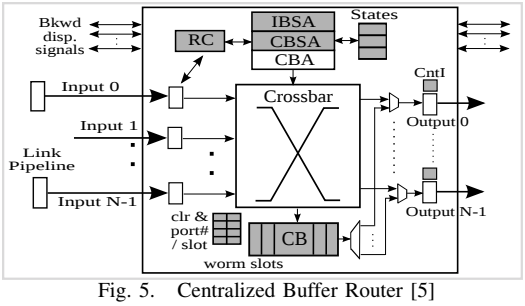

Fig. 5. Centralized Buffer Router [5]

The modifications to CBRs required to implement both deterministic and adaptive bubble sharing is given as shaded regions in Fig 5. As explained earlier, the DAMQ-style central buffer is organized as small worm-bubble sized slots. A $color$ and a $portno$ field is added to each slot to determine the color and the ring assigned to the slot. A logic of few gates is added during the allocation and deallocation of each slot, to determine whether a slot can be assigned a specific color and ring, based on its current color and status. IBSA stage is modified to implement algorithm 1, and rules of adaptive bubble sharing. This, however, works in parallel to the allocation logic, having minimal impact in its critical path. Other modifications like CntI for each ring, CntH in each head flit, progressive movement, and backward displacement signals are added similar to the original WBFC and BCS schemes. Backward displacement hardware is slightly modified to accommodate exchange of CntI signals. Overall, the modifications added a few gates, flip flops, and control signals, with almost negligible impact on the critical path of the router.

D. Worm Bubble Coloring (WBCS)

The idea of adaptive routing with single-VC wormhole networks can also be applied to edge buffer routers. We only need to satisfy the three conditions given above. Condition 1 and 3 can be satisfied by WBFC, and by allowing head flits to leave only when *credit=input_buffer_size*. The problem of drainage from the virtual ring can be solved by providing a small *packet_size - worm_bubble_size* entry central buffer. Ejection from the virtual ring is only allowed when this buffer is empty, and there is no other packet leaving any of the virtual ring. If a packet leaving the ring is detected to be stuck, it is moved to the central buffer to allow subsequent packets to contest for the virtual ring. We call this scheme as Worm-Bubble-Coloring and used it for evaluation purposes.

IV. Results

A. Simulation Setup

The proposed schemes are evaluated with an in-house router micro-architectural simulator. We also developed 3 edge buffer router models for comparison: First using Worm Bubble Coloring, second using WBFC, and third using standard Duato's protocol [4] to avoid deadlocks. Any extra VCs in all edge buffer routers use minimal adaptive routing. We used worm size of 2 flits for central buffers in all our simulations. For

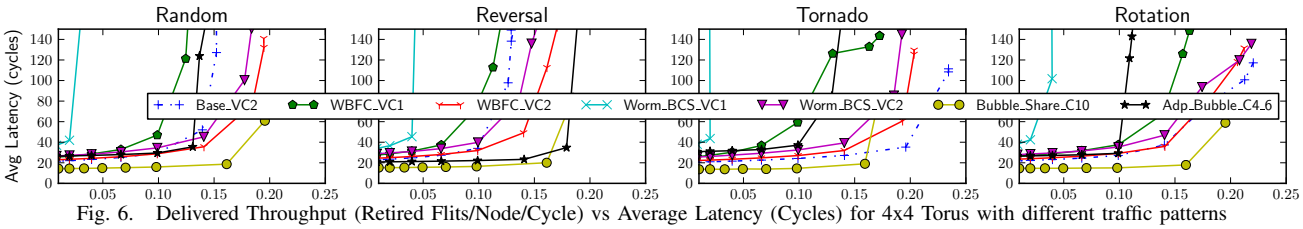

Fig. 6. Delivered Throughput (Retired Flits/Node/Cycle) vs Average Latency (Cycles) for 4x4 Torus with different traffic patterns

adaptive bubble shared routers, the routing logic provides minimal adaptive paths along with the non-minimal escape rings. We prioritize minimal paths over non-minimal ones during allocation. We assume that backward displacement or progressive movement of bubbles can take place in a single cycle regardless of link delays. This can be done by having a separate network for control signals.

Parametric configurations of each of the routers is given in Table I. The table also shows the abbreviations used in the results section for each of the router. Here *_VCx* means edge buffer router with *x* number of VCs. Similarly, *_Cx_y* represents bubble sharing routers with *x* entries reserved for white bubbles, and *y* entries reserved for yellow bubbles. Note that the reserved slots for blue bubbles (1 per ring) are additional from the ones given by *_Cx_y*. We have assumed 1-message class and single flit output buffers, with 128 bit wide links for all the routers. Injection & ejection queue size is kept to be 20 flits for all cases.

Router	Abbreviation	InBuff	CBuff
Baseline	Base_VCx	2*x	0
Worm Bubble Flow Control	WBFC_VCx	2*x	0
Adaptivity with Edge Buffers	Worm_BCS_VCx	2*x	4
Bubble Sharing	Bubble_Share_Cx	1	x+8
Adaptive Bubble Sharing	Adp_Bubble_Cx_y	1	x+y+4

TABLE I
SYSTEM CONFIGURATIONS OF VARIOUS ROUTERS

We model 4 different topologies, namely 4x4 and 8x8 torus, and 4x4 and 8x8 generalized hyper-cube (GHC), representing various degrees of high and low radix routers. The torus networks have single cycle link delays between subsequent routers. The GHC models multi-cycle links, equal to the number of routers between the source and destination, that is, the link delay for node 2 and node 3 from node 0 in x-dimension will be two and three cycles, respectively.

Four different synthetic traffic patterns, (random, tornado, bit-reversal, and bit-rotation), are used. The packet size for synthetic traffic is kept to be 6 flits. All simulations are done for 10 million cycles. Application traces are taken by running 64 threaded version of PARSEC and SPLASH benchmarks with 64 cores, 16 MC configuration, using an in-house simulator, with DRAMSim2 [13] as the main memory model. The traces are generated at the back side of the L1 cache, and messages are classified into read/write/coherence type requests. A reply of 6 flits is generated from the destination for all read requests. Read requests and coherent messages consists of 2 flits, and write messages are 6 flits. This allows us to test our scheme for variable size packets as well.

For area & power modeling, Orion 2.0 [16] is used. We modified Orion to accurately model centralized buffer routers (CBRs). As a conservative estimate, EB links are modeled to take 3x more leakage power and 3x more area in routing logic than non-EB links. Since CBRs have 3 allocation stages, they take 3x more area and power in arbiters. Furthermore, they have an additional crossbar output port going to the central buffer, increasing their area. The extra injection & ejection logic used for WBFC takes few gates and is ignored. Similarly, power taken by extra backward displacement signals, and extra injection and ejection logic are assumed to be negligible. The network is modeled to be running at 2GHz with Vdd = 1.0V and 45nm technology. Activity for different components such as crossbar, input and, output buffers, etc., are taken directly from performance simulations, and fed as activity of different components to Orion.

B. Performance with Synthetic Traffic

2D Torus Topologies: Figure 6 compares throughput and average packet latency of various schemes in a 4x4 torus network. The performance of single-VC WBFC and single-VC Worm-BCS is low. This is because of the injection limitations required to enter a ring. However, with 2VCs, the performance increase significantly, more than the baseline 2VC router for most traffic patterns. The reason lies in the availability of more non-ring channels that does not suffer the injection limitation. Bubble Sharing gives the best result both in terms of low load latency and saturation throughput. Reduction in low-load latency is attributed to elastic links and single cycle pipeline delay in the router. Furthermore, having multiple bubbles per ring reduces the injection delay, which results in improved throughput as compared to WBFC. Adaptive bubble sharing suffers performance loss, both in terms of no-load latency and saturation throughput, because of the ejection limitations discussed in section III-B. Since, there are a few number of non-ring channels, ejection limitation does not allow packets that have already entered the ring to leave it, limiting the throughput equal to the throughput of the virtual ring. However, the minimum number of buffers required is extremely low, even with respect to bubble shared router. This is explained in section IV-C. 8x8 Torus topology shows similar results (not shown due to space constraints).

2D GHC Topologies: Figure 7 shows the same result with a 4x4 GHC. In this case, the ejection limitation is reduced because of the high number of non-ring channels available. Furthermore, prioritizing minimal hops reduces the low-load latency of the adaptive bubble sharing router, as well as its saturation throughput. Both edge buffer routers suffer due to the presence of credit base flow control. Figure 8 compares throughput and average packet latency with an 8x8 GHC. It can be seen that the adaptive bubble scheme surpasses others by a large margin. Again, this is because of the high number of

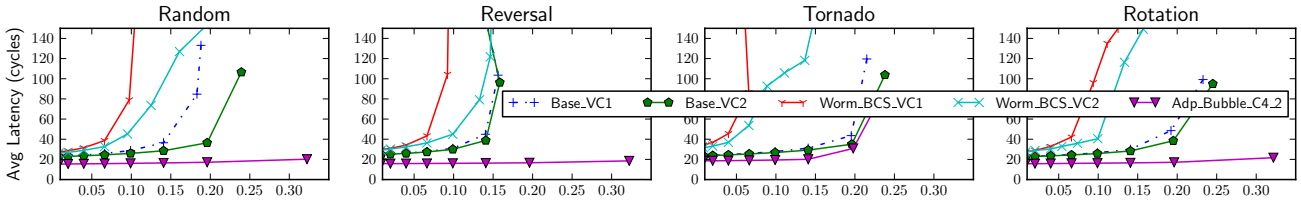

Fig. 7. Delivered Throughput (Retired Flits/Node/Cycle) vs Average Latency (Cycles) for 4x4 GHC

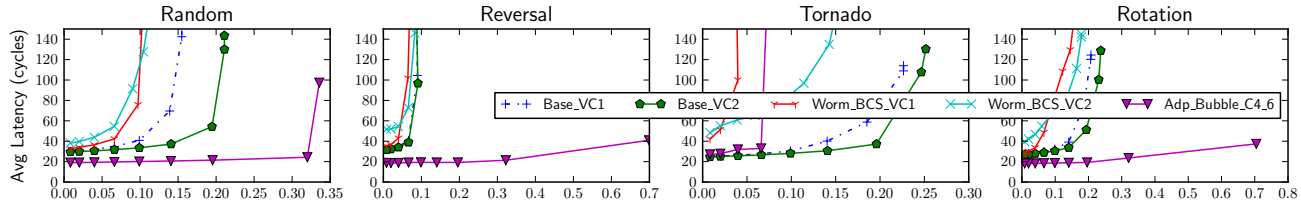

Fig. 8. Delivered Throughput (Retired Flits/Node/Cycle) vs Average Latency (Cycles) for 8x8 GHC

non-ring channels available along with EB links. Furthermore, the buffering requirement per router did not change, even with high number of ports. Extremely low no-load latency is also achieved due to the presence of EB links, priority for minimal hops, and low hop count.

We conclude this section by making the following remarks. For low radix networks, adaptivity using bubbles does not help. In such a case, deterministic bubble sharing scheme performs the best with reduced buffer space due to sharing. However, with high radix topologies that have many routes to destination, adaptivity provided with bubbles significantly improves performance without increasing the buffer area.

C. Buffer Space Analysis and Impact on Area

Edge buffer routers requires at least 1 worm-sized entry per input port per VC. Worm_BCS also requires a $packet\_size$ - $worm\_size$ central buffer for minimum operation. WBFC requires *PktS_WB Black and 1 Grey* bubble to be injected per ring at initialization. For a 6 flit packet with *bubble_size* of 2, this means four bubbles per ring. The total number of rings in a 4x4 torus are 16, four in each direction. Thus, 4x16=64 bubbles will be injected at initialization. Edge buffer routers use input buffers to provide these bubbles. With central buffers, minimum number of entries required by each router is 64/16=4 bubbles. Assuming that we have at least 1 white bubble per router at initialization, the number of entries required at each router = 5 worm-bubbles or 10 flits. Furthermore, each router requires 1 blue bubble per ring making the minimum central buffer size to be 18 entries for routers with bubble sharing. It should be noted that with an 8x8 torus, this will be reduced to 12 entries per router. In the adaptive bubble sharing case, the 2 virtual rings require 4*2=8 bubbles anywhere in the network, in addition to the 2 blue bubbles per router. This makes the minimum requirement with adaptive bubble sharing to be 6-8 entries per router.

Table II gives the total buffer space requirements of different routers with different configurations. The formula for calculating the buffer space is $[P*(I*VC+O)+C]*L$, where L is the link width, P is the number of ports, I, O, C is number of flits in input, output and central buffers, respectively. For 2D-ring topologies, the 2VC baseline and WBFC routers require 400 bytes per router. Worm_BCS requires slightly more with a small central buffer. Since, $worm\_size$ is only 2 flits, and we have 1 flit staging buffer in the centralized buffer routers, the total buffer space in bubble sharing router with central buffer entries of 20 is high. However, the buffer size increases very slowly with increase in the number of ports and gets extremely low, e.g., for 8x8 GHC. The adaptive bubble sharing router will always have smaller buffering requirement than others because of its small entry central buffer. Furthermore, its throughput with large number of adaptive channels is high, making it an ideal candidate for high-radix routers.

RowNo	Parameter	2D-Torus	4x4-GHC	8x8-GHC
1	**Baseline_VC2**	400	560	1200
2	**WBFC_VC2**	400	560	1200
3	**Worm_BCS_VC2**	464	624	1264
4	**Bubble_Share_C10**	448	512	768
5	**Bubble_Share_C12**	480	544	800
6	**Adp_Bubble_C4_2**	320	384	640
7	**Adp_Bubble_C4_6**	384	480	704

TABLE II
BUFFER SPACE PER ROUTER (BYTES) WITH DIFFERENT CONFIGURATIONS

Fig 9a) gives the area distribution of a 4x4 Torus and an 8x8 GHC router with different configurations. Baseline has a large area in the input buffer for both 4x4 and 8x8 routers even with a single message class. The input buffer area for bubble sharing routers is low. However, central buffer takes a significant amount of area as well. In 2D-Torus, this area dominates over other parts making bubble shared routers 3% more expensive than the baseline 2VC router. However, with adaptive bubble sharing and small buffers, the area decreases by 27% making it the cheapest. Furthermore, with high radix, such as in GHC, the increase in central buffer area is very low, as compared to area requirements of multi-VC input buffers and crossbars, thus reducing the area of adaptive bubble router by 46% and 52% compared to the Base_VC2 and Worm-BCS_VC2 routers, respectively. Note that the crossbars are configured as $input\_channels$ X $output\_ports$, that makes their area larger depending upon the number of VCs. Since centralized buffer routers do not have VCs, their crossbar area does not increase as significantly as edge buffer routers.

D. Power Analysis

Figure 9b) compares the static and dynamic power dissipation of different routers at low loads configured in a 4x4 torus and 8x8 GHC topology. At low loads, most of the router power

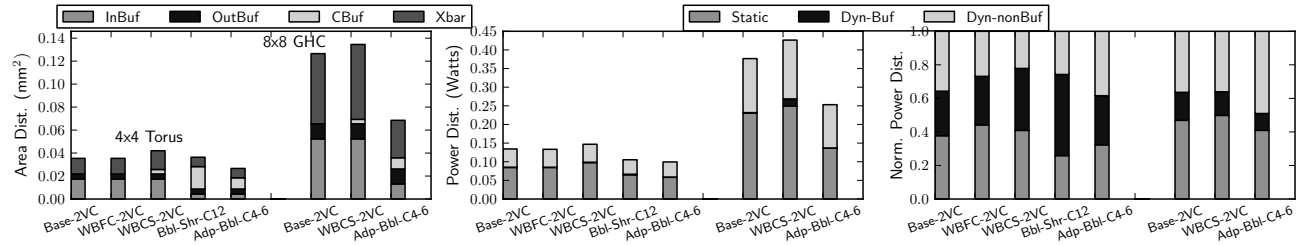

Fig. 9. Router Area & Power with different configurations a) Area Distribution b) Low Load Power Distribution c) High Load Power Distribution

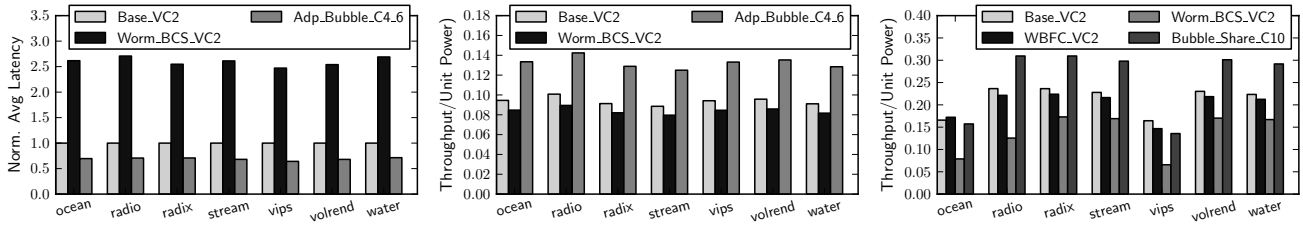

Fig. 10. Real Application Results. (a) Normalized average packet latency for 8x8GHC. Throughput per unit power for (b) 8x8GHC (c) 8x8Torus

is static, with very small dynamic power in the buffers. The static power of the bubble shared router is 24% lower than the baseline in 4x4 torus topology. This is due to the presence of a smaller crossbar (single-VC) in it. Adaptive bubble shared routers further reduces it to 32% & 41% for 4x4 torus and 8x8 GHC, respectively, because of their smaller central buffers.

At high loads, as shown in Figure 9c), buffers take a significant portion of the overall dynamic power, with reduced distribution in links & crossbars. The highest for torus topology is taken by bubble sharing routers because of the largest buffer size. However, the adaptive bubble sharing case, as can be seen with the GHC topology, requires the least amount of dynamic buffer power even with the highest throughput.

E. Results with Real Benchmarks

Fig 10a) gives the average packet latency of various schemes with an 8x8 GHC topology normalized with respect to the baseline 2VC router. As can be seen, adaptive bubble sharing consistently gives lower latency across a range of benchmarks. The percentage improvement on average is 31%. Similar results can be seen in Fig 10b) showing throughput per unit power. Average percentage improvement in this case is 41%. With 8x8 torus (Fig 10c), although adaptive bubble sharing did not perform as good, our bubble sharing scheme outperforms all other schemes with an average improvement in throughput per unit power by 13% and 25% compared to Base_VC2 and WBFC_VC2, respectively. Average packet latency results for an 8x8 torus (not shown) are similar.

V. Conclusions

This paper addresses the buffer space reduction problem in on-chip networks by proposing to use variants of bubble flow control in centralized buffer environment, specially for high radix networks. A wormhole-based version of the bubble coloring scheme is also presented to provide adaptivity without the use of VCs in centralized buffer routers. The routers presented use less buffering, with lower power and improved throughput, as compared to traditional multi-VC edge-buffered routers. Our results indicate an average improvement of 41% in throughput per unit power for PARSEC and SPLASH benchmarks, configured in an 8x8 GHC topology. Future work includes the design of better virtual rings to reduce the latency of non-minimal escape paths and allowing separation of flows for different message classes within the rings.

Acknowledgements

This research was supported in part by the National Science Foundation under grant CNS 0855110 and Sandia National Laboratories. We also acknowledge the detailed and constructive comments of the reviewers.

References

[1] L. Chen and T. M. Pinkston, "Worm-bubble flow control," in *High Performance Computer Architecture (HPCA), 2013*.

[2] L. Chen, R. Wang, and T. Pinkston, "Critical bubble scheme: An efficient implementation of globally aware network flow control," in *Parallel Distributed Processing Symposium (IPDPS), 2011*.

[3] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[4] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks*. Morgan Kaufmann, 2002.

[5] S. M. Hassan and S. Yalamanchili, "Centralized buffer router: A low latency, low power router for high radix nocs," in *NOCS*, 2013.

[6] M. Hayenga, N. Jerger, and M. Lipasti, "Scarab: A single cycle adaptive routing and bufferless network," in *IEEE/ACM Microarchitecture*, 2009.

[7] Y. Ho Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Trans. Parallel Distrib. Syst.*, 2003.

[8] S. Ma, N. E. Jerger, and Z. Wang, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *HPCA*, 2012.

[9] G. Michelogiannakis and W. Dally, "Elastic buffer flow control for on-chip networks," *Computers, IEEE Transactions on*, 2011.

[10] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ACM SIGARCH Computer Architecture News*, 2009.

[11] V. Puente *et al.*, "Adaptive bubble router: a design to improve performance in torus networks," in *ICPP, 1999.*

[12] R. Ramanujam *et al.*, "Design of a high-throughput distributed shared-buffer noc router," in *Networks-on-Chip (NOCS), 2010*.

[13] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*

[14] L. shiuan Peh and W. J. Dally, "Flit-reservation flow control," *IEEE Transactions on Parallel and Distributed Systems*, 2000.

[15] Y. Tamir and G. Frazier, "Dynamically-allocated multi-queue buffers for vlsi communication switches," *Computers, IEEE Transactions on*, 1992.

[16] H.-S. Wang *et al.*, "Orion: a power-performance simulator for interconnection networks," in *MICRO 35*, 2002.

[17] R. Wang, L. Chen, and T. M. Pinkston, "Bubble coloring: Avoiding routing- and protocol-induced deadlocks with minimal virtual channel requirement," in *International Conference on Supercomputing*, 2013.