Centralized Buffer Router with Elastic Links and Bubble Flow Control

Syed Minhaj Hassan
Georgia Institute of Technology
Email: minhaj@gatech.edu

Sudhakar Yalamanchili
Georgia Institute of Technology
Email: sudha@ece.gatech.edu

***Abstract*—While router buffers have been used as performance multipliers, they are also major consumers of area and power in on-chip networks. In this paper, we propose centralized elastic bubble router - a router micro-architecture based on the use of centralized buffers (CB) with elastic buffered (EB) links. At low loads, the CB is power gated, bypassed, and optimized to produce single cycle operation. A novel extension to bubble flow control enables routing deadlock and message dependent deadlock to be avoided with the same mechanism having constant buffer size per router independent of the number of message types. This solution enables end-to-end latency reduction via high radix switches with low overall buffer requirements. Comparisons made with other low latency routers across different topologies show consistent performance improvement, for example 26% improvement in no load latency of a 2D Mesh and 4X improvement in saturation throughput in a 2D-Generalized Hypercube.**

I. Introduction

A general design goal of an on-chip processor-memory network is to provide low latency, low power communication. Since wires are present in abundance, the networks are not bandwidth limited. This is fundamentally different from off-chip networks where efforts seek to improve network throughput under a constraint wiring density. In those cases, abundant wiring is typically utilized by increasing the router radix and reducing the network hop count per request, resulting in reduced latency. The problem in using the increased number of wires to reduce latency in on-chip networks is the large amount of buffering associated with high radix router. The minimum depth of each of the buffers on a router port has to be equal to the credit round-trip latency (to avoid bubbles between successive flits and maximize link utilization). This size grows with link length. Furthermore, buffer utilizations are typically low due to routing constraints, number of single flit packets (e.g., in coherent shared memory processors), and bursty behavior of traffic. This results in an over provisioned network with large buffer space that is underutilized. Buffers consume a significant amount of static power and area in on-chip networks degrading the energy efficiencies. Increasing router radix to reduce latency amplifies buffer needs and exacerbates energy inefficiency. The pressure has been towards low radix routers with focus on reducing the latency within the router, e.g., speculation [18], bypassing [5], express channels [13], etc.

In this paper, we propose the use of a centralized buffer (CB) in on-chip wormhole routers to decouple the required buffer space per router from its radix. Furthermore, we propose to use this router in conjunction with Elastic Buffers (EB) on the links [16]. The CB is power gated so that at low loads traffic bypasses the CB and it operates as a single cycle router, while at higher loads it operates as a buffered router. A novel extension to bubble flow control is used to realize deadlock freedom. The *same* mechanism avoids both routing deadlock as well as message dependent deadlock using a constant CB size per router independent of the number of message types. The result is a compact, energy and area efficient physical channel router whose low load performance approaches that of buffer-less routers and high load performance approaches that of buffered routers.

This paper makes the following contributions.

1) Propose a new energy-efficient router architecture with
   - a centralized buffer (CB) and elastic buffer (EB) links
   - optimizations to produce single cycle operation at low loads
   - load dependent power gating of the CB
2) Provide an efficient deadlock freedom mechanism that realizes both routing deadlock and message dependent deadlock with a fixed buffer size independent of the number of message types

The remainder of the paper is organized as follows. Section II provides a brief background. Section III gives the detailed discussion of our router micro-architecture and the associated optimizations. Finally, section IV compares power and performance of our router with different state of the art low latency routers for various topologies.

II. Background & Motivation

A. The Problem

Traditional virtual channel based routers use buffers for deadlock freedom and performance optimization. While total storage is optimized for performance, actual buffer occupancies can be very low. The critical importance of energy efficiency has motivated several approaches to reduce buffering requirements with minimal compromises in performance.

Buffer-less routers [17] represent one such approach. However, as load increases these routers increase both network traffic and average packet latency as packets are routed to non-minimal directions. Also, they are fundamentally throughput limited designs. This is because congestion at any node propagates quickly in these networks causing other packets to stall or take non-minimal routes. Elastic buffer networks [16] have been proposed which retain the minimal buffering requirement without the use of deflection routing. However, EB networks face obstacles in integration with standard performance-optimized router architectures. For example, virtual channels cannot be integrated in the normal manner, and multiple physical channels are recommended for routing and message dependent deadlock avoidance, further increasing pressure on router radix and hence buffering. Ideally, we would like to make use of the advantages of both buffer-less routers and EB links in a power efficient way. Section III describes our approach towards that goal, but first we need to define some router basics.

B. Basic Router Pipeline Optimizations

A typical VC-based input-buffered on-chip router consists of 6 pipeline stages [5]. Input buffering (IB), route computation (RC), virtual channel allocation (VCA), switch allocation (SA) and switch traversal (ST) along with link traversal (LT) cycles. The head flit has to travel all 6 stages. The body flit can skip the RC and VCA stage. The IB and LT stages can be reduced to single cycle specially for the case of single VC routers. Similarly, SA and VCA stages can be combined for single VC designs or in the case of speculative routers [18]. Lookahead routing [6] has been used to perform RC in parallel with SA and VC. The basic idea is to perform the route computation logic in the previous router and send that information along the control path. Thus, the router pipelines have been reduced to 2 stages only, RC/VCA/SA and ST, along with combined LT/IB stage [18]. Note that LT can be of multiple cycles based on length of the link. We proposed a router micro-architecture that provides single cycle operation in the common case and 3 cycle operation in the uncommon case.

C. Elastic Buffer Channels and Bubble Flow Control

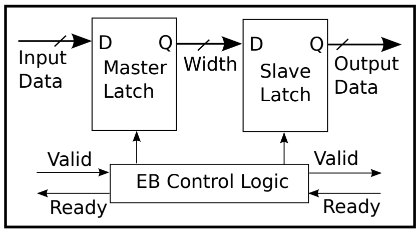

Fig. 1. An Elastic Buffer (From [16])

An elastic buffer (Figure 1) adds simple control logic to the master-slave latches of a D flip-flop to make them 2 independent storage locations (2 slot FIFO). EBs use a ready-valid handshake to move a flit forward. Pipeline bubbles created with ready-valid handshake are avoided by providing 2 slots per EB within a single clock cycle delay. A channel consisting of multiple such pipeline buffers, instead of repeaters, is called an elastic buffer channel that makes it similar to a distributed FIFO. Elastic buffered links are used to provide link level flow control. A fundamental problem with elastic buffer flow control is that they face challenges in providing multiple virtual channels. The main reason of this is pipelined EB links (Although, we can have multiple virtual channel buffers in the router, link pipelining without the knowledge of buffer space in the next router creates dependencies among flits of different VCs within the link. Thus, we must pursue deadlock freedom by other means. This is achieved, in this paper by proposing a novel extension to bubble flow control.

Bubble flow control [19] avoids deadlocks in a packet-based ring by ensuring that at least one empty buffer exists in the ring, so that every packet in that dimension can (eventually) make progress. In a multidimensional tori, any packet entering the ring in a new dimension must not violate this property. A simple way to ensure this locally is to permit injection into the ring (or dimension traversal) only if at least two empty packet buffers are available. Clock cycles that span multiple dimensions are avoided by dimension order or turn-model [8] based routing. A fundamental problem, however, with bubble flow control is that it has been proposed with packet based SAF and VCT schemes. For these schemes, the worst case bubble requirement for packet insertion is 2 packets per port per node, which is very high for small buffers in on-chip networks. We extend this approach to the flit level and CB router to be able to utilize EB links with reduced bubble penalty.

III. Centralized Elastic Bubble Router

In this paper, we propose the use of a centralized buffer (CB) in on-chip wormhole routers with EB links. There is minimal buffering at the inputs/outputs. At low loads, the CB is power-gated off, and the packets bypass the router taking 2 cycle bypass path. At high loads, the flits are streamed through the router taking 4 cycles (buffered path). We further proposed lookahead switch allocation which reduces these paths to 1 cycle and 3 cycles only. This section describes the router micro-architecture in detail and the novel deadlock freedom application of bubble flow control that can support both routing and message-dependent deadlocks.

A. The Baseline Centralized Elastic Bubble Router

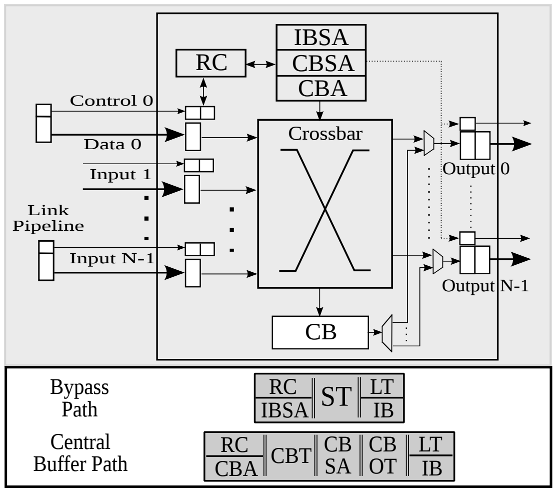

Fig. 2. Router (a) Micro-architecture (b) Pipeline Stages

Router Micro-architecture: Figure 2(a) shows the internal router design. It consists of a large crossbar with single flit input and 2-flit output staging-buffer per port. It also consists of a centralized buffer (CB), which is only utilized when a packet from input buffers cannot make progress to the corresponding output buffers. The control and data information are split in the links. This separation will be explained in section III-B. Central buffer allocator (CBA) performs allocation to the central buffer, while input buffer switch allocator (IBSA) and central buffer switch allocator (CBSA) performs output port allocation for input buffers and central buffers, respectively.

The central buffer is a DAMQ [23] style multi-ported output queue in which flits destined to different outputs are kept separate from each other. It can be considered as multiple output queues (1 per port) which share each others space. We kept the number of read and write ports of the CB to 1. This means that if 2 or more flits need to travel in or out of the CB in a single cycle, they need to be serialized. This serialization is achieved by the corresponding switch allocation stages. The performance overhead of this serialization is small (CBs do not require very high throughput), however, the power reduction by reducing the number of ports is significant. Furthermore, having a single port at the input of the CB keeps the number of output ports of the crossbar in check. The new crossbar is $Inports \times (Outports + 1)$ switch with small area and power dissipation.

Pipeline Stages: Figure 2(b) illustrates the different pipeline stages of the baseline CEB router. A flit or packet entering the input buffer can take 2 different paths in the router. 1) Bypass Path - A flit traversing this path encounters 2 stages within the router. IBSA (the switch allocation stage for flits in the input buffer) and ST stage. 2) Central Buffered Path - A flit traversing this path will encounter 4 pipeline stages within the router. Allocation (CBA) and traversal (CBT) to the central buffer, and allocation (CBSA) and traversal (CBOT) from the central buffer to the output port. At low load, path 1 will be chosen. If the corresponding output port is busy servicing another packet from a different input port, path 2 will be selected. Since flits within a packet need to arrive in order, if a path is chosen for head flit of a packet, all subsequent body and tail flits will follow the same path. Furthermore, since interleaving of flits of different packets is not allowed, once an output port is picked by either CB or any of the IBs, it is not released until the whole packet is traversed.

Every cycle, 3 allocation operations (IBSA, CBSA and CBA) are performed simultaneously. The IBSA tries to allocate a flit at an input buffer to the output port, and if granted set the necessary crossbar and mux signals. The CBSA in the mean time will try to allocate a flit in the central buffer to the corresponding output port. Among the 2 allocations, CBSA is given higher priority, since these packets arrived earlier than the packet in IB stage. In parallel to these allocations, CBA will also try to allocate central buffer space to packets in the input buffers (1 packet per output port at a time). A packet will be allocated to a CB only if the CB has enough space to hold the whole packet. However, if IBSA wins in allocation, CBA will be ignored. Based on which allocation wins, 1 or 2 of the 3 traversals will be performed in the next cycle.

B. Lookahead SA

Baseline CEB encapsulates an EB router reducing the input buffering requirements. However, since allocation and traversal are 2 different stages, the minimal buffering requirement is 2 flits for 100% utilization. We further reduce the input buffers to single flit by performing the switch or CB allocation (IBSA/CBA) in parallel with the last LT/IB stage. This will also reduce the latency within the router to 1 cycle. Performing allocation in parallel with IB is achieved by separating the data and control information of a single flit and sending the control information 1 cycle ahead of the data. Note that lookahead routing decides the output port of the next router in the previous one and sends this information along the data (other control information includes flit type, etc). If we can forward this information one cycle ahead of the actual data e.g., during the ST cycle, it will arrive at the downstream router earlier, allowing it to contest for allocation 1 cycle earlier. Thus, when a flit reaches its downstream input buffer, it will perform the ST or CBT stages immediately in the next cycle without waiting for the IBSA or CBA stage to complete. Since route computation can also be done in parallel with allocation, we can again send the next router output port information during the ST stage i.e., one cycle ahead. Note that this is different from prediction router [14], as allocations are deterministic and not predictive. Also, note that the flit control information is already sent out-of-band in most on-chip routers. Even if it is sent in-band, it can be sent with the flit information of the previous flit. Thus, there is no extra wiring overhead of this scheme. We will assume out-of-band control information in this paper. A problem with lookahead SA is to guarantee that the control information always arrive 1 and only 1 cycle ahead of the corresponding data. This is a necessary condition because if SA wins earlier than the actual data arrival in the input buffer, random data will be propagated forward from that buffer. Note that in general, this is not guaranteed because data and control can get misaligned along the pipeline.

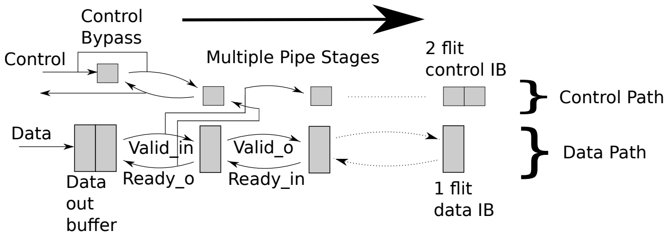

Fig. 3. Guaranteeing 1 cycle ahead - Ready Valid handshake signals of data & Control path

Guaranteeing 1 cycle lookahead: We achieved this goal by utilizing the ready-valid handshake signal of the previous pipeline stage in the data path to traverse the next pipeline stage in the control path. This can be seen from Figure 3. The ready_out signal of the data path is also routed to the ready_in signal of the same pipeline stage in the control path. Similarly, the valid_in signal of this pipeline stage in the data path will be sent to valid_in of the next pipeline stage in the control path. This will ensure that once the control information is in 1st pipeline stage of the link and the corresponding data is next to leave the output buffer, both will progress across the link, with control information always moving 1 cycle ahead. However, since the data output buffer can be of multiple flits, ensuring this condition itself requires small control output buffer as shown in Figure 3. When the output buffer is empty, the control information is directly sent to the first control pipeline stage. If the output buffer is not empty, the control information is sent to the control output buffer. The control output buffer, which is 1 flit smaller than the data output buffer, keeps sending control information corresponding to the second last flit of the data buffer to the first pipeline stage of the link in synchronization with the last flit of the data buffer, thus ensuring that control information always remain 1 and only 1 cycle ahead.

At the input side of the downstream router, it is possible that an allocation operation is not successful. Since the data in the last pipe stage of the link sees an empty slot in the input data buffer, it will move forward, resulting in the flit behind it in the link to move forward as well. The corresponding control information also needs to progress to the input control buffer, which already contains the control information of the mis-allocated flit. This means that it is necessary for the input control buffer to be of size 1 flit larger than the data flit buffer. In the case, a control flit does not win allocation, it should still move ahead in its buffer, allowing the control information of the flits behind it to move forward as well. When the input data buffer is full, the flit in the last pipeline stage of the link cannot progress as the ready_in signal will not be asserted, the control path will stop as well. As long as the flit remain stuck in the input buffer, the corresponding control flit resides in its buffer. As soon as the data flit leaves, its corresponding control information also expires.

C. Deadlock Avoidance

Routing and message dependent deadlock avoidance is achieved by extending the bubble flow control technique to flit level using central buffers. This technique keeps the routing minimal thus ensuring minimum no load latency as well. Before explaining our scheme, we like to reiterate 3 conditions that are required for bubble flow control to work. 1) Every ring or cycle must have a bubble 2) If there is a bubble in the ring, packets within the ring cannot wait indefinitely on any other condition within the ring i.e. they have to make progress. 3) External packets entering the ring are not allowed to destroy the bubble.

Avoiding Routing Deadlock: The idea of avoiding routing deadlock is simple. For every ring, even having a single flit bubble is enough to ensure forward progress of flits. For flits entering the ring, we need to ensure that the whole packet will be allowed to enter the ring along with maintaining the original bubble of the ring. This is a necessary condition because of the following reason. If the whole packet is not allowed to enter the ring, even having a multi flit bubble in the ring, e.g., in the input buffer, will not guarantee forward progress, i.e., condition 2 above will not be satisfied. This means that a bubble of packet length+1 is required when changing dimensions to ensure deadlock freedom. This bubble can be provided with the output based CBs without increasing the size of input and output buffers. Furthermore, since packets in the central buffer of the current router are part of the overall ring (corresponding to that output port), looking at the space of the next router's CB (which will require CB credit information to flow upstream) is not required. This is because if there is enough space in current router's CB, it is guaranteed that flits from the previous routers will move forward to create at least an equal amount of space in the next router. Thus the condition to avoid deadlock only requires looking at empty buffer space of itself and no credit information of the downstream router is needed which makes our technique perfectly suitable for EB-based channels. A formal proof is given below.

Definitions: These definitions are derived from [3], [4]. Let Q be the set of input, output, and link pipeline buffers associated with the routers. For each $q_i \in Q$, let $cap(q_i)$ be its maximum capacity in flits and $size(q_i)$ be its current occupancy. A bubble of X flits in q_i means that $size(q_i) <= cap(q_i) - X$. Let Q_y be a subset of Q consisting of all input, output, and link pipeline buffers that belong to a ring y. Let $q_i \rightarrow q_j$ defines the case when a flit form q_i moves to q_j. To avoid interleaving of flits of different packets, if a head flit from buffer q_i moves to q_j, q_i will hold q_j until the whole packet is transferred, i.e. no other buffer q_k can move any flit to q_j. Let L be the size of all the packets.

Rule 1: A unidirectional ring y is deadlock free as long as there exists a single flit bubble in Q_y, i.e.,

$$\exists q_i \in Q_y : size(q_i) <= cap(q_i) - 1. \tag{1}$$

Proof: The rule is a direct consequence of Theorem 1 in [3]. In that paper, the minimum bubble size is equal to the size of input queues. However, this is a necessary condition for adaptive routing schemes, where head flit cannot make progress if the downstream input queue is not free. Since, we are only dealing with deterministic routing, this condition gets relaxed to single flit in the downstream input queue and hence the bubble size of 1 flit. □

Rule 2: If a head flit of packet a from q_i in the ring x wants to move to q_j in ring y without causing a deadlock, it can do so only if,

$$\sum_{\forall q_k \in Q_y} size(q_k) \leq \left[\sum_{\forall q_k \in Q_y} cap(q_k) \right] - L - 1 \ \Lambda$$
$$\nexists q_m \in Q_x : q_m \rightarrow q_n \in Q_y, \forall Q_x \subset Q. \tag{2}$$

i.e., free space of packet length+1 is available in the ring and there is no other flit of packet b entering the ring y at the same time.

Proof: Multiple cases exist. (i) One monolithic bubble of length $L+1$. In this case, this bubble will flow back such that q_j will get a bubble. Head flit from q_i will move to q_j in ring y and the remaining bubble length will be L. Since no other head flit or packet can enter ring y (bubble length is smaller), only flits of packet a can enter the ring. $\sum_{\forall q_k \in Q_y} size(q_k)$ after all flits of a have entered the ring is $\left[\sum_{\forall q_k \in Q_y} cap(q_k)\right] - L - 1 + L$, i.e., bubble length of 1 will still be there. Rule 1 above will hold and allow ring y to be deadlock free. (ii) If multiple smaller bubbles are available with aggregate equal or more than $L+1$. This means that q_i sees a bubble of length smaller than $L+1$, lets say l. However, at least one more bubble exists in the ring apart from the bubble in q_j, i.e., $\exists q_m \in Q_y \backslash q_j : size(q_m) <= cap(q_m) - 1$. Rule 1 implies that flits in upstream buffer q_n will move forward to q_m creating bubbles in q_n. Let q_m denote the new buffer with the bubble. The previous process continues, until $q_n \equiv q_j$. Since the number of such bubbles is $L+1-l$, the resultant monolithic bubble in q_j will be equal to $L+1$. Case (i) above is applied. (iii) Bubble length is less than or equal to L. If it is equal to L, after all flits of a enter ring y, $\sum_{\forall q_k \in Q_y} size(q_k) = \sum_{\forall q_k \in Q_y} cap(q_k)$. No bubble exists in ring y, rule 1 will not be satisfied any more. If the bubble length is less than L, whole packet a is not allowed to enter ring y. Since, incomplete packet traversal means q_i hold q_j, no other buffer q_k will move flits to q_j even if there are bubbles in the ring. Similarly, no other buffer q_m will be able to send to q_k and so on. Thus no flit will be able to move forward resulting in the ring being deadlocked. (iv) If any other packet is allowed to enter ring y, simultaneously with packet a, it will reduce the bubble size resulting in case (iii) above. □

Implementation of rule 2 is difficult, since it requires global information to restrict packets from injection into the ring based on whether any other packet is being injected at the same time. A much simpler local condition is to check the bubble in the central buffer of the local router. Suppose that the central buffer reserves 1 packet for each ring. Let Q_{y2} be the set of reserved packet spaces for ring y in all the central buffers corresponding to nodes in ring y. Thus, Q_{y2} becomes part of the new extended ring y. Let Q_{cy} be the union of Q_y and Q_{y2}. Above 2 rules can be applied to the extended ring.

Rule 3: Rule 2 can be satisfied by the following 3 conditions as well. (i) Look for a space of L in local central buffer corresponding to ring y, (ii) Look for a space of 1 flit in the output buffer of ring y, and, (iii) Not allow any other packet of the local router to enter ring y at the same time.

Proof: The proof is straight forward, since having bubble of $L+1$ in the local router for extended ring y satisfies equation 2 with Q_y being replaced with Q_{cy}. Also, since we are only looking at the local router, not allowing any other packet to enter ring y at the same time is straight forward. This also means other packets can enter ring y in other routers. Case (iv) of the proof of rule 2 will never happen with flits entering the ring in other routers. □

Reiterating the minimum deadlock condition:

$$FreeSpace = \begin{cases} 1 & i = j \\ PktLength + 1 & i \neq j \end{cases}$$

where i, j refer to different dimensions of travel. We have used empty space of $PktLength$ in CB and space of 1 flits in corresponding OB. This condition is checked during allocation of both OBs and CBs i.e. during IBSA and CBA to ensure a bubble is maintained in the ring. Furthermore, checking full packet space is not required during CBSA as the packet has already entered the ring and therefore, same dimension condition will be applied here. This makes the minimum CB buffer size requirement to $2 * dim * PktLength + 1$ flits. In practice, we can reduce the CB size to $2 * dim * (PktLength - 3) + 1$ by leveraging the fact that the 2-flit output and 1-flit input buffer is part of the overall ring. We fixed the size of CB to 18 flits. Starvation is also possible with CBs. We ensured starvation does not occur by round robin allocation of central buffers to each port. We would also like to mention here that this solution is feasible for on-chip networks where packet size is not large. In fact, all bubble flow control techniques except worm-bubble [3] are not good solutions for networks with large packet sizes. Variable packet sizes are allowed as long as each ring keeps a bubble of the maximum packet size.

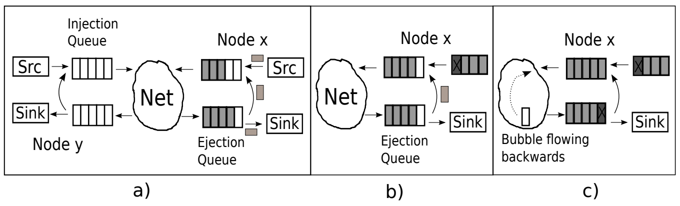

Fig. 4. Message Dependent Deadlock

Avoiding Message Dependent Deadlock: Message dependent deadlocks are usually solved by providing separate virtual networks as explained in [12]. The basic idea is to provide separate virtual network (channels) to every class of a message dependent chain. We propose the use of bubble flow control technique to avoid message dependent deadlocks as well. Note that every reply of a request message (e.g in request reply networks) can be considered a 180 degree turn of the same packet, allowing the possibility of cycles between 2 or more different request-reply pairs. The packet source injecting new requests can be considered as an external entity that inserts new packets in this cycle (Figure 4(a)). These cycles will be deadlock free as long as we ensure that the 3 conditions of deadlock avoidance mentioned earlier are satisfied.

Condition 1 and 3 can be easily satisfied by inserting request messages in the injection queue of the network interface (NI) only when there is a space of at least 2 packets, (Figure 4(a)). Satisfying condition 2 means that if there is only 1 empty space left in the injection queue, the reply message of a request will still be generated, even if there is a new request message pending to get inserted in the injection queue (Figure 4(b)). Thus a packet present within the message cycle (request message in the ejection queue) is allowed to progress to the next buffer (generate a reply message) with space of only 1 packet downstream (injection queue). External packets (pending request messages in the message sources) have to wait. Implementation of this scheme in network terminals means having the ability to accept new request messages and generate the corresponding reply, if there is an empty space in the injection queue. If it has no empty space, (Figure 4(c)), request messages can wait in the ejection queue but since there will be a bubble in the message cycle somewhere, this bubble will always propagate back to the injection queue allowing the request messages to get serviced. Note that this scheme is valid for any number of message classes without adding VCs.

The use of bubble flow control in the preceding manner makes it possible to deal with routing deadlock and message dependent deadlock with the *same* mechanism, e.g., there is no need for additional storage such as separate request and reply networks. In particular, the cost of dealing with message dependent deadlock is fixed independent of the number of message classes. Overall, the cost of deadlock freedom at a router is independent of the network size or the number of message types.

D. Power Gating of CB

Since CBs are utilized only at high loads, we applied a simple coarse grained power gating technique to it. Power gating CB is simplified as it does not interfere with the main path of the router. Deadlock avoidance will be guaranteed as long as it will turn on in some finite amount of time when a packet is blocked. Initially, the CB is kept off. Whenever 2 packets collide for an output port, a counter starts counting the wait cycles of the unallocated packet. When the wait becomes X cycles, the CB is turned on. It takes 3 cycles for the CB to turn on completely. Once on, unallocated packets can be pushed into it allowing the blocked packets to move ahead. When an on CB is empty, and minimum on-time (set to 10 cycles) has passed, it is turned off. At low loads, this simple power gating technique keeps the CB off. At high loads, since we wait for X number of cycles before turning it on, this technique can potentially reduce performance. In fact, the value of X provides the throughput power consumption trade-off. We will explore sensitivity to the value of X in our results section.

IV. Results

A. Simulation Setup

We have developed 4 different router micro-architecture models to understand the latency and throughput impact of our CEB design. The baseline router consists of a standard 2 stage pipeline router with 2VCs per virtual network. The other 2 routers implemented are 2 stage EB router and a simple flit deflection (FD) router similar to Flit BLESS from [17]. Parametric configurations of each of the routers is given in Table I. The DAMQ based central buffer is organized into 6 slots of 3 flits each. Better implementation of the central buffer is left for future work. The default wait time before turning on the power gated CB is 500 cycles. Furthermore, to reduce the latency and buffering requirements of the deflection router, we retire the packets as soon as the tail flit arrives without waiting for head and all body flits to reach the destination. This makes the deflection router very optimistic. Since EB requires duplicate physical channels, we have assumed its links to be half wide with twice number of flits per packet.

Parameter	Baseline	FD	EB	CEB
Pipeline Depth	2	1	2	1
InBuf Size (per port)	5*VC	1	2*Virt. Net	1
OuBuf Size (per port)	2	2	2	2
CBuf Size	na	na	na	18
Inj/Ej Que Size	20	20	20	20

TABLE I
SYSTEM CONFIGURATIONS OF VARIOUS ROUTERS

We have implemented mesh, torus and generalized hypercube (GHC) topologies for both 2D and 3D networks. The 2D networks are 8x8 while 3D networks are 4x4x4. Number of ports for mesh and torus are $2 * dim + 1$ and $(k-1) * dim + 1$ for GHC topologies. Thus 2D-GHC, with $k = 8$ has the highest number of ports and therefore has the highest power consumption. The torus and mesh networks have single cycle link delays between adjacent routers. The GHC models multi-cycle links equal to the number of routers between the source and destination i.e. the link delay for node 2 and 3 from node 0 in the x dimension will be 2 and 3 cycles respectively. The packet size is kept fixed (= 5 flits) except EB routers which are 10 flits as discussed earlier. All links are assumed to be 128 bits wide. All designs except deflection routers use minimal dimension order routing. For torus topologies with single VC and no central buffering, tranc routing from [20] is used which is an up down style non-minimal routing technique that does not use VCs.

4 different synthetic traffic patterns (random, bit complement, bit reversal and tornado) are evaluated. Unless otherwise stated, all results present an average of all 4. Random distributes the traffic evenly and has high throughput. All others are adversarial traffic patterns with relatively low throughput. Tornado travels equal or more than k/2 hops in each dimension and thus has the highest no load latency. All simulations are performed for 50 million cycles. Applications 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 [22] as the main memory model. The traces are generated at the back side of L1 and messages are classified into read/write/coherence type requests. A reply of 5 flits is generated from the destination every time a read request is received. Read requests and coherent messages for all networks including EB consists of 2 flits and write messages are 5 flits except in the EB network in which they are 10 flits wide. This allows us to test our scheme for variable size packets as well.

For power modeling, Orion 2.0 [25] is used which calculates the router power as the sum of the power in its buffers, crossbar, arbiters and allocators along with the link power. We modified Orion to get more accurate results. As a conservative estimate, EB links are modelled to take 3x more device power and 3x more leakage power in routing logic than non-EB links. Similarly the CEB router which has 3 arbiters takes 3x more power in arbiters. VC allocator power is assumed to be negligible for all cases. Segmented crossbars with 2 segments are used. For GHC topologies, partitioned crossbars are used. Baseline and EB routers are assumed to have 2

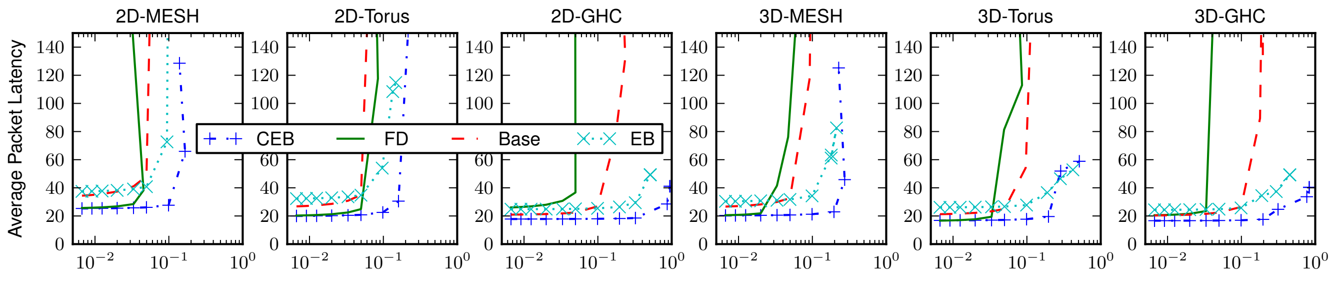

Fig. 5. Throughput (Retired Flits/Node/Cycle) vs Average Latency (Cycles) for different network configurations

message classes, with 2 VCs per message class. FD and CEB do not model any VC or message class. CEB has an additional component of power due to its central buffer. All buffers are assumed to be register based. The network is modeled to be running at 2GHz with Vdd = 1.0V and 45nm technology. Activity for different components such as crossbar and input output buffers etc. are taken directly from performance simulations and fed as activity of different components of Orion. Power Gating a CB is assumed to reduce its leakage to 20% of the original.

B. Performance with Synthetic Traces

Comparison with Other Routers: Figure 5 compares throughput and average packet latency of CEB routers with that of other routers with different network configurations. Note that throughput is defined as retired flits per node per cycle. At low loads, CEB network has the latency equal to that of deflection (FD) router. This is because of the single cycle latency within the router. Both baseline and EB has 2 cycle latency within the router resulting in increased no load latency. It should be noted that pipeline bubbles are avoided in these designs by keeping their buffer sizes larger. Furthermore, EB has higher serialization latency since each link is narrower than the other routers.

Baseline and deflection routers have the lowest saturation throughput. For deflection routers, the greater the number of ports per router, more numerous the deflections are, and thus saturation throughput does not increase with the number of ports. This can be seen in the case of GHC where deflection router saturates quickly compared to others. The baseline router has higher throughput than deflection in most cases but because of extra bubbles created due to credit based flow control, their throughput is low as compared to routers that use elastic links even with multiple VCs. This difference increases with longer links in GHC topologies.

Both EB and CEB have much higher throughput due to the use of elastic links. CEB has higher saturation throughput due to the removal of head of line blocking made possible through the central buffering. However, since travelling to the central buffer increases latency within the router, this is not always true. This can be seen in Figure 6 which shows the same graph of 3D torus but with individual traffic patterns. Note that for Tornado traffic EB performs better than CEB. Since Tornado is an adversarial traffic pattern, it requires larger number of packets to traverse the central buffer and thus increased latency within the router and lower throughput. A similar behavior can

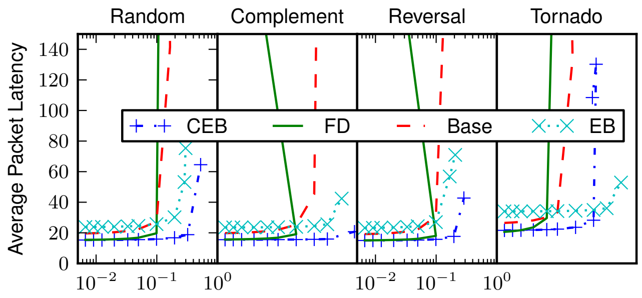

Fig. 6. Throughput (Retired Flits/Node/Cycle) vs Average Latency (Cycles) with different traffic patterns

be seen for the 3D Mesh topology in Figure 5 where the CEB curve starts going backwards. This means very high utilization of CB is also not desirable as it increases the latency within the router.

Impact of Individual Optimizations: CEB uses various optimizations for different purposes e.g. lookahead SA for latency reduction, power gating for power reduction and bubble flow control for deadlock avoidance. We compare the advantages of individual optimizations in Figure 7 for various network topologies. In the figure, NOBUBBLE represents the case without any optimization and no bubble flow control. NOSA adds bubble flow control to the NOBUBBLE case. NOGATE adds lookahead SA to the NOSA case without power gating. It can be seen that NOGATE and GATE cases which have single cycle latency in the router by adding lookahead SA optimization has significantly low no load latency. Their throughput, therefore, is higher in general. The torus topologies with NOBUBBLE have higher no load latency due to non-minimal routing (remember we use tranc routing for these cases). However, the saturation throughput of 3D torus with non-minimal routing is higher which shows the overhead of having bubbles in the network. Note that both NOSA and NOBUBBLE case has 2 flit input buffer as opposed to single flit in other cases. Lastly it should be noted that power gating closely tracks the case with no power gating specially in the case of GHC topologies, thus its performance overhead is low.

Comparison of different topologies The above figures can be also used to compare the results of different topologies for CEB routers. Note that they have different link bandwidth and buffer requirements and therefore different area and power. The no-load latency of GHC topologies are the lowest. Their saturation throughput is close to 1. Mesh topologies have the

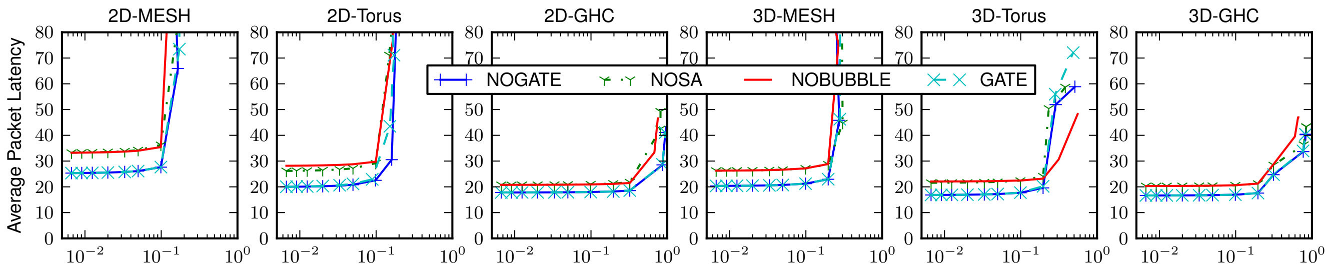

Fig. 7. Performance Impact (Throughput vs Latency) of individual optimizations

highest no-load latency due to greater number of hop counts and lowest throughput. In general, the greater the number of ports, the lower will be the hop count and lower will be the no-load latency with higher saturation throughput. This is not true in the case of other routers like deflection and baseline which has slow increase in performance with increase in number of ports. This makes CEB well suited for high radix routers.

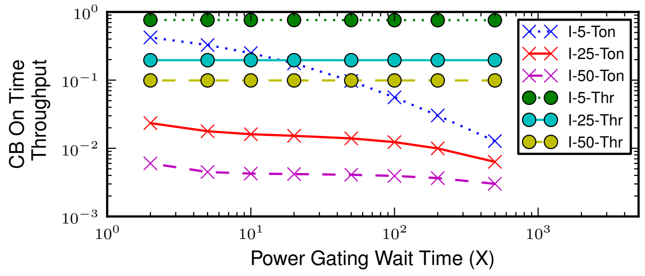

Fig. 8. Sensitivity to Power Gating Wait Time (Cycles)

Sensitivity to Gate Wait Time: As discussed in section III-D, central buffer is only turned on after waiting for X number of cycles. The wait time should have an impact on both throughput and power. Figure 8 plots saturation throughput (Thr) and the percentage of time CB was on (Ton) by changing the value of X for 2D GHC topology with different injection rates. Here, I-5, I-25 and I-50 mean that the maximum injection time between subsequent packets are 5 (1 cycle per flit), 25 and 50 cycles, respectively. We can see that CB on-time greatly reduces with increase in the wait time, while the reduction in throughput is extremely small. Thus large values of X should be used to reduce leakage. We fixed the value of X to be 500 in all our power gating simulations. Dynamically adjusting the wait time at different loads can further reduce power reduction and improve throughput but is left for future work.

RowNo	Parameter	2D-Torus	3D-Torus	3D-GHC	2D-GHC
1	**Baseline-M1**	100	124	110	145
2	**EB-M1**	60	68	60	70
3	**Baseline-M4**	280	376	320	460
4	**EB-M4**	120	152	120	160
5	**FD-P4**	55	61	70	85
6	**FD-P20**	135	141	150	165
7	**CEB-GATE**	55	61	70	85
8	**CEB-NOGATE**	73	79	88	103
9	**CEB-NOSA**	78	86	98	118

TABLE II
BUFFER SPACE (KB) WITH DIFFERENT CONFIGURATIONS

C. Buffer Space Reduction Analysis

Since our main goal is to reduce the buffering requirement of the network, we perform buffer space analysis for different routers and optimizations of CEB. Table II gives the total buffer space requirements of different routers with different topologies. The formula for calculating the buffer space is $[(P * (F * VC + O) * M) + C + I + E] * L$, where L is the link width, P is the number of ports per router, F, O, C is number of flits in input, output and central buffer respectively and I and E are the injection and ejection queue size in flits. M is the number of virtual networks required to support different message classes. For this analysis, torus topologies use 2 physical channels or 2 VCs in EB or baseline router respectively and GHC use 1 VC.

We can see that the baseline router requires large buffer space even with single message class (row 1). EB with 1 message class requires less storage but it increases significantly with the increase in number of message classes as can be seen by row 4 with 4 message classes. Since GHC topologies use only 1 VC or virtual network, the storage requirement is reduced, however this will reduce throughput as well (not simulated). FD (row 5) requires the least buffering space. However, if we consider that it has to re-organize flits coming out of order at the network interface, which requires larger storage, the buffer space requirement of FD will also increase. If we increase the flits space in ejection queue to 5 times, the buffering requirement of FD easily surpass most other networks (row 6). This is because the total ejection queue size aggregated over all NIs is 20K which will be increased to 100K. Thus reorganization overhead of FD is high both in buffer space and latency (which we have not model). Baseline

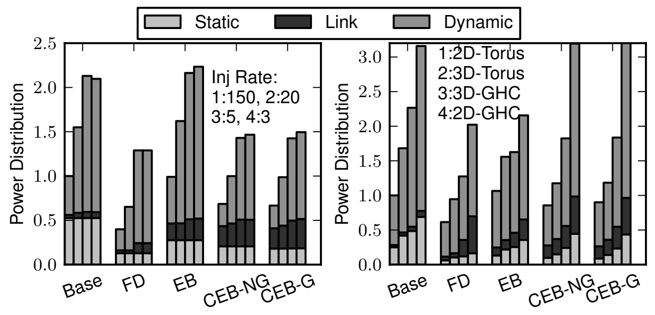

Fig. 9. Static and Dynamic power distribution (a) varying injection rate, (b) varying topology

CEB requires more buffer space than EB for single message class due to the presence of central buffer (row 8). But since, it does not require extra buffer storage to handle message classes, the storage requirement does not increase. When the gating is turned off, CEB has equal amount of buffer power as that of FD with no re-organization overhead. With gating on, the storage requirement increases slightly, however the increase is small specially for high radix topologies. On the other hand, as discussed earlier the throughput will be extremely high as compared to FD. Finally, the last row shows that lookahead SA saves 5K and 15K of buffer space for 2D torus and GHC topologies respectively.

D. Power Analysis

Figure 9 (a) compares the static and dynamic power dissipation of different routers configured in a 2D torus topology and normalized to the baseline case at injection rate of 150 cycles per packet. The different bars represent different injection rates (maximum time between 2 successive packets from a node) as given by the text in Figure 9(a). i.e., bar 1 represents injection rate of maximum 150 cycles per packet, bar 2 represents injection rate of 20 cycles per packet and so on. As obvious, baseline and FD have the maximum and minimum static power, respectively (the bottom component of each bar). Among the routers with elastic links, EB has more static power than CEB. This is because of the minimum requirement of having 4 physical networks (2 for each message class). This also results in EB having the highest dynamic power specially at high loads. Note that EB routers are 64 bit and individual networks have lower power. Elastic links have high power compared to others. This is because of both high activity and larger unit power (power required to traverse a flit). The dynamic power of CEB is low compared to baseline and EB routers due to its small buffering space. Power Gating of the central buffer, although, reduces static power but its advantages at high loads are small.

Figure 9 (b) shows the same plot with various topologies. This time it is normalized to the baseline case with 2D torus topology. CEB router has static power increase comparable to EB router. The dynamic power of CEB, however, increases rapidly. This is because of the very high saturation throughput and thus high activity of the CEB routers. Small increase of dynamic power in EB routers is attributed to thinner channels and crossbars. Although, this along with high saturation throughput makes EB routers a good candidate for NOCs, they loose on no-load latency. Furthermore, their power increases dramatically with increased number of message classes.

E. Results with Real Benchmarks

Figure 10 (a) shows average packet latency of 2D-Mesh network with different routers normalized to the CEB case. In general, FD has the maximum average latency while CEB has the least. In few of the benchmarks, this latency is extremely high. This is due to unnecessary deflections and lack of starvation avoidance in FD routers. In these routers, packets at the injection queue are prioritized lower than the packets already present in the network guaranteeing availability of ports. However, this can potentially lead to starvation at very high load and thus increased latency. Average latency of baseline and EB routers increase by 1.4-1.6x than CEB due to increased no load latency and lower throughput. The trends are similar across different benchmarks. Figure 10 (b)

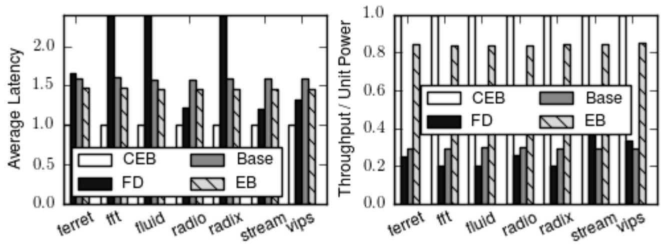

Fig. 10. (a) Normalized average packet latency (b) Normalized throughput per unit power for real application traces

shows the throughput per unit power of the same configuration normalized to the CEB case. CEB performs better than all other routers. Again, FD routers perform the worst because of its large packet latency. All other routers have similar throughput as they retired almost equal number of packets in a fixed amount of time. However, the power consumed by baseline and EB router is higher than the CEB case. This behavior is directly attributed to higher latency and larger buffering requirements of both the baseline and EB routers. We conclude that CEB reduces power at fixed load and decreases average packet latency. If further reduction in latency is required, high radix topologies can be used. Under a fixed load CEB will perform better (both in terms of latency and power).

V. Related Work

Reducing latency within the router has been explored in a number of works. Scarab [11] and Chipper [7] uses buffer-less routing similar to BLESS. [9] uses packet dropping to remove deflections in buffer-less routers. Roshaq [24] and IBM-SP2 [1] uses the same concept of having shared central buffers along with a fast bypass path for low load common case. Both use credit based flow control and has inherent limitations with longer pipelines. 'High Throughput Shared Buffer NOC' router [21] uses multiple shared buffers and 2 crossbars and are power hungry. Our design performs better than these cases due to minimal deterministic routing, EB links and single central buffer. Prediction routers [14] speculatively perform SA stage in parallel with IB. The overhead of speculation, however, is high. Our lookahead SA is deterministic, thus does not have any prediction overhead.

Some other designs have used EB links to reduce pipeline bubbles. Hybrid EB-VC [16] adds VC buffers to avoid deadlocks in EB. They use a technique similar to on-off flow control for drainage of flits into VC buffers. Again, on-off flow control requires these buffers to be large enough increasing their area and power. Kilo-Noc [10] uses EB for its MECS topology. They fall back to VC based buffering space in the routers to avoid flits of different virtual channels to deadlock each other. Furthermore, their approach is tailor-made for MECS topology. The scope of our router is much broader as it can work with many different topologies and favours high radix networks with reduced buffering requirement.

Chen et. al [4] recently proposed Critical bubble and Worm bubble flow control [3]. These needs to be incorporated with our design to increase throughput further. Dimensional bubble flow control [2] uses bubble to provide adaptivity in Mesh networks using single VC. We have not explored adaptive routing with CEB. But providing adaptivity using similar approach is a key next step. Finally, power gating [15] has been explored but as discussed earlier, its scope is limited in input based VC routers. Central buffers are a natural component for gating.

VI. Conclusions

In this paper, we present CEB, a novel low latency, low power router micro-architecture and compared it with other low latency designs. We have shown that a small central buffer in an EB channel based router can be used to avoid deadlock and improve throughput without the need of having separate physical networks. We further presented lookahead SA which is used to achieve no load latency comparable to wire latency. Our results show an average improvement of 3x in throughput / unit power and 1.6x in average latency for PARSEC and SPLASH benchmarks configured in a 2D Mesh topology. The improvements get higher with higher radix topologies. The key next step is to provide support for adaptivity and QoS guarantees (generally provided using VCs) in CEB networks that does not have multiple VCs.

References

[1] T. Agerwala *et al.*, “Sp2 system architecture,” *IBM Systems Journal*, 1995.

[2] X. Canwen *et al.*, “Dimensional bubble flow control and fully adaptive routing in the 2-d mesh network on chip,” in *Embedded and Ubiquitous Computing, 2008.*, dec. 2008.

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

[4] 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.*

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

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

[7] C. Fallin, C. Craik, and O. Mutlu, “Chipper: A low-complexity bufferless deflection router,” in *HPCA, 2011.*

[8] C. J. Glass and L. M. Ni, “The turn model for adaptive routing,” *J. ACM*, vol. 41, no. 5, pp. 874–902, 1994.

[9] C. Gómez *et al.*, “Reducing packet dropping in a bufferless noc,” *Euro-Par 2008–Parallel Processing*, 2008.

[10] B. Grot *et al.*, “Kilo-noc: A heterogeneous network-on-chip architecture for scalability and service guarantees,” in *ISCA, 2011.*

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

[12] 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.

[13] A. Kumar *et al.*, “Toward ideal on-chip communication using express virtual channels,” *Micro, IEEE*, jan.-feb. 2008.

[14] H. Matsutani *et al.*, “Prediction router: Yet another low latency on-chip router architecture,” in *HPCA 2009.*

[15] H. Matsutani *et al.*, “Ultra fine-grained run-time power gating of on-chip routers for cmps,” in *Networks-on-Chip (NOCS), 2010.*

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

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

[18] L.-S. Peh and W. Dally, “A delay model and speculative architecture for pipelined routers,” in *HPCA 2001.*

[19] V. Puente *et al.*, “Adaptive bubble router: a design to improve performance in torus networks,” in *International Conference on Parallel Processing, 1999.*

[20] D. Rahmati *et al.*, “Power-efficient deterministic and adaptive routing in torus networks-on-chip,” *Microprocessors and Microsystems*, 2012.

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

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

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

[24] A. T. Tran and B. M. Baas, “Roshaq: High-performance on-chip router with shared queues,” in *ICCD 2011.*

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