# Brownian Bubble Router: Enabling Deadlock Freedom via Guaranteed Forward Progress

**Mayank Parasar** **Ankit Sinha** **Tushar Krishna**
*School of ECE, Georgia Institute of Technology, Atlanta, GA, USA*
mparasar3@gatech.edu ankit.sinha@gatech.edu tushar@ece.gatech.edu

***Abstract*—Deadlocks are a bane for network designers, be it a Network on Chip (NoC) in a multi-core or a large scale HPC/datacenter network. A routing deadlock occurs when there is a cyclic dependence between the buffers of network routers. Most modern systems avoid deadlocks by placing routing restrictions or adding extra virtual channels, in turn hurting performance and adding overhead respectively.**

**In this work, we demonstrate that instead of placing such restrictions, we can, in fact, design routers to themselves guarantee deadlock-freedom, by (i) ensuring that every router always has at least one bubble (i.e., free buffer slot) at any input port, and (ii) this bubble pro-actively moves between input ports. We call this a Brownian Bubble Router (BBR). A BBR guarantees forward progress in any network topology, without requiring any routing restrictions or additional virtual channels.**

**With our BBR design we provide 4× better throughput over state of art deadlock recovery schemes and 40% better throughput over traditional deadlock avoidance schemes in a 8x8 Mesh at negligible area and power overheads.**

***Index Terms*—Computer architecture, Network-on-chip, Interconnection network, Deadlock**

## I. Introduction

In interconnection networks, a deadlock is defined as a cyclic dependence between router buffers that renders forward progress impossible, since every upstream packet in the cycle waits indefinitely for the downstream packet to leave. Fig. 1(I) shows an example. Network designers put a lot of effort in making sure that the network is guaranteed to be deadlock free. This is because deadlock freedom is a *correctness* issue rather than a performance problem. Naturally, networks need to be formally proven to be deadlock free [1], [2] for any traffic flow, as one may not know, in general, what kind of traffic/runtime condition the network would be subjected to when deployed. In addition to dynamic traffic conditions, there could also be runtime changes in the network topology due to faults, or power gating of network components. This makes the problem of deadlocks even more challenging [3], [4], [5].

Almost all modern networks use one of the following two techniques to *avoid* deadlocks - turn model or vc-partitioning. A turn model leverages Dally's theory [1] to ensure that a cyclic dependence will never get created for any traffic pattern, by restricting certain turns from never being taken. This guarantees deadlock-freedom, but comes at the cost of reduced path diversity. VC-partitioning based solutions leverage Duato's theory [2] and ensure that there is at least one *escape* VC that maintains turn restrictions (and is thus guaranteed to be deadlock-free via Dally's theory [1]) while the remaining VCs can allow any turn. This guarantees deadlock-freedom, but comes at the cost of more resources (VCs) within each router. For example, an escape-VC based solution requires at least two VCs in a mesh, and at least three in a dragon-fly [6][1].

Bubble Flow Control (BFC) [7], [8], [9] is another technique for deadlock avoidance that works on the principle that ensuring the presence of one *bubble* within a ring via controlled injection can guarantee forward progress. Unfortunately, it only works in ring (and by extension Torus) topologies.

There has been another class of solutions arguing for deadlock *detection and recovery*, since deadlocks are quite rare [10], [11], [12]. However this does not come for free, and requires expensive circuitry and overhead to detect deadlocks and extra buffers added at design-time to be used for the recovery process [10], [11], [12].

In this work we make a case for a new perspective for guaranteeing deadlock freedom. Instead of avoiding deadlocks or recovering from them, we show that we can allow cyclic dependences and deadlocks to form, but ensure that they never persist by guaranteeing *forward progress* through every router. We define forward progress as a requirement for every packet to *eventually* leave the current (upstream), router and move to its downstream router. This begets a key question: How can a router ensure forward progress when that depends on the state of the network at the downstream routers (and may in fact cycle back on to the upstream router in case of a deadlock). We show that this can be ensured if we guarantee the following *invariant*: for every buffered packet, there will eventually appear at least one free buffer (i.e., bubble) at the desired input port of its downstream router. Theoretically, this is sufficient to guarantee deadlock-freedom, since the packet can keep moving forward.

To guarantee this invariant, we introduce the novel idea of a *brownian bubble (BB)*, which is a bubble that keeps *moving* through the input ports of the router at a certain frequency. The bubble is not an additional buffer - instead one of the empty VCs (say $VC_i$) at one of the input ports (say port $p_a$) of the router acts as a BB at a certain time. Bubble movement essentially means moving a packet from a full VC (say $VC_j$ at port $p_b$) to the empty $VC_i$ within the router. From this point on, $VC_j$ at port $p_b$ is now the BB.

BBR works with arbitrary topologies and routing algorithms, requires no turn-restrictions, or additional VCs. It can enable fully-adaptive routing with just 1 VC, unlike current solutions. We observe 1.4-4 × higher throughput on average across traffic patterns on a 64-core mesh compared to the state-of-the-art deadlock avoidance and recovery schemes respectively.

[1]Here we are talking about the minimum number of VCs required purely for the sake of avoiding routing deadlocks. A network will use more VCs to avoid protocol-level deadlocks or to avoid head-of-line blocking

978-1-4673-9030-9/18/$31.00 2018 IEEE.

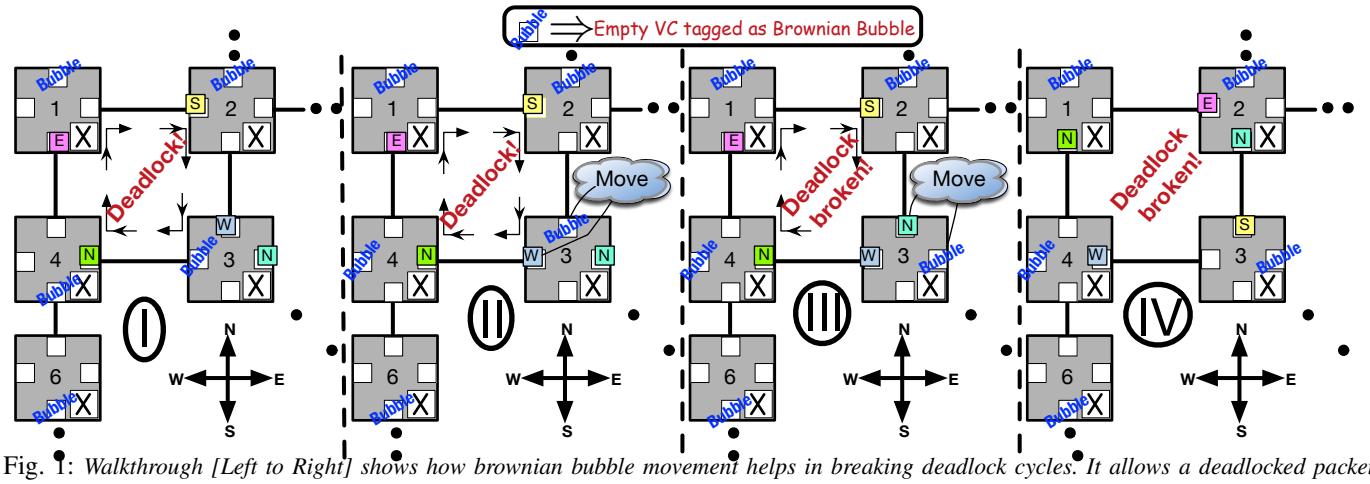


Fig. 1: *Walkthrough [Left to Right] shows how brownian bubble movement helps in breaking deadlock cycles. It allows a deadlocked packet to move to some other port in its router, and other packets, not part of the deadlock ring, to acquire its place and eventually leave the router, thus breaking the deadlock ring. In this example, it takes two bubble movements to break the deadlock.*

## II. Background and Related Work

A rich body of work has been explored in NoCs to provide deadlock-freedom. We classify these into three broad categories: *Deadlock avoidance*, *Deadlock detection and recovery* and *Deflection* and briefly discuss prominent techniques in each.

### A. Deadlock Avoidance

*1) Turn Model:* Routing algorithms are designed such that there are no cycles in the extended *Channel Dependency Graph* (CDG). Example of such routing algorithms are dimension-ordered XY routing, West-First routing [13], in a Mesh and Up-Down [14] routing in irregular topologies. These routing algorithms work by restricting certain turns that a packet can take, thus effectively restricting the formation of deadlock cycle. The main shortcoming of the Turn model routing is that the path diversity is not fully utilized. Moreover, in case of dynamic faults in the network, this technique can still deadlock [3], [11].

*2) Virtual Channel (VC) Partitioning:* VC partitioning overcomes the shortcomings of the turn model by adding virtual channels (VCs) and ensuring that there is no cyclic dependence between the VCs. Packets need to change VCs on every turn, or on some turns. Each VC by itself implements a turn-model, and is thus deadlock-free. Examples of this implementation include O1TURN [15] in a mesh and UGAL [16] in dragon fly. A more optimized version of this design is an escape VC (eVC) based design that allows all turns to be taken by all VCs except one - the *Escape VC*, which implements a turn-model. VC partitioning techniques allow better link utilization since together all VCs do allow all links to be used. However, they introduce area overhead as now we need at least 2 (or more) VCs per input port in the network.

*3) Flow-Control-based Schemes:* Bubble Flow Control [7] is used in rings and provides deadlock avoidance is by runtime injection restrictions. There is no turn restriction imposed on the path taken by the packet, so the CDG may be cyclic. However, injection into the ring is restricted such that there is at least one bubble in every router [7], [9] or in the ring [8]. BFC can also be extended to a torus by viewing a turn (say X to Y) as an injection into the Y ring [9]. Unfortunately BFC only works in rings. Extending the idea of injection restriction to arbitrary topologies would require prior knowledge of the traffic flow, and is thus not general-purpose. Adaptive flit dropping [17] allows flits to be dropped if they fail to traverse a switch beyond a threshold number of times, which frees up buffers, thereby providing deadlock-freedom (and reducing congestion).

BBR leverages the same principle as BFC in terms of requiring a bubble to ensure forward progress, However, in BFC [7] and its variants [9], [8], the bubble moves only when a packet moves from one router to the other, while in BBR, our *brownian* bubble pro-actively moves through different ports of the router which allows us to dynamically introduce bubbles in deadlocked rings. BBR thus works with arbitrary topologies and does not require any injection restriction.

*4) Spanning Trees:* In irregular network topologies - either by design or due to faults or power-gated components, deadlocks are avoided by constructing spanning trees over the topology [3], [18], [19], [20]. These techniques ensure deadlock-freedom by routing packets via the root thereby avoiding cyclic paths, but leads to non-minimal paths between certain source-destination pairs, and also introduces additional complexity for spanning tree construction [3], [18], [19], [20].

### B. Deadlock Recovery

*Deadlock recovery* proposes to detect the deadlock during runtime and recover from it [21], [11], [12]. The rationale is that deadlocks are often quite rare - therefore turn-model restrictions or VC over-provisioning done by avoidance schemes is an overkill. However, because of complicated logic for tracking dependence cycles in the network and then providing guaranteed mechanism of recovering from it, none of these scheme have made it to commercial designs.

### C. Deflection

*Deflection*-based networks misroute packets whenever there is contention [22], [23], [24]. Since a packet never stalls, these are naturally deadlock-free. However, mis-routing creates congestion in the network [22], [23], and higher energy consumption in the links due to non-minimal routes.

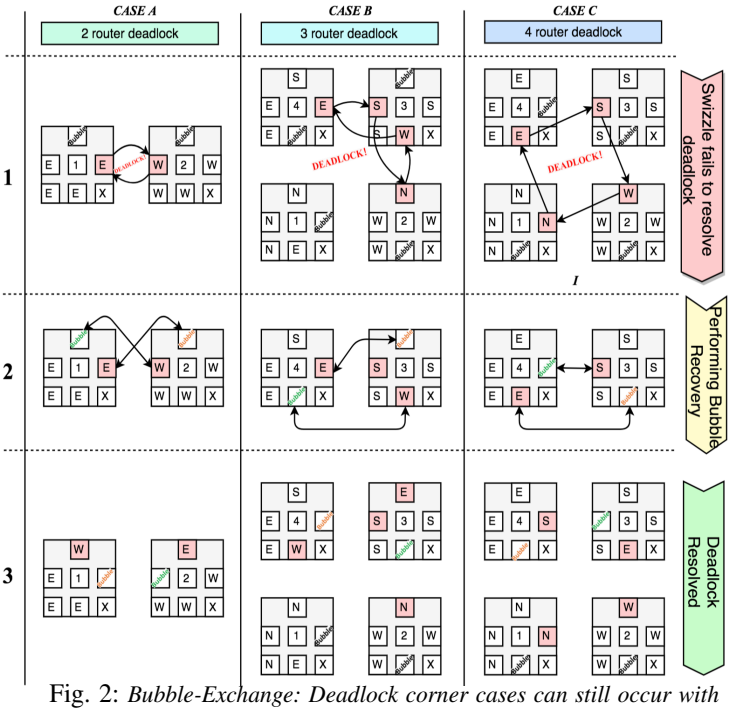


Fig. 2: *Bubble-Exchange: Deadlock corner cases can still occur with simple bubble movement technique (§III-A1). In each column, the first row shows the deadlock ring with involves 2, 3 and 4 routers respectively; the second row shows the bubble-exchange state in action, and third row finally shows the routers state after deadlock is broken.*

## III. Brownian Bubble Router

Brownian Bubble Router proposes to instrument routers with the ability of moving a *bubble* across the input ports of the router. **A bubble refers to an empty packet-sized input VC in a virtual cut through router.** This provides an opportunity to a packet that is currently part of a deadlock cycle, to move into an empty VC in the router, potentially breaking the deadlock cycle. We have assumed each VC to be as deep as 1-packet in this work. See §III-B for extensions.

At the beginning of the network run, one of the VCs (VC-0 for simplicity) at one of the input ports (excluding the injection port) of each router is tagged as the "brownian bubble (BB)". The *invariant* of BBR is that, there will always be one BB present per router. To maintain this invariant, no packet from any other router is allowed to enter the BB. Otherwise the bubble might get consumed while the deadlock continues. At a certain user-specified frequency, the BB is moved across input ports. Moving the bubble essentially means tagging some target VC at some other input port as the BB. If the target VC is non-empty, all flits of its packet are explicitly moved into the original bubble - for e.g., for a 5-flit packet, this step takes 5 cycles. If the target VC is empty, some additional credit signals are required, as we discuss later in §III-B2.

The frequency of bubble movement (BM) is based on an epoch counter. We use *BBR-k* to refer to a BM every $k$ cycles, where $k$ is user-specified. The target VC that the bubble is moved into is also a design choice. For the purpose of simplicity, in our implementation the target VC is *randomly* selected by an arbiter with higher priority being given to empty over non-empty VCs, to avoid explicit packet moves.

### A. Key Concept

*1) Walk-through Example of Bubble Movement:* Fig. 1 illustrates functioning of of BBR in detail. As shown in Fig. 1(I), packets present in the *South, West, North* and *East* input ports of router 1, 2, 3 and 4 respectively are currently in a deadlock. The direction (N/S/E/W) written on the packets is the direction in which the packet is destined to go, in order to reach to its destination. The empty VC at the West input port is tagged as a "bubble". For the purposes of this example, consider only Router 3 (R3). In Fig. 1(II), the packet at the North input port of R3 is moved into the bubble - thus the packet now sits in the West input port, while the VC at the North input port is now free (and tagged as the BB). As explained earlier, to maintain the invariant of keeping one BB per router, no external packet (from another router) is allowed to enter the BB. Thus even though there is a free VC in the deadlocked loop, the deadlock persists. Next, in Fig. 1(III), the bubble moves to the East input port, and the packet sitting there comes to the North input port.' This packet wanted to go North (to router 2) which is *unblocked* (since Router 2's South input port is free), and will thus leave the router. This is shown in Fig. 1(IV). At this point, it will free the VC at the North input port, into which the packet sitting in Router 2's West input port can move, leading to forward progress. Fig. 1(IV) shows the final state of the network, where the four packets originally part of the deadlock ring have made forward progress and now there is no deadlock. Note that the direction initial written on the packets still represents the direction in which the packet needs to move in order to reach to its destination.

*2) Proof of Deadlock Freedom:* Suppose there are one or more deadlocked rings going through a router.

**Definition: Unblocked Packet**. A packet that will eventually leave the router because of a free credit at its downstream router.

**Theorem:** As long as a router has (a) at least one empty input VC (not the BB) or (b) at least one *unblocked packet*, BBR breaks any cyclic dependence going through this router.

**Proof:** Bubble movement can move one of the deadlocked packets into the empty VC (in case (a)), or into the VC originally occupied by the unblocked packet (in case (b)). In case (b), the unblocked packet will eventually leave the router, leaving an empty VC equivalent to case (a). The introduction of an empty VC into the deadlocked ring can guarantee forward progress, breaking the deadlock.

The walk-through example in Fig. 1 had an unblocked packet in Router 3. However, this might not always be true. Next, we discuss how an unblocked packet can be introduced into the router, in case it does not exist, via a *Bubble Exchange*.

*3) Bubble Exchange:* With the current BM technique discussed so far, it is still not guaranteed that deadlock will be broken, depending on the output directions of the buffered packets. Consider the three cases present in Fig. 2. Each column in the figure shows a deadlock scenario, bubble exchange and final state of the router after exchange. Like before, the initial of the direction present on the packet is the direction in which the packet is intended to go. Colored packets in the router are the ones which are involved in the deadlock ring, there state

is show before deadlock, during bubble exchange and after deadlock in row-I, II and III respectively.

Consider Row-I. Case A shows a 2 router deadlock. Packets intending to go East are sitting in the east input port at the first router, and those going west are sitting at the West input port of the neighboring router, leading to a deadlock dependence. This would not occur in a baseline design if u-turns are not allowed. However, with bubble movement, this scenario is possible - recall that in Fig. 1(III), the North input port of Router-3 houses a packet that wants to go North. In Row-1, Case A, no amount of bubble movement in the two routers can resolve the deadlock, since all packets in the respective routers point to the same direction. From the necessary condition described in §III-A2, this is because of the lack of an unblocked packet in either of the routers. Row-1, Case B shows a 3-router deadlock. Here bubble movement is possible with the packets requesting different output ports. We do not enumerate all possible scenarios here in the interest of space, but all bubble movements will still end up with a 2-router or a 4-router deadlock. Finally, Row-2 Case C shows a 4-router deadlock, where no amount of bubble movement can resolve it, again due to the absence of an unblocked packet in any of the routers.

To resolve deadlocks in such scenarios, we introduce the concept of *bubble exchange (BE)*. The idea is to force forward progress of one of the packets by moving it into the bubble at its downstream router, and then recover the bubble by moving one of the packets at the downstream router into this router. BE is initiated by the upstream router when all the neighboring *downstream* routers that the packets of **this** upstream router want to get to have an *occupancy* of $N-1$, where occupancy is defined as the number of non-empty VCs in the router across all ports (except local), and $N = num\_inport \times num\_vcs\_per\_inport\_except\_local$. In other words, BE is initiated when the downstream routers are completely full, except their BBs. BE takes two steps:

① Upstream router routes one of the packets to the downstream router to sit at the BB of the downstream router. This is equivalent to the upstream router consuming the BB of the downstream router. This leads to a situation where there are 2 bubbles present at upstream router and none at the downstream router, breaking the BB invariant temporarily.

② The downstream router mis-routes one of its packet to upstream router's original bubble, to recover its bubble back. The input VC of the packet chosen to mis-route is selected at random, and becomes the BB at the downstream router.

Note that steps ① and ② described above, are performed in tandem on bi-directional link connected between the upstream and downstream routers involved in the bubble exchange.

Row-2 in Fig. 2 shows bubble exchange in action for all three cases. The deadlocks in all cases are broken in Row 3.

**Why does Bubble exchange guarantee deadlock freedom?** As discussed in §III-A2, BBR works only if a router has an unblocked packet that is guaranteed to *eventually* leave. Bubble exchange forces one of the packets in the router to make forward progress towards its destination, essentially making it an unblocked packet for the purposes of the proof. In the worst case, a packet might move all the way to its destination via bubble exchange, where it will eventually get consumed[2].

**Does Bubble exchange require deadlock detection?** No. It is important to note that the bubble recovery algorithm is a heuristic for exchanging bubbles between neighboring routers. We do not actually detect the deadlock, thereby do not pay its associated overheads in terms of timeout counters and probes [11]. This implies that there can be false positives.

**Why is Bubble exchange performed at an occupancy of N-1?** In BBR, the necessary condition for a deadlock is an occupancy of (N-1), since the brownian bubble is always empty. Since explicit deadlock-detection is not performed, an occupancy of N-1 triggers a guaranteed forward movement of one of the blocked packets via bubble exchange. This guarantees that there will never be any false negatives, though there may be false positives (i.e., an occupancy of N-1 due to congestion and not a true deadlock).

**Does Bubble exchange lead to mis-routing?** Sometimes, but not always. Bubble exchange always leads to forward progress of at least one packet. In certain cases, the other packet might be moved to a neighbor that is not its actual preferred output port. However, in other cases, such as Fig. 2-Case A, both packets might end up making forward progress.

**Does Bubble exchange lead to livelocks?** It is theoretically possible, though extremely unlikely for the same packet to keep getting misrouted as part of the bubble exchange condition at every router it enters, never reaching its destination, leading to a livelock. Livelocks can be avoided by disallowing more than a certain number of misroutes for any packet, like prior works on mis-routing have explored [23]. We do not implement it in BBR for simplicity. Our evaluations show that the number of misroutes is actually quite low.

### B. Implementation

Fig. 3 shows the BBR microarchitecture. In addition to modules such as VC Allocator, Route Compute, Switch Allocator and Crossbar which have their usual function as in a baseline router, we introduce a few additional ones to implement BM and BE. We implemented the BBR modules in RTL, and observed around 7.4% area overhead (Fig. 3) and 4.3% power overhead over a 4-VC baseline router [25] post-layout at 28nm. BBR introduces an additional mux in front of each VC, but meets timing at 1GHz like the baseline. Thus BBR's additions are extremely light-weight.

We also show a a flow-chart of the BBR operation in Fig. 3. Each functional unit specific to BBR is color-coded with the same color as its microarchitecture counterpart. We describe key components next.

*1) Bubble Movement Epoch Unit:* Based on the configurable epoch parameter k, the Bubble Movement Epoch unit triggers a BM every k cycles. BM may be aborted in a special case discussed later in the credit management unit.

**Bus.** We add a small bus inside the router connecting the outputs of all the VCs excluding the port VCs. This is to

[2]We assume that the protocol is deadlock-free, and any packet in a router may stall but will eventually get consumed by its destination.

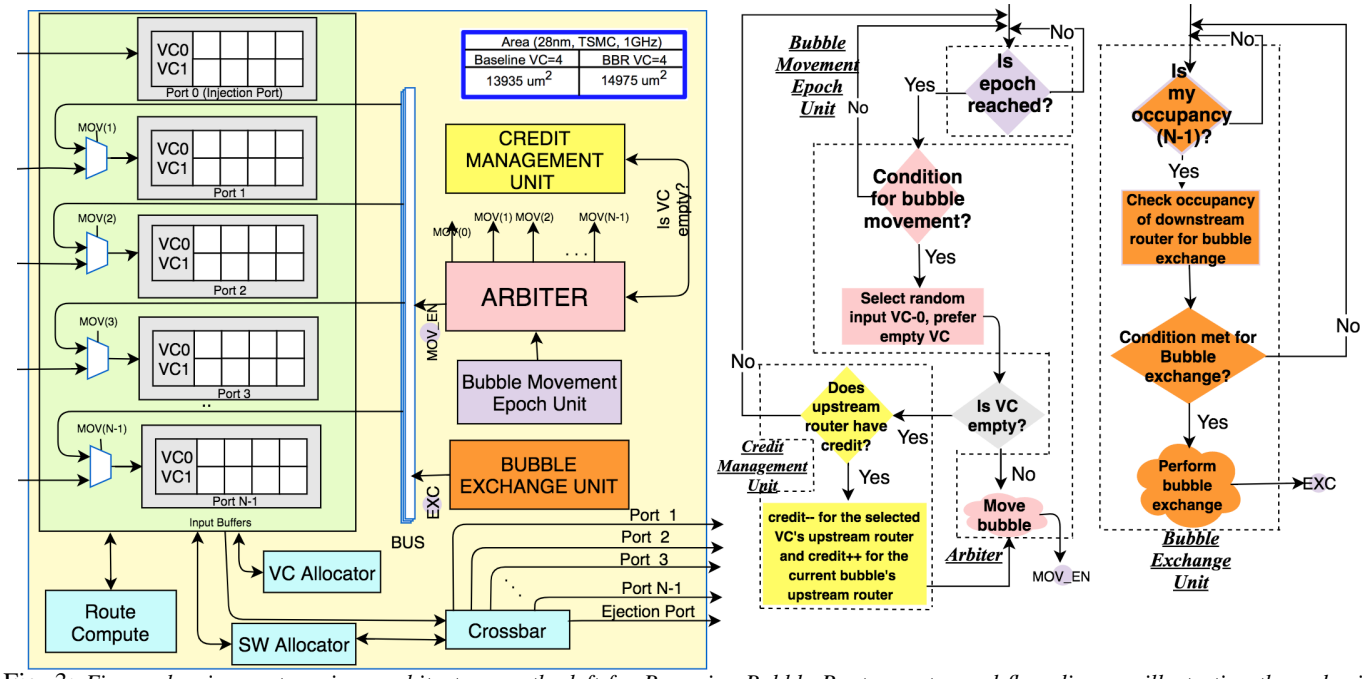


Fig. 3: *Figure showing router micro architecture on the left for Brownian Bubble Router router and flow diagram illustrating the order in which Brownian Bubble Router specific actions are performed on right. Note that Brownian Bubble Router router concept is generic to any underlying topology, hence number of ports are kept as N for generality of the idea. Here VC stands for virtual channel. Specific details about each module are discussed in §III-B. The area consumed by the router at 28nm is also shown.*

facilitate bubble movement between two ports. We chose as bus based on the insight that at any point in time there is only one packet which would be moved to the BB to perform BM. This implies that there will never be contention on this interconnect media, which suits the bus. Also, since we randomly move bubble across input ports by giving preference to empty input ports over non-empty input ports, a bus which connects all input ports fits our purpose. The input to the VCs can be multiplexed between the input link and the bus. This is determined by the arbiter which handles all the multiplexers using control signals (MOV(1), MOV(2), ...). There will never be any contention for the input port of a VC between the link and the bus, since the VC into which a packet is being written from the bus was the BB, and will never receive a packet from the input link.

**Arbiter.** If a BM is triggered, the arbiter chooses the input port for BM by choosing an input VC at an input port in a round-robin manner. Priority is always given to a empty VC, if available. The bus-arbiter unit sends out two signals, the MOV_EN and the MOV signals. The MOV_EN signal is sent to the bus to indicate that a BM is impending in the next cycle while the MOV signal is used to select the port to where this movement will happen (in other words, the port from where a packet will be read out and put on the bus to be inserted into the current BB). This signal remains active till all flits of the packet have been moved.

*2) Credit Management Unit:* Credit management is an integral part of the BBR. As mentioned earlier, no packet is allowed to come and occupy the VC tagged as the brownian bubble. This is ensured by *not* sending a credit for this VC to the upstream router so that it believes that this VC is actually occupied. Thus a BB simply looks like a full VC to the upstream router. During BM, two cases arise.

Case I: The bubble is moved to a full VC (i.e., the packet from the full VC is moved into the bubble). In this case, no credits need to be sent to the respective upstream routers. This is because both upstream routers connected to the downstream router believe that the VCs are full - it is agnostic to the fact that one has an actual packet, one is the BB, and the bubble moved between them.

Case II: The bubble is moved to an empty VC. The VC that becomes the BB needs to send a *decrement credit* signal to the upstream router to inform it that this VC is actually not empty. The original VC which was the BB needs to send an *increment credit* signal to the upstream router signaling that this VC is now free. This is done after one cycle of delay to manage a corner case where the upstream router may have already started sending flits for a new packet into this VC, which is currently on the link. This is handled by *aborting the bubble movement* as follows: (a) if the upstream router receives a decrement credit for a VC that it has already started sending flits to, it ignores the decrement credit, (b) if the downstream router receives flits into a VC that became the BB in the previous cycle, the original (empty) VC is tagged as the BB again. (c) the original VC sends its increment credit signal after waiting for a cycle only if the above scenario does not occur.

*3) Bubble Exchange Unit:* Each router keeps track of its *occupancy*, which was defined earlier in §III-A3. If the occupancy of the router reaches (N-1), where N is the total number of input VCs at all ports of the router (except local), it collects the occupancy of its neighbors. If the neighbors have an occupancy greater than a certain threshold (max threshold is N-1), BE is triggered by setting the EXC flag. This reads one

TABLE I: **Network Configuration.**

| | |
|---|---|
| **Network** | |
| **Topology** | 8x8 Mesh |
| **Router latency** | 1-cycle |
| **Num VCs** | 1, 2, 3, 4 |
| **Buffer Organization** | Virtual Cut Through<br>Single packet per virtual channel |
| **Target Networks** | |
| **Deadlock Avoidance** | West-first and Escape VC |
| **Deadlock Recovery** | Static Bubble [11] and SPIN [12] |
| **Brownian Bubble Router** | BBR-k (k= BM frequency) |

of the packets from the router, and sends it out of the crossbar and output link to the neighboring router. The neighbor in turn sends a packet to this router which is added into this router's BB. A subtle point to note is that BE does not steal any useful link bandwidth since the occupancy of (N-1) at both routers means that they were unable to send packets to each other via regular switch allocation due to the lack of credits, and so the links between them were anyway idle.

BE is a measure of how reactive the router is towards recovering from the deadlock by exchanging the bubble. The occupancy metric we use to trigger BE is just a heuristic. From a correctness point of view, BE can be triggered more proactively or at a fixed time epoch in alternate implementations.

### C. Adding BBR over Alternate Router Microarchitectures

BBR's underlying mechanism of periodic bubble movement across input ports *within* a router, followed by occupancy-driven bubble exchanges between *neighboring* routers, can be applied to any input buffered VC router to guarantee deadlock freedom in the network, as we showed in §III-A2. This makes it agnostic to the underlying topology (mesh/high-radix/irregular/reconfigurable [26]), routing algorithm (XY/adaptive [27]) and router bypass optimizations [25]. BBR can also work with wormhole routers, but will require additional complexity (such as packet truncation [22]) to manage ordering since parts of the same packet might end up at different input ports of the same router due to BM.

## IV. Evaluation

### A. Methodology

We model Brownian Bubble Router in the Garnet [28] cycle-accurate NoC simulator. For BBR, we implement a minimal *fully adaptive random routing* algorithm. We use credits at the downstream router to decide the direction if more than one choice exists. Table I lists the system configurations we evaluated. We contrast BBR against both classic deadlock-avoidance (West-first and escape VC) and state-of-the-art deadlock-recovery (Static Bubble [11] and SPIN [12]) schemes. All networks use a single-cycle router. Recall that the rate of bubble movement (BM) is a knob given to the network designer to tune the frequency of bubble movement within the router. We evaluate BBR with multiple BM epoch values and report results with 1 (i.e., every cycle), 64 (every 64 cycles) and 1024 (every 1024 cycles). For BE, we empirically set the occupancy threshold at downstream routers to 4.

### B. Correctness

The primary claim of Brownian Bubble Router is to make sure there is no deadlock that persists in the network. We show

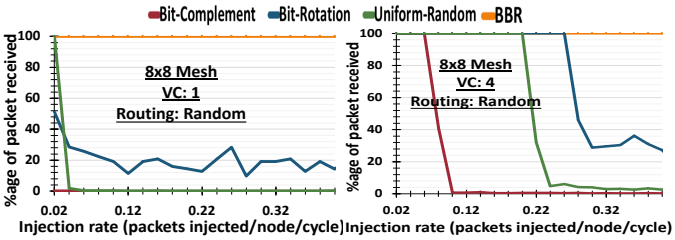


Fig. 4: *Correctness of Brownian Bubble RouterFor a fixed number of packets for the simulation, x-axis shows total packets injected in network per node per cycle and y axis shows %age of total packets received at the end of simulation. Different traffic patterns deadlock with fully-random routing, but BBR never deadlocks with any traffic.*

this in Fig. 4 for a $8 \times 8$ mesh topology. On y-axis we plot the percentage of packets received over fixed packets injected in the network. X-axis shows the injection rate at which these packets are injected in the network. All traffic patterns use fully random routing. Fig. 4 also shows how sensitive the network is towards the number of VCs present per input port in the router. We see that network deadlocks at much lower injection rate when number of VCs is 1 compared to when it is 4. This is especially stark in bit complement traffic which deadlocks almost immediately in a 1 VC design. BBR performs consistently by delivering 100% packets that are injected in the network at all injection rate for all traffic patterns.

### C. Performance

Next, in Fig. 5, we evaluate BBR against state-of-the-art deadlock freedom techniques for for a $8 \times 8$ mesh at different VC counts. In the interest of space, we only present results for transpose, shuffle, uniform-random and bit rotation traffic pattern for 2 and 4 VCs respectively. With 4 VCs, we observe 37% throughput improvement over WestFirst and escapeVC (deadlock avoidance) on average and $3\times$ improvement over Static Bubble and SPIN (deadlock recovery). With 2 VCs, we observe 44% improvement over WestFirst and escape VC, and $.2.5\times$ improvement over Static Bubble and SPIN. For many of the patterns, the performance improvements are higher at 2 VCs as opposed to 4 because of the path diversity provided by fully adaptive routing enabled by BBR in all VCs. In contrast, West-first lacks path diversity in the west direction, while escape VC only provides full path diversity within one of its VCs (the other one restricted to west-first).

With 1 VC, however, we found the latency with BBR to be erratic at a few injection rates. This is because the input port where the bubble resides essentially gets blocked for the upstream router for a period of time, leading to uneven and unpredictable delays until packet reaches its destination. However, it is important to note that a 1 VC BBR design is deadlock-free, as we showed earlier in Fig. 4.

We also performed an experiment to understand the performance overhead BBR adds on top of an already deadlock-free routing algorithm, such as XY. In Fig. 6, we plot the reception rate for a baseline XY scheme and various BBR schemes with 2 and 4 VCs. We notice that an aggressive BBR-1 (that tries to perform bubble movement every cycle) leads to a 25% drop in throughput for uniform random and 35% drop in thoughput for bit complement averaged over all VC count . But with higher

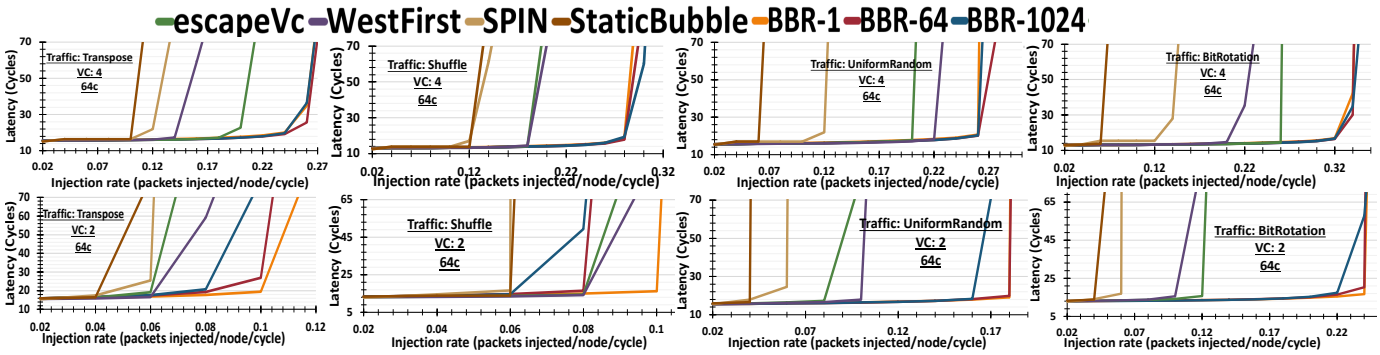


Fig. 5: *Performance of Brownian Bubble Router technique compared against recently proposed deadlock recovery schemes and well known deadlock avoidance schemes such as escapeVC and WestFirst Routing, proving its superiority. Here x-axis shows total packets injected in network per node per cycle and y-axis shows the average latency incurred by packets in cycles.*

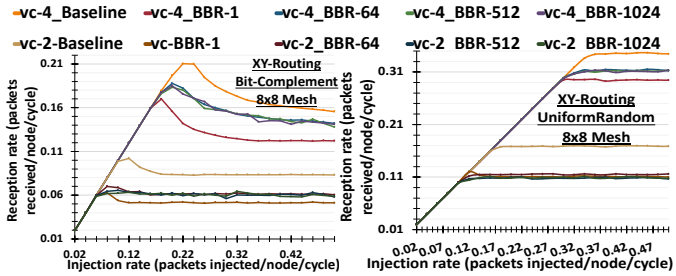


Fig. 6: *Overhead introduced when adding BBR over a baseline deadlock-free XY routing algorithm. Here x-axis shows the packets injected in network per node per cycle and y-axis, similarly shows packets received in the network per node per cycle.*

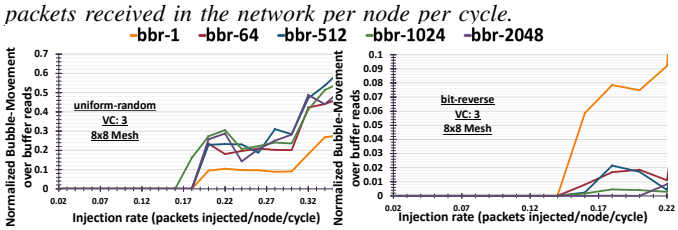


Fig. 7: *Bubble Movement Frequency: y-axis shows ratio of buffer reads (or writes) due to BM over the baseline buffer reads (or writes) and x-axis shows the packets injected in the network per node per cycle. BBR-1 shows the highest BM for bit-reverse compared to other BBR-k; this behavior is opposite in uniform random traffic. This shows distribution of BM across BBR-k is highly traffic dependent.*

values of the epoch, the drop is only 9%. Recall that at high loads, BE kicks in once the upstream and downstream routers start becoming full, which leads to a performance differential at high loads, even if the BM epoch is set very high. If we restrict BE to occur on a very high fixed threshold, rather than based on occupancy, BBR would have essentially no overhead if the underlying algorithm is inherently deadlock free.

### *D. Bubble Movement and Bubble Exchange Frequency*

Having shown the correctness and performance benefits of BBR, next we study the potential overhead. As discussed earlier in §III-B, the area overhead for implementing BBR is negligible. However, each bubble movement involves reading a packet out of its current VC and writing it into an empty VC, increasing the total buffer activity. The same occurs in during a bubble exchange as well, across neighboring routers. This can naturally have energy implications.

We quantify this overhead in the next set of experiments. In Fig. 7 and Fig. 8 we plot the ratio of additional buffer reads

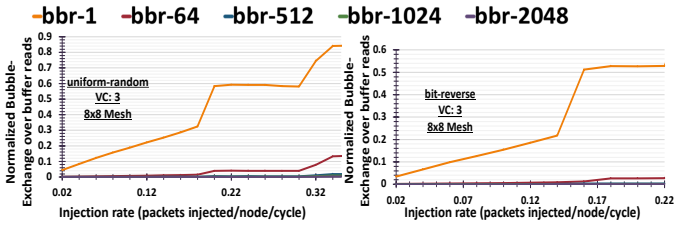


Fig. 8: *Bubble-Exchange Frequency: here y-axis shows ratio of buffer reads (or writes) due to BEs over the baseline buffer reads (or writes) and x-axis shows the packets injected in the network per node per cycle. We see that BBR-1 has highest BE over any other BBR-k.*

(or writes) due to BM and BE respectively over the baseline buffer reads (or writes) for various values of BBR-*k*, where *k* is the frequency of BM. In the interest of space, we plot the behavior for two patterns - uniform-random and bit-reverse which show contrasting characteristics. Other traffic patterns showed similar behavior to one of these patterns. Note that a BM between empty VCs does not count as a buffer read/write.

In Fig. 7, we highlight that there is *no bubble movement* up to a certain injection rate; this is because at low loads, more than two VCs are empty across all ports of the router in most cases, so a BM does not need an explicit packet read and write. This shows that BM actually adds no energy overhead, especially at low to medium loads which is the common operating point for most NoCs. At high injection rates, the network is more susceptible to deadlock, and naturally the number of BMs go up as well. One might expect low values of *k* (i.e., high frequency of BM) to lead to higher number of buffer reads/writes. This can be seen as true in bit rotation, where BBR-1 shows up to 4x more buffer reads/writes than BBR-64 post saturation. Counter intuitively, though, the opposite is seen in uniform random traffic, where a higher values of *k* actually end up leading to more bubble movements overall. This is because with a low frequency of BM, deadlocks persist for longer and end up requiring more BMs to provide forward progress. This shows a subtle yet important feature of BBR that it is adaptive; hence the energy consumption will be more only if the traffic is susceptible to deadlock.

In Fig. 8, we see that BBR-64 and beyond, the number of BEs is negligible. A notable exception is BBR-1 (moving bubble every cycle) which shows the highest number of BEs over any other BBR-k (lower frequency of bubble movement).

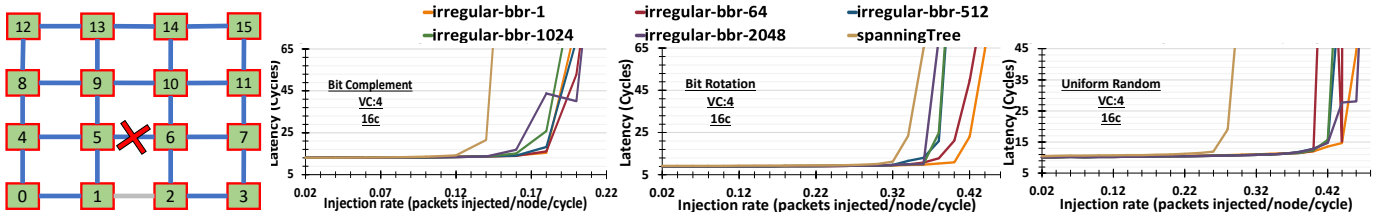


Fig. 9: *A 4x4 Mesh with a faulty link (shown with X). XY routing can no longer work. Traditional deadlock avoidance (Spanning Tree) will disable the use of the grey link to avoid cycles, leading to non-minimal routes. Thus BBR provides higher saturation throughput.*

This can be understood as follows - high BM ends up moving packets to other free VCs in the router very frequently, leading to more occupancy within the router. This in turn triggers BE more frequently. In contrast, at low BM frequency, the router tends to remain emptier (especially at low loads), leading to fewer forward movements due to BE.

In summary, the impact of the BM frequency depends heavily the traffic pattern and injection rate. For uniform random traffic, we observe that BE, not BM actually tends to dominate. Moreover, the energy overhead can be controlled via various knobs exposed to the designers such as the BM frequency and BE occupancy threshold.

### *E. BBR for Irregular Topologies*

Next, we study BBR performance with an irregular topology as shown in left most sub-figure in Fig. 9. Here the link between routers 5 and 6 in a mesh is broken, which could be due to various reasons such as power gating or dynamic faults in the network [3]. A challenge with irregular topologies is that traditional turn-restrictions (such as XY) will not longer work - for e.g., any packet from 5 to $\{10, 11, 14, 15\}$ will have to make a Y to X turn at 9. Similarly, a Y to X turn at 2 will have to be allowed. Such turns could lead to a $5 \rightarrow 9 \rightarrow 10 \rightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 5$ deadlock. In such scenarios, the deadlock free routing option is to construct a *spanning tree* [3], [18], [19], [20], which will not allow the use of the link between router 1 and 2 (highlighted in grey) to avoid cycles. BBR does not need any such restrictions and enjoys the full path diversity. This translates to around 40% higher throughput on average over the spanning tree routing algorithm as shown in Fig. 9.

## V. Conclusion

We propose a novel deadlock-freedom scheme called Brownian Bubble Router (BBR) which guarantees forward progress of packets through every router in an interconnection network. This is done by a clever mechanism of circulating one bubble through all ports of the router, and periodically exchanging it between neighbors. This prevents any deadlock from persisting in the network. BBR provides the following advantages over baseline schemes: (i) it is topology and routing algorithm-agnostic since BM occurs within the router, and BE between direct neighbors. This makes it highly applicable across domains - homogeneous many-cores, heterogeneous SoCs, faulty (or power-gated) NoCs with missing links/routers, (ii) it requires no turn restrictions or escape VCs, thereby providing full path diversity - which in turn translates to higher throughput, (iii) is extremely light-weight. On a 8x8 mesh, across a suite of traffic patterns, BBR provides 40% and 4× higher throughput on an average, respectively, over state-of-the-art deadlock avoidance and recovery techniques.

## References


[1] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, pp. 547–553, 1987.

[2] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, 1993.

[3] K. Aisopos *et al.*, "ARIADNE: agnostic reconfiguration in a disconnected network environment," in *PACT*, 2011.

[4] D. Lee *et al.*, "Brisk and limited-impact noc routing reconfiguration," in *DATE*, 2014.

[5] A. Samih *et al.* , "Energy-efficient interconnect via router parking," in *HPCA*, 2013, pp. 508–519.

[6] J. Kim *et al.* , "Technology-driven, highly-scalable dragonfly topology," in *ISCA*, 2008, pp. 77–88.

[7] C. Carrion, *et al.* , "A flow control mechanism to avoid message deadlock in k-ary n-cube networks," in *HIPC*, 1997.

[8] L. Chen *et al.* , "Critical bubble scheme: An efficient implementation of globally aware network flow control," in *IPDPS*, 2011, pp. 592–603.

[9] V. Puente *et al.* , "The adaptive bubble router," *J. Parallel Distrib. Comput.*, pp. 1180–1208, 2001.

[10] Y. H. Song and T. M. Pinkston, "A new mechanism for congestion and deadlock resolution," in *ICPP*, 2002.

[11] A. Ramrakhyani and T. Krishna, "Static bubble: A framework for deadlock-free irregular on-chip topologies," in *HPCA*, 2017, pp. 253–264.

[12] A. Ramrakhyani *et al.*, "Synchronized progress in interconnection networks (spin) : A new theory for deadlock freedom," in *ISCA*, 2018.

[13] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *J. ACM*, pp. 874–902, 1994.

[14] M. D. Schroeder *et al.* , "Autonet: A high-speed, self-configuring local area network using point-to-point links," *IEEE Journal on Selected Areas in Communications*, 1991.

[15] D. Seo *et al.*, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," in *ISCA*, 2005.

[16] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

[17] Y. Xue and P. Bogdan, "User cooperation network coding approach for noc performance improvement," in *NOCS*, 2015, pp. 17:1–17:8.

[18] R. Parikh and V. Bertacco, "udirec: Unified diagnosis and reconfiguration for frugal bypass of noc faults," in *MICRO*, 2013.

[19] V. Puente *et al.*, "Immunet: A cheap and robust fault-tolerant packet routing mechanism," in *ISCA*, 2004.

[20] R. Parikh *et al.*, "Power-aware nocs through routing and topology reconfiguration," in *DAC*, 2014.

[21] K. V. Anjan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," in *ISCA*, 1995.

[22] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA*, 2009.

[23] C. Fallin *et al.*, "Chipper: A low-complexity bufferless deflection router," in *HPCA*, 2011, pp. 144–155.

[24] P. Abad *et al.*, "Rotary router: an efficient architecture for CMP interconnection networks," in *ISCA*, 2007, pp. 116–125.

[25] H. Kwon and T. Krishna, "Opensmart: Single-cycle multi-hop noc generator in bsv and chisel," in *Proc of the IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2017.

[26] C. Jackson and S. J. Hollis, "Skip-links: A dynamically reconfiguring topology for energy-efficient nocs," in *SoC*, 2010, pp. 49–54.

[27] J. Hu and R. Marculescu, "Dyad: smart routing for networks-on-chip," in *DAC*, 2004, pp. 260–263.

[28] N. Agarwal *et al.*, "GARNET: A detailed on-chip network model inside a full-system simulator," in *ISPASS*, 2009.