QORE: A Fault Tolerant Network-on-Chip Architecture with Power-Efficient Quad-Function Channel (QFC) Buffers

Dominic DiTomaso†, Avinash Kodi†, and Ahmed Louri‡,

†*Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701*
‡*Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721*
dd292006@ohio.edu, kodi@ohio.edu, louri@email.arizona.edu

***Abstract*—Network-on-Chips (NoCs) are quickly becoming the standard communication paradigm for the growing number of cores on the chip. While NoCs can deliver sufficient bandwidth and enhance scalability, NoCs suffer from high power consumption due to the router microarchitecture and communication channels that facilitate inter-core communication. As technology keeps scaling down in the nanometer regime, unpredictable device behavior due to aging, infant mortality, design defects, soft errors, aggressive design, and process-voltage-temperature variations, will increase and will result in a significant increase in faults (both permanent and transient) and hardware failures. In this paper, we propose QORE - a fault tolerant NoC architecture with Quad-Function Channel (QFC) buffers. The use of QFC buffers and their associated control (link and fault controllers) enhance fault-tolerance by allowing the NoC to dynamically adapt to faults at the link level and reverse propagation direction to avoid faulty links. Additionally, QFC buffers reduce router power and improve performance by eliminating in-router buffering. Our simulation results using real benchmarks and synthetic traffic mixes show that QORE improves speedup by 1.3× and throughput by 2.3× when compared to state-of-the art fault tolerant NoCs designs such as Ariadne and Vicis. Moreover, using Synopsys Design Compiler, we also show that network power in QORE is reduced by 21% with minimal control overhead.**

I. Introduction

The shrinking of transistor sizes has enabled the remarkable growth in the number of cores that can be integrated within a single chip (called Chip Multiprocessors (CMPs)). As the number of cores continue to scale and conventional on-chip bus-based communications approach their limits, architects were urged to consider other scalable communication strategies. Network-on-Chips (NoCs) have emerged as the de facto communication paradigm by offering scalability through a modular design [1], [2]. In NoCs, segments of links are connected via routers in order to overcome global wire delays and scalability requirements. However, the combination of links and routers incur a power and area expense which adversely affects NoC performance. Extensive power optimization techniques have been used to mitigate the NoC power consumption. The NoC of Intel's 80-core TeraFlops chip [3] consumes 28% of the total tile power using simple cores, whereas the NoC in the more recent Intel 48-core SCC [4] consumes 10% of the tile power using regular cores. Power optimizations of the NoC fabric is a critical piece of the puzzle to sustain and continue the drastic growth in CMP performance.

A typical NoC hardware consists of routers and communication channels connecting the routers. The router is comprised of input/output ports, buffers, routing logic, and a crossbar connecting input ports to output ports for packet routing. Research has shown that router buffers are responsible for 46% of router power [5] and 30% of router area [6]. This has motivated architects to implement buffer optimization techniques such as elastic buffering [7], [8], [9] and bufferless routing [10], [11]. By completely eliminating buffers and implementing bufferless routing, recent work has reduced the average network energy by 40% [10]. Buffers have been moved from the router to the channels by replacing repeaters on the channel with either flip-flops called elastic buffers [8] or tri-state repeaters called channel buffers [9]. These channel buffers can store packets when the register buffers are congested or propagate data forward when necessary, thereby mitigating power and area penalties associated with router buffers.

Since the buffers have been moved to the channels, hard errors in the channel buffers can cause complete failure, impeding communication between routers. Researchers have been tackling channel failure in NoCs [12], [13], [14], [15], [16], [17], [18], [19]. Recently, the fault tolerant Ariadne network [12] overcame channel faults by reconfiguring packet routing to move around failures. Using up*/down* routing and a series of flag broadcasts, the two unidirectional links between routers were dynamically assigned as up or down to create a tree network that avoid faulty channels. Another fault tolerant NoC called Vicis [13] routed around faults and placed turn restrictions at routers to avoid deadlocks. The BulletProof architecture [20] concentrates on the router (not the channels) and provides efficient fault tolerance schemes for routers to overcome transient and permanent faults.

With the increasing number of cores, NoCs must manage the communication demands, especially when faults are present. Since NoCs are designed to handle peak traffic loads, many communication channels can go under-utilized when network

978-1-4799-3097-5/14/$31.00 ©2014 IEEE

load is high or the workload is unbalanced. Recent research on NoC performance has tackled above mentioned problems using techniques such as reversibility or coding schemes [21], [22], [23], [24], [25], [26]. Hesse et al. propose a bandwidth-adaptive router (BAR) that aims to take advantage of these under-utilized links with bidirectional, adaptive channels [22]. These bidirectional channels adapt channel bandwidth at a fine-granularity according to network traffic demands. Research has shown that channel reversibility can achieve higher throughput and lower average packet latency in NoCs.

In this paper, we propose **QORE** - a fault tolerant network-on-chip architecture with power-efficient Quad-Function Channel (QFC) buffers. We use forward and backward propagation in channel buffers to simultaneously target both power reduction as well as fault tolerance. Utilizing these QFC buffers, we differ from previous work in that we create reversible channel buffers with multiple functionalities: on-demand data storage, on-demand forward data propagation, and backward data propagation. On-demand data storage enables communication channels to act as buffers and store data when the network load is low and function as repeaters when the network load is high. QFC buffers simplify buffering because packets can be routed straight from the router input channel to the crossbar without being buffered in the router as conventionally done in baseline NoC routers. Often, the traffic patterns of real applications leave some channels under-utilized which can waste bandwidth. We use QFCs to improve performance by adapting them to traffic demands, thereby, efficiently utilizing network resources. In addition to improved performance, QORE uses these QFCs to also increase NoC fault tolerance. QORE avoids non-minimal paths around faults by reversing non-faulty channels to allow back and forth communication between routers. We design controllers to reverse links on a router by router basis to improve performance and propose a backup ring network to further enhance reliability in the presence of faults. While prior research has analyzed channel buffers for reducing power in routers with marginal performance penalty and channel reversibility primarily for improving performance, to be best of our knowledge no prior work has examined the use of channel buffers and channel reversibility to simultaneously tackle the challenging issues of power, performance and fault-tolerance. The proposed QORE architecture attempts to address three issues of power, performance and fault-tolerance in a cohesive manner. The major contributions of this work include:

- **Quad-Function Channel (QFC) Buffers:** We design channel buffers that can dynamically function as (1) forward repeaters, (2) backward repeaters, (3) forward buffers, and (4) backward buffers. This enables significant power reduction with no router buffers while enabling circuits to both improve performance and reliability.
- **Improved Performance:** We use our QFC concept to enhance network performance by dynamically monitoring channel utilization and adapting to network traffic. QORE is able to maintain similar throughput and latency to the high performance BAR network [22].
- **Fault Tolerance:** Using the proposed QFC buffers and a backup ring network, we design a fault tolerant network that can adapt to faults with minimum control overhead. Our QORE network is able to handle faults better than the Ariadne and Vicis networks while improving throughput by 2.3×, reducing network power by 21%, and improving speedup by approximately 1.3× on average when running real applications (PARSEC [27], SPEC CPU2006 [28], and SPLASH-2 [29] benchmarks).

II. Motivation

In this paper, we focus on two important concerns in NoCs while also maintaining network performance: high power dissipation and declining reliability. As previously mentioned, buffers consume a major portion of the router power. This has been the key concern that has motivated researchers to develop novel ideas such as bufferless networks [10], [11], dynamic VC allocation [30], and elastic or channel buffers [8], [9]. Buffers consume significant dynamic power when traffic load is high as well as static power due to leakage. Figure 1(a-b) shows the total power breakdown (in mW) for a 5x5 router from Synopsys Design Compiler using the TSMC-LPBWP 40 nm technology library with a nominal supply voltage of 1.0 V and an operating frequency of 2 GHz. The dynamic power breakdown in Figure 1(a) shows that buffers consume 33% of router power (buffers+crossbar). With the same amount of buffer space, channel buffers can lower dynamic power by 90%. Figure 1(b) shows the leakage power breakdown of the router components in μW. As shown, the leakage power of the buffers consume 68% of the total router leakage power (buffers+crossbar). Channel buffers dissipate more leakage power than the register buffers; however, this increase is compensated by the very low dynamic power. Clearly, high buffer power is a problem in NoCs that needs to be addressed.

The next major concern in NoCs is reliability. The extreme shrinking of transistor feature sizes has made NoCs vulnerable to failures and data corruption. To examine the number of link faults in an NoC, a fault model was used which was similar to the model used in [12] in which a router design consists of 20,413 gates. Faults were injected randomly and weighted by the size of the gates. Therefore, gates with a larger number of transistors have a higher probability of failing. Figure 1(c) shows the number of faulty links caused by gate failures for reversible channel buffers (explained in Section III), non-reversible channel buffers, and conventional links without channel buffers. Non-reversible channel buffers are less reliable than conventional links due to the extra two transistors added to each link. Reversible channel buffers are even less reliable because of the eight additional transistors as explained in Section IV-A. Therefore, robust fault tolerant techniques are even more critical when using channel buffers.

In addition to power and fault tolerance, high network performance is another concern in NoCs. Looking at a NoC router, the amount of traffic entering and leaving the router

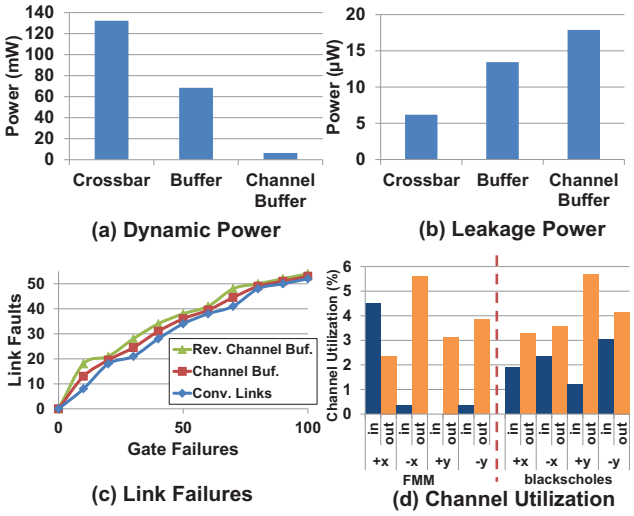


Fig. 1: (a-b) Dynamic and Leakage power of buffers, (c) link failures with and without channel buffers, and (d) link utilization of a router.

will be similar when averaged across the whole application. However, due to dynamic traffic patterns in NoC applications, there will be period of time where the majority of traffic will be either entering or leaving the router. This unbalanced traffic can cause certain links to become under-utilized during certain epochs. Figure 1(d) shows the link utilization of a router in a 64 core network for two real applications. Each side of the router (+x, -x, +y, and -y) has a link going in and out. For the applications FMM and blackscholes [27], [29], many links are under-utilized, thereby, wasting bandwidth. For example, on the +x side of the router, the "in" channel utilization is approximately double the "out" channel utilization. On other links, the "in" utilization is much lower than the "out" utilization. Using reversibility, links can change direction providing bandwidth where needed. Channel buffers can reduce dynamic power while marginally increasing leakage power; and reversible channel buffers could maximize resource utilization and improve execution time, but would need fault tolerant techniques to overcome the higher fault rates observed in channel buffers.

III. Quad-Function Channel Buffers

In this section, we will explain the circuit and implementation details of our proposed QFC buffers. Channel buffers have been shown to eliminate router buffer power by moving storage to the channels with the side benefit of reducing the area overhead with marginal performance penalty [9], [8]. In this work, we uniquely modify the previously proposed channel buffers to function as bidirectional channel buffers with similar advantages of reduced power while providing on-demand storage. Figure 2(a) shows two physical channels with four channel buffer stages per channel. The inset shows a conventional channel buffer which uses four transistors and a release (rel) control line to store or propagate packets in one direction. The working of channel buffers to either store or propagate packets based on router congestion and receive signals via a control block has been discussed previously [9]. The proposed reversible channel buffer circuit is shown in Figure 2(b). By adding eight transistors to act as four transmission gates, the channel buffers can propagate packets in both directions in addition to storage. The four transmission gates are controlled by the reverse signal (rev) sent from the router. A table showing all possible functions of the reversible channel buffer based on the inputs rel and rev are also shown in Figure 2(b). Figure 2(c) shows various combinations of reversible channel buffer functionalities; either as on-demand storage or repeater, and with data propagating either in forward or backward directions.

- **Forward Buffer:** When rel=0 and rev=0 data can be stored in the forward direction (left to right). The data is cut off from V_{dd} and GND and the data is stored on the capacitance of the transistors.
- **Backward Buffer:** When rel=0 and rev=1 data can be stored in the backward direction (right to left). Again, the data is cut off from V_{dd} and GND and the data is stored on the capacitance of the transistors.
- **Forward Propagation:** When rel=1 and rev=0 data can propagate forward. The transistors connected to V_{dd} and GND are enabled to allow propagation and the forward propagation transmission gates are also enabled.
- **Backward Propagation:** When rel=1 and rev=1 data can propagate backward. Again, the transistors connected to V_{dd} and GND are enabled to allow propagation and now the backward propagation transmission gates are enabled.

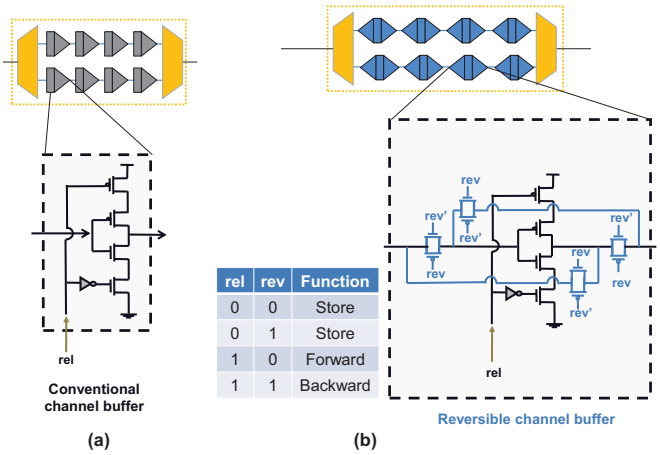
We show four functions of our QFC for high network loads (forward and backward buffers) and for low network loads (forward and backward propagation). When our QFCs act as buffers, the capacitance of the transistors must be large enough to store the data for many cycles. Figure 3 shows the discharge time of a channel buffer implemented with 130 nm transistors using the Virtuoso Analog Design Environment from the Cadence tools. As shown, the discharge time of the channel buffers is in the magnitude of milliseconds which corresponds to millions of clock cycles with a 1 GHz clock.

IV. QORE Architecture

In this section, we will describe the QORE architecture including reversibility, the design and operation of the fault tolerant network, the details of the fault and link controllers, the router microarchitecture, and proof of deadlock-freedom.

A. QFC without Faults

Conventional routers, that use virtual channels (VCs) and fixed connections between routers, can become a bottleneck if there is high traffic in any direction. To reduce the buffering bottleneck, QORE uses our reversible channel buffers to dynamically allocate buffers to adapt to traffic patterns. Figure 4(b) shows the links between routers in QORE. In order to have the same amount of buffering as a conventional 4 VC/input router, we place a set of N=4 links between routers



rel	rev	Function
0	0	Store
0	1	Store
1	0	Forward
1	1	Backward

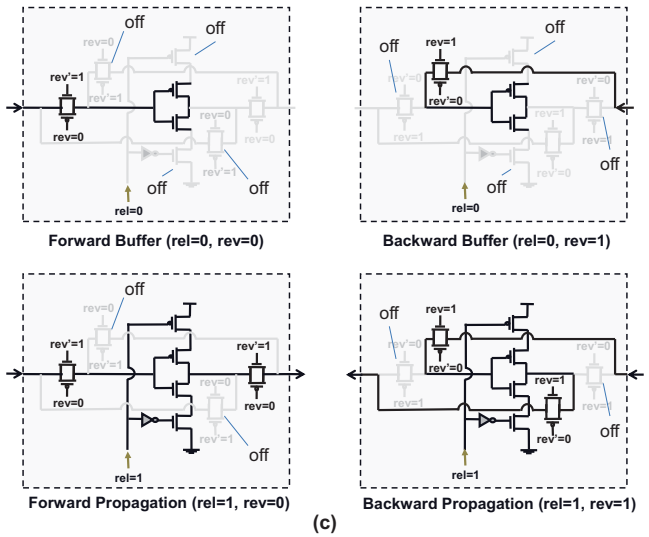


Fig. 2: (a) Conventional channel buffer, (b) our reversible channel buffer, and (c) storage and propagation for both forward and backward links.

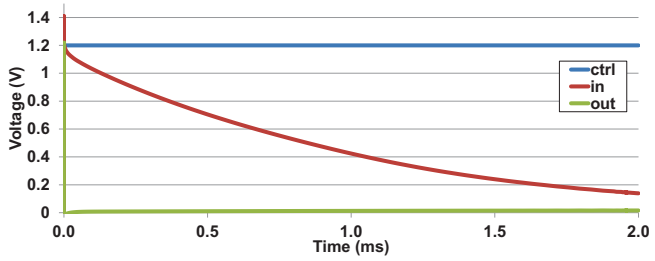


Fig. 3: Discharge time of channel buffer.

each consisting of two channel buffer lines. Each link consists of two channel buffer lines to alleviate HoL blocking [7]. Additionally, since QORE has more links between routers than the two links in conventional routers, we have reduced the bandwidth of our links for a fair comparison, as explained in the evaluation section. Therefore, the wire area overhead of QORE is equal to the conventional baseline networks. However, a designer can choose N to be a different number de-

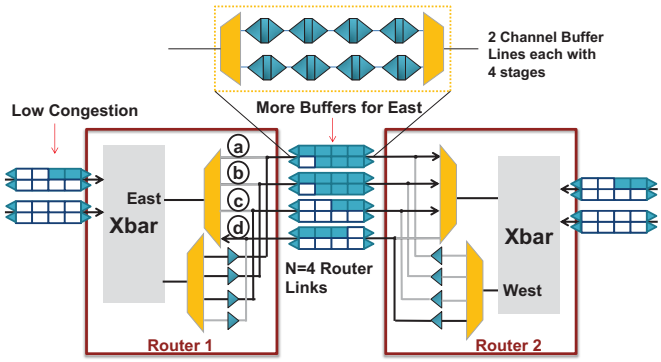


Fig. 4: QORE's four reversible router links each consisting of two channel buffer lines.

pending on system requirements. Each router link is reversible, allowing communication in both directions. However, the two channel buffer lines in each link will always be directed the same way. This will ensure that at any time, a packet will have at least two VCs to choose from, which in turn will alleviate HoL blocking. In QORE, when there is high traffic in one direction, the links can change direction according to the traffic load, thereby increasing buffer space. For example, in Figure 4(b), when there is high eastbound traffic, three links (a-c) can be allocated to the east direction while one link (d) remains in the west direction. The three east links can, therefore, use the under-utilized westbound buffers and provide more buffering for eastbound traffic. This additional buffering will relieve congestion at router 2 as well as router 1 and other upstream routers. Meanwhile, the one west link can still provide buffering for westbound traffic. As a result, both eastbound and westbound traffic can have ample buffering, thereby, decreasing packet latency. Therefore, reversing router links in QORE can reduce traffic bottlenecks caused by under-utilized links and buffers.

Determining which direction to allocate links is critical in QORE. Network traffic is measured using hardware counters to store the number of link traversals in each direction. A two-stage controller, which is detailed more in Section IV-C, is used to allocate links to the appropriate direction based on traffic demands. The first stage (link controller (LC)) of the controller uses the counters to determine which direction has the highest traffic called as the *"majority"*. The second stage (fault controller (FC)) will assign all but one link to the majority direction or allocate equal links to both directions if the link utilizations are similar. In the example in Figure 4(b), each time a flit traverses links (a-d), both routers 1 and 2 will increment their counters. Since there is high eastbound traffic in this example, the link controllers will determine that the majority of the traffic is moving from west to east. At this point, the fault controllers in both router 1 and 2 will allocate the first three links (a-c) to the east and allocate link (d) to the west. If there are packets currently stored in the channel buffers when the reversing occurs, then these packets will be flushed out to escape VCs inside the downstream router.

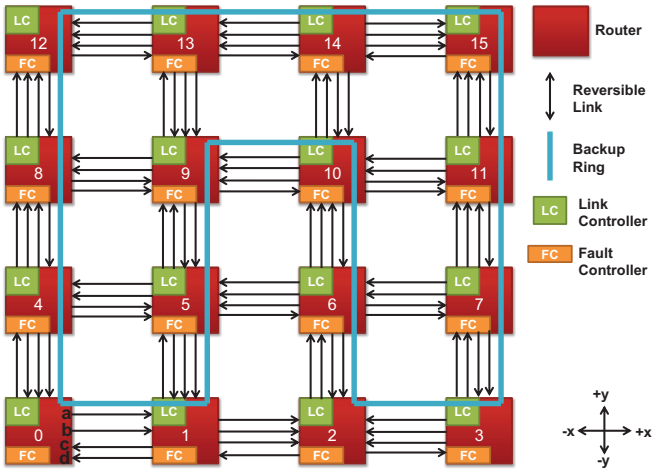


Fig. 5: Layout of QORE showing links configured to an arbitrary traffic pattern.

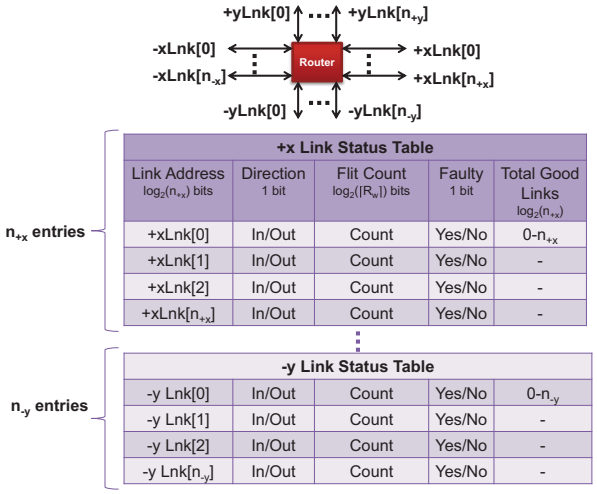


Fig. 6: Link status tables.

B. QFC with Faults

QORE uses QFC buffers to overcome hard faults in the network. When a link in one direction is faulty, another link can reverse its direction to overcome this fault. Figure 5 shows the overall layout of the QORE network for 16 routers and can be easily scaled to large numbers. The routers are connected to each other in a grid-like fashion similar to a mesh network. However, instead of the two unidirectional links between routers as in a mesh, QORE has four, narrower reversible links between each router. Again, each reversible link consists of two channel buffer lines. Also, the links are narrower than the baseline links as explained in the Evaluation section so there is no area overhead. The additional links create redundant paths between routers to improve both performance and reliability while avoiding HoL blocking. The link setup shown in Figure 5 is arbitrary; each link can reverse in either direction depending on traffic demands. QORE also has a backup ring network which is used when there are a large number of faults that potentially could isolate healthy routers. Each router has a link controller (LC) and a fault controller (FC) (Detailed in Section IV-C) that analyze link utilization and determine which links to reverse.

Each set of four links can handle up to three faulty links before using the backup ring. If a fault is detected in any of the links of a set, then the remaining non-faulty links will point in the directions specified by the LC and FC. For example, suppose the four links on the +x side of router 0 are initially setup as shown in Figure 5 with two links facing east (E) and two links facing west (W). If faults are detected in both links 0 and 1, then links 2 and 3 can overcome these faults by changing their directions to E and W, respectively. This will maintain connectivity between routers 0 and 1 so that packets can still be transmitted to both sides. If three of the four links fail then the fourth link can be used to communicate both ways since it is reversible. However, if all four links between two routers fail, then the backup ring network must be used. The backup ring network consists of two unidirectional rings, so that packets can traverse the shortest path, either clockwise or counterclockwise, to their destination. For example, if all four +x links of router 0 fail and the destination is router 5 then the packet will be routed on the ring network from router 0 to router 1, and so on up to router 5. Once a packet is on the ring network, it must stay on the ring network until it reaches its destination in order to avoid livelocks and deadlocks.

C. Link and Fault Controllers

In order to keep track of the status of each link, Link Status Tables (LSTs) are implemented in hardware. There are four LSTs per router in QORE; one for each set of links. The naming convention is shown in the top portion of Figure 6. The set of links on the right-side of the router are labelled as the +x links, links on the left are labelled -x links, etc. Each set of links has a LST containing information about the links. Each table has as many entries as links in each direction. In this paper, there are always 4 links in each direction. Hence, $n_{+x} = n_{-x} = n_{+y} = n_{-y} = 4$ and each table has four entries.

Each link in a specified direction has a unique identifier stored in the *Link Address* field. Whether the link is facing in towards the router or out away from the router is specified in the *Direction* field. This field will be read by the routing computation (RC) to determine valid routing paths and will be set up by the algorithm in the FC. The *Flit Count* data field stores the number of flit traversals on the link within the reconfiguration window, R_w. These counters are read by the LC to determine traffic demands. Each counter is incremented every time its corresponding link receives a flit and is decremented every time its corresponding link sends a flit. The *Faulty* data field stores whether or not the link is useable. This data field is read by the FC and RC. The field is set when its corresponding link detects a fault. Detection of faults can be done by implementing BIST (Built-In System Test) [14], [31]; however, fault detection is beyond the scope

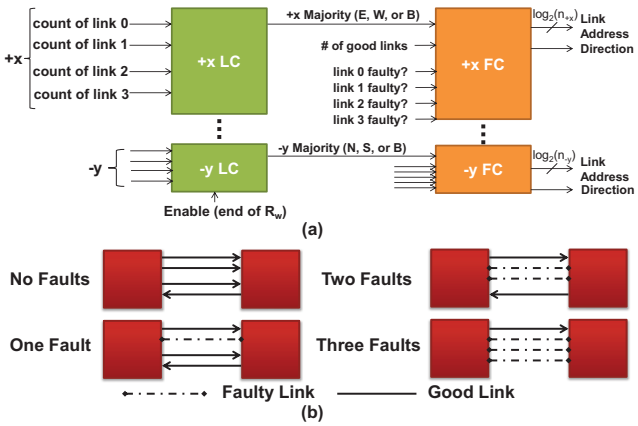


Fig. 7: (a) Block diagrams of link controller (LC) and fault controller (FC) and (b) Example of fault adaptability.

of this work. Finally, each table stores the total number of working links which is set each time a fault is detected.

The block diagrams for the LC and FC are shown in Figure 7(a). The LC and FC are split into 4 independent blocks corresponding to each direction (+x, -x, etc.). The inputs of the LCs are the direction fields for each of the 4 router links. The output of the LCs indicates which direction (N=north, E=east, S=south, W=west, or B=both) the majority of the flits were traveling during the last R_w cycles. If the traffic was roughly equal (within Δ where Δ=5% of total flit traversals in this paper) then a B is output and an equal number of links will face in each direction. The LC output gives a good measure on the traffic demand so that link bandwidth can be properly allocated. The simple algorithm to determine the majority of the +x (px) links is shown in Algorithm 1. At the end of R_w, the LCs total up the counts from their corresponding LSTs. Since the counters are incremented when a flit is received and decremented otherwise, a positive total would indicate the majority of the traffic is moving "West" for the set of +x links and a negative total would indicate more "East" Traffic. At the end, the counts in the LSTs are cleared for next R_w. The majority output is then fed to the FCs.

The inputs for the FC, shown in Figure 7(a), are the majority signal, the total number of good links, and the fault status of each link. The FC determines the new directions for each link by outputting their link address and updating the direction field in the LST. The algorithm to determine the directions for the +x set of links is shown in Algorithm 1. If the LC determines that the majority is W, then the FC will try to assign a majority of the links to the W direction as shown in Figure 7(b). The FC also tries to maintain connectivity by assigning at least one link to the opposite direction of the majority when possible. When there is only one non-faulty link, then the FC must break connectivity and assign the link to the majority direction. However, this will cause starvation as packets cannot be sent in one direction. We resolve this by allocating 60% of R_w to the majority direction and reserve 40% of R_w to the opposite direction. We chose 60% because our simulation results showed that this value gave the best average performance over all the benchmarks.

Algorithm 1 Link Controller and Fault Controller Pseudocode for +x (px) Links

```
// Link Controller
if(Enable){
    for(all links 0 to n_{+x} − 1)
        total_count = total_count + pxLnk[i].count;
    if(total_count ≫ 0)
        pxMajority = West;
    else if(total_count ≪ 0)
        pxMajority = East;
    else
        pxMajority = Both;
    clear_all_counts();
}
// Fault Controller
if(Majority of traffic is West){
    if(pxLnk.totalGood == 1)
        assign_one_link(West);
    else{
        assign_one_link(East);
        assign_remaining_links(West);
    }
} else if(Majority of traffic is East){
    // Same as above except interchange West and East
} else if(Traffic is similar in both directions){
    assign_half_links(West);
    assign_half_links(East);
}
```

D. Router Architecture

Figure 8 shows the router microarchitecture of QORE. The four links to the left of the router can act as outputs or inputs. When acting as an output, the data comes from the crossbar and is demultiplexed onto the four channel buffers. As an input, the data is multiplexed into the crossbar. After a signal is multiplexed, it is normally sent straight to the crossbar. However, it can be sent to an escape buffer. This escape buffer is used to move packets from the channel buffers when the links are reversed. They are also used to avoid deadlocking [32] as explained in Section IV-E. When the escape buffers are full the upstream router will receive a congestion signal and will not send packets to the channel buffers; therefore, guaranteeing that the escape buffers will have enough room to flush out the channel buffers. Six buffers are used because at most six channel buffer lines will face in one direction. Therefore, the six escape buffers ensures that a packet will have a buffer to go to when the links reverse. Each time a flit traverses the links, the counters in the LSTs are incremented or decremented based on the link direction. The inset in Figure 8 shows the counter for link 0. When a flit traverses a link, it signals the counters and increments the $flitcount$ in the LST if the direction is in or decrements the $flitcount$ if the direction is out. The LC and FC blocks access information from the LSTs as described in the previous section. The route computation (RC) is modified to determine which link to send data on in addition to which direction to send the packet. The

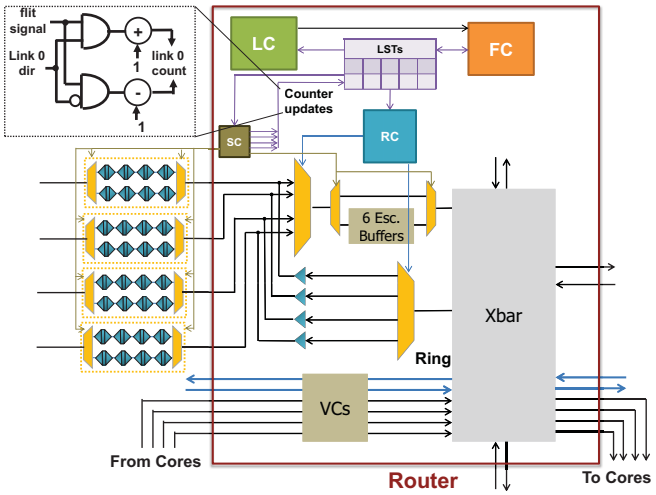


Fig. 8: Router Microarchitecture showing inputs/outputs, LC, FC, and RC.

link decision is based on which link has the lowest count in the LST. Therefore, the traffic will be spread evenly among the links. The switching control (SC) sends the release (rel) and reverse (rev) signals to the channel buffers. When there is contention at the crossbar or downstream router, the SC notifies the channel buffers to store the data by setting the rel signal to 0 as explained in Section III. The SC also reads the LSTs to obtain the rev signal, notifying the channel buffers of the correct direction.

E. Deadlock Avoidance and Reliability Concerns

In conventional NoCs, XY routing algorithm is used to avoid deadlocks by avoiding turns (Y-to-X). However when links reverse, if not handled properly, there is a potential for deadlock as communication in one direction can be cut off leading to starvation. In QORE, we avoid deadlocks by a) maintaining connectivity, thereby, eliminating starvation, b) using escape VCs to flush out channel buffers during reversing, and c) keeping packets on the backup ring network until their destination is reached. To prove that our network is deadlock-free we examine the three possible states of the N links between routers:

Case I: Zero to N-2 links are faulty. In order to prevent deadlocks, connectivity must remain between routers. FC algorithm 1 first assigns one link to the non-majority direction then assigns the remaining links to the majority direction. This ensures that there is always a connection in both directions. Conventional deadlock-free algorithms such as XY routing can, therefore, be applied and deadlocks are completely avoided.

Case II: N-1 links are faulty. Again, in order to prevent deadlocks, connectivity must remain between routers. However, in this case only one link is available. The algorithm of the FC will assign this one link to the majority direction. Then at 60% of R_w, FC will change the $Direction$ field in the LST to the opposite direction. This will cause the link to flush out the data from the channel buffers to the escape VCs located at the downstream router, thereby, allowing packets to be sent in the opposite direction. Therefore, 60% of R_w will be allocated to the majority direction and 40% of R_w will be allocated for the opposite direction, providing full connectivity.

Case III: All N links are faulty. In this case, no channel buffers are available and protocol states that packets must use the backup ring network to proceed. To avoid deadlocks and livelocks, packets must remain on the backup ring network until their destination is reached. We ensure the packet stays on the ring by adding a one bit $ring$ field to the packet that indicates to the RC whether or not the ring network should be used. When the ring bit is "1", the RC will always send the packet on the ring even if router links are available. If the ring bit is "0" then the router links must be used. To avoid circular dependencies once on the bidirectional ring, a separate set of VCs is allocated to each direction.

Protocol deadlocks can be avoided since each link has two buffer lines. One buffer line can be assigned to requests while the other is used for response traffic. Other than deadlocks and livelocks, another concern may be the issue of the fault tolerant components themselves failing such as the backup ring network or the fault controllers. The backup ring network adds redundancy to links between routers. Moreover, since this backup ring network does not use reversible channel buffers, it has 10 less transistors at every repeater creating a more robust connection between routers. For the LC, FC, and LSTs, since they have a very small overhead, as shown in Section V-A, these components would be ideal for dual modular redundancy (DMR) or triple modular redundancy (TMR).

V. Evaluation

In this section, we first consider the overhead for our reconfiguration controllers and reversible buffers. Next, we evaluate the fault tolerant performance of QORE compared to the Ariadne [12], Vicis [13] networks by evaluating throughput and power on synthetic traffic as well as speedup on real benchmarks. Lastly, we consider the effect of our reversibility on the overall performance of QORE when no faults are present by comparing to BAR [22] which is not a fault tolerant network.

For open-loop measurement, we varied the network load from 0.1-0.9 of the network capacity. The simulator was warmed up under load without taking measurements until steady state was reached. Then a sample of injected packets were labeled during a measurement interval. The simulation was allowed to run until all the labeled packets reached their destinations. All designs were tested with different synthetic traffic traces such as Uniform Random (UN), non-uniform random (NUR), Bit-Reversal (BR), Butterfly (BFLY), Matrix Transpose (MT), Complement (COMP) and Perfect Shuffle (PS).

For closed-loop measurement, the full execution-driven simulator SIMICS from Wind River [33] with the memory package GEMS [34] was used to extract traffic traces from real applications. The Splash-2, PARSEC, and SPEC CPU200

TABLE I: Cache and core parameters used for Splash-2, PARSEC, and SPEC2006 application suite simulation.

Parameter	Value
L1/L2 coherence	MOESI
L2 cache size/assoc	4MB/16-way
L2 cache line size	64
L2 access latency (cycles)	4
L1 cache/assoc	64KB/4-way
L1 cache line size	64
L1 access latency (cycles)	2
Core Frequency (GHz)	5
Threads (core)	2
Issue policy	In-order
Memory Size (GB)	4
Memory Controllers	16
Memory Latency (cycle)	160
Directory latency (cycle)	80

workloads were used to evaluate the performance of 64-core networks. Table I shows the parameters for the cache and core used for the Splash-2, PARSEC, and SPEC2006 benchmarks. We assume a 2 cycle delay to access the L1 cache, a 4 cycle delay for the L2 cache, and a 160 cycle delay to access main memory. The power and area results were estimated using the Synopsys Design Compiler with the 40 nm TSMC technology library.

For fair comparison, every network had 4 VCs per input and each network was assumed to have a concentration of four cores to a single router as this has been shown to minimize energy and latency while allowing a larger number of cores on a chip [35]. Additionally, we maintained similar bi-sectional bandwidths for each network. The conventional router design (both Ariadne and Vicis) will have two links between each router (one for each direction) and QORE has at most six links between routers (four reversible links, two unidirectional links for the ring) for a ratio of 1:3. However, the bandwidth of each link in QORE is 32 bits/cycle so the total bandwidth between routers will be 192 bits/cycle. Therefore, each link in the conventional design will be 192/2=96 bits/cycle which is 3X the bandwidth of a QORE link. We have assumed that the backup ring network is fault-free and the packet size is four flits each 128 bits.

A. Power, Area, and Timing of Reversibility Overhead

Table II shows the power overhead for the network components of one router estimated from the Synopsys Design Compiler with a nominal supply voltage of 1.0 V and an operating frequency of 2 GHz. A buffer for the baseline design is a a four flit register buffer and a buffer for QORE is a four stage reversible channel buffer. Each router, in either design, contains 32 buffers (4 inputs $\times$ 8 buffers). The buffers in QORE consume 19.8 mW of power; approximately 82.3% less than the baseline register buffers. The amount of leakage power for the reversible channel buffer was found to be 2.44 nW. The overhead of the LC and FC is approximately 96 nW of power and a timing of 0.07 ns. The power is a minimal fraction of the total router and the timing is within or clock period. The link power for both baseline and QORE are equal since the total link bandwidth is kept equal. An crossbar power overhead of 27.9% is due to the backup ring network in QORE leading to a slightly larger crossbar.

TABLE II: Power overhead for the components of one router.

	Baseline	QORE	Percent Diff.
Storage	111.6 mW	19.8 mW	-82.3%
LC	0	96.27 nW	-
FC	0	96.64 nW	-
Link	(2×96 bits) 307.2 mW	(6×32 bits) 307.2 mW	0%
Crossbar	(8×8) 67.4 mW	(9×9) 86.2 mW	+27.9%
Total	486.2 mW	413.2 mW	-15.0%

TABLE III: Area overhead for the components of one router.

	Baseline (μm^2)	QORE (μm^2)	Percent Diff.
Storage	43,712	147,392	+237.2%
LC	0	1.41	-
FC	0	1.42	-
Link	(2×96 bits) 23,629	(6×32 bits) 23,629	0%
Crossbar	(8×8) 580,007	(9×9) 622,418	+7.3%
Total	647,348	793,442	+22.6%

Table III shows the area overhead of each router component. The buffers in QORE occupy 147,392 μm^2 which is 3.4$\times$ more area than the baseline register buffers. However, unlike register buffers and conventional channel buffers, our channel buffers serve three functions: storage, reversibility, and a link repeater. The area overhead of the LC and FC components are approximately 1.4 μm^2 which is minimal compared to the other router components. The timing for our reversible channel buffers was estimated to be 0.39 ns which is within our specified clock period of 0.50 ns. The critical path of the four stage reversible channel buffers was composed of eight pass gates (0.22 ns) and four non-reversible channel buffers (0.17 ns). The timing of the critical path as well as estimate of power and area accounted for all additional wiring required between routers.

B. Speedup on Real Applications

The speedup of BAR (B), QORE (Q), Ariadne (A) relative to Vicis (V) for different real applications is shown in Figure 9. The networks were simulated on all applications; however, we only have space to show four applications in the figure. QORE reconfigures its links every $R_w = 50$ cycles. Different values of R_w are evaluated in Section V-F. Before runtime, faults were randomly inserted into a percentage of links ranging from 0% to 50%. Since BAR is not a fault tolerant network, it is only shown for 0% faults. At 0% faults, the performance optimized BAR has the largest speedup for all applications as expected. At low to medium faults (0-30%), QORE has an average speedup of 1.68$\times$ across applications for all benchmarks. At a high number of faults (40-50%), QORE has a worse speedup of 0.51$\times$ on average. However, this can be misleading because the high number of faults causes Ariadne

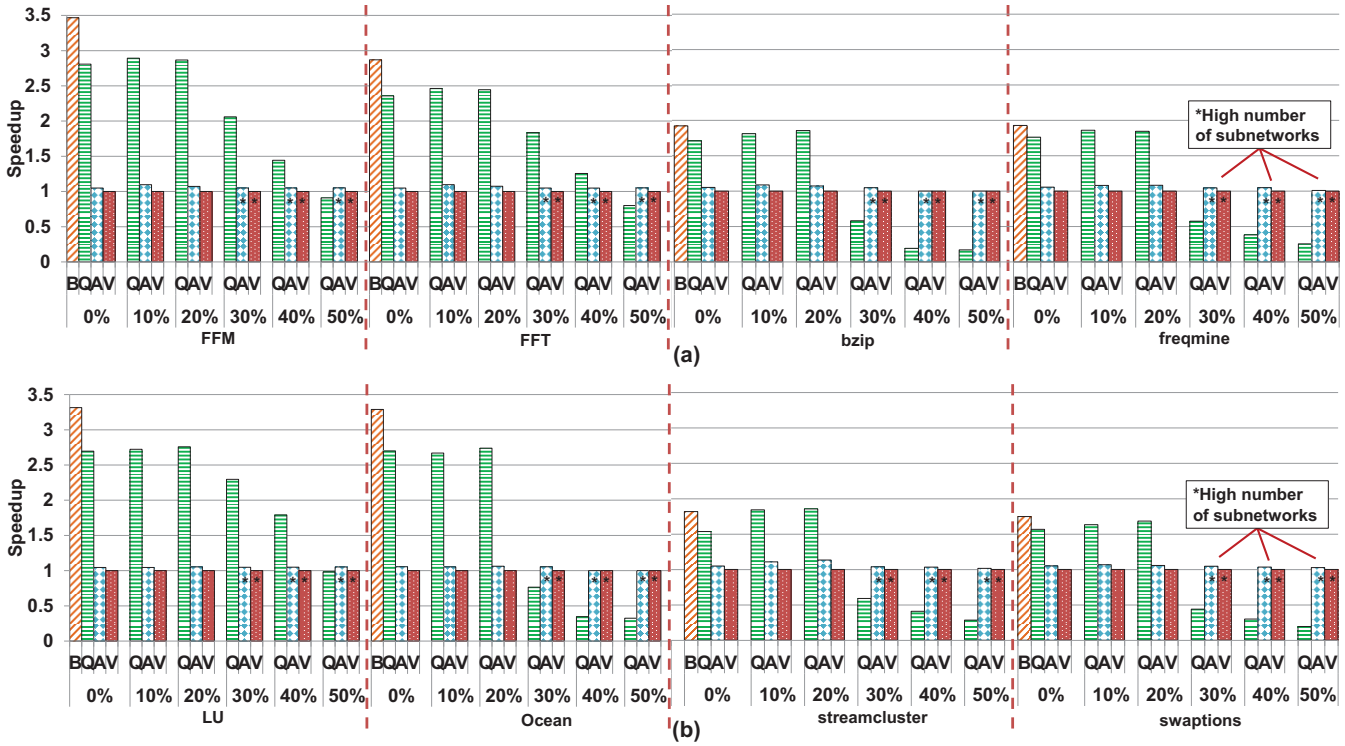


Fig. 9: Speedup of relative to Vicis with varying number of faults where BAR (B), QORE (Q), Ariadne (A) relative to Vicis (V) for 64 cores with SPLASH-2, PARSEC, and SPEC CPU2006 benchmarks.

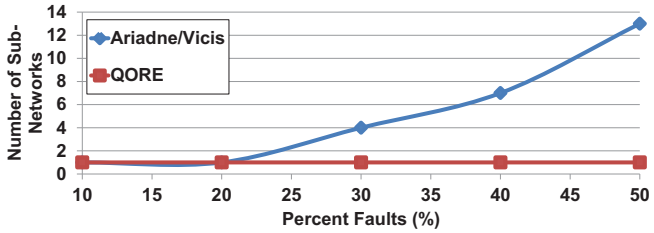


Fig. 10: Average number of subnetworks of QORE compared to Ariadne and Vicis.

and Vicis to be split into small subnetworks. Subnetworks are very undesirable because cores from one subnetwork will not be able to communicate with cores from another subnetwork. The average number of subnetworks for each network is shown in Figure 10. QORE always maintains connectivity through the backup ring network. The number of subnetworks in Ariadne and Vicis increase with the number of faults. Subnetworks partition the chip, blocking communication to many cores. The subnetworks, therefore, lead to a false increase in speedup as also observed in [12]. Whereas, the reversibility of links makes QORE more resilient to communication blocking.

C. Network Throughput with Faults

The saturation throughput of the networks for different synthetic traffic mixes is shown in Figure 11. Four different types of traffic mixes we examine are shown in Table IV using the abbreviations defined previously in Section V; However, due to space constraints we only show two mixes. Each mix randomly cycles through each pattern every *TP=250* cycles. QORE reconfigures its links every $R_w = 50$ cycles.

In Figure 11, QORE consistently has similar throughput to BAR and a higher throughput than both Ariadne and Vicis. Averaged over each traffic mix and fault percentage, QORE's saturation throughput is 2.3× and 2.9× higher than Ariadne and Vicis, respectively. Similar to the speedup results, an increase in throughput can be seen in all mixes when the fault percentage changes from 20% to 30%. Again, this is due to link faults causing the network to be partitioned into smaller subnetworks. When the faults increase to a high percentage (40-50%), few flits are sent on a network that has many cores, so the throughput (flits/cycle/core) starts to decrease again.

From 0-20% faults, QORE only sees a drop in performance of 3.5% averaged over all mixes compared to an approximately 70% drop for Ariadne and Vicis. QORE is able to sustain performance due to the adaptability of its links. When a wire between two routers is faulty in Ariadne or Vicis, then all communication between those two routers is blocked even if other wires are non-faulty. With many faults, this limits the number of paths in the network. Therefore, many packets are sharing the same paths which causes a drastic increase in contention for links. QORE, on the other hand, can overcome one or more faulty wires by reversing the available non-faulty links. Reversibility preserves paths between routers which relieves contention. Maintaining minimal contention for links is a main factor for maintaining high throughput.

TABLE IV: Traffic Mixes

Mix	Patterns
Mix 1	BR, BFLY, COMP
Mix 2	NUR, BR, PS
Mix 3	UN, BFLY, MT
Mix 4	UN, BR, COMP, PS

D. Packet Latency with Faults

Figure 12 shows multiple plots for the packet latency at various fault percentages for traffic mix 1. Latency plots at 10%, 20%, 40%, and 50% faults were not shown due to space constraints. At 0% faults, BAR saturates at the highest load due to its adaptability, and fine-grained flit transmission. The low load latency for both BAR at 0% faults and QORE for all faults is higher than both Ariadne and Vicis. This is due to the serialization delays combined with narrow links in BAR and QORE. However, QORE saturates at a higher load for most fault percentages. At 10%, 20%, and 30% faults, QORE saturates at least 77%, 160%, and 150% higher than Ariadne and Vicis. Faults in Ariadne and Vicis can easily shut down communication between routers. The fault tolerant schemes in these networks forces many packets to take additional hops to reach their destinations because they must move around routers. The increase in hop count greatly increases packet latency for the Ariande and Vicis networks. QORE is able to route more packets minimally to their destination to keep latency low. At 50% faults, Ariadne and Vicis saturate 87.5% higher than QORE. However, this is due to the many unreachable cores in Ariadne and Vicis which create very small subnetworks resulting in packets with little to no contention.

E. Network Power

The total network power for the networks is shown in Figure 13 for different numbers of link faults and two mixes, although we evaluated the network on all four mixes. Even though reversible channel buffers have a smaller power than register buffers, we have assumed that all the networks have the same buffer power of 618.5 nW and area of 4,606 μm^2 as shown in Table I. This is done to ensure that the no network has an unfair advantage due to a different buffer technology which trades off

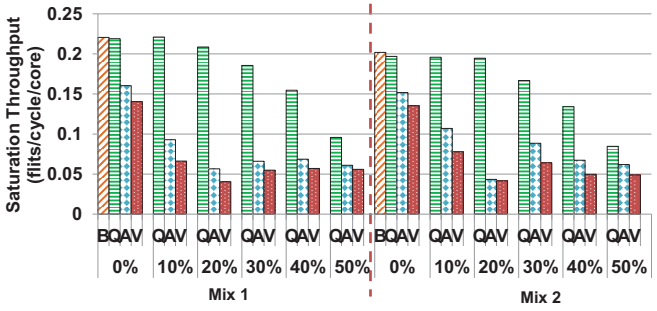


Fig. 11: Saturation throughput for varying percentage of link failures for different traffic mixes for BAR (B), QORE (Q), Ariadne (A) and Vicis (V).

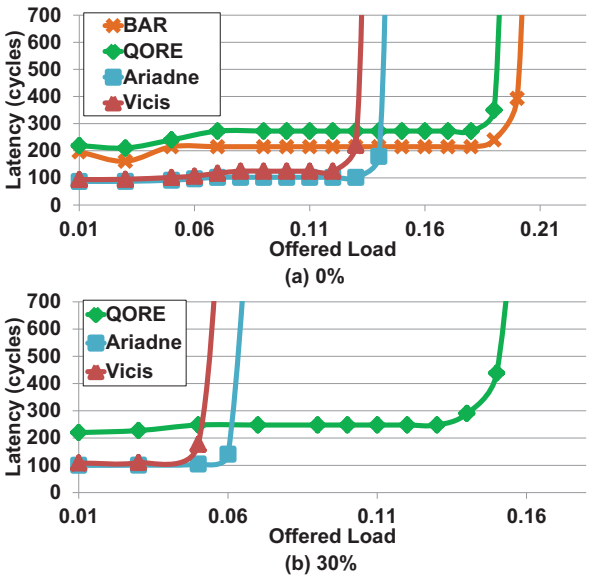


Fig. 12: Latency plots for traffic mix 1.

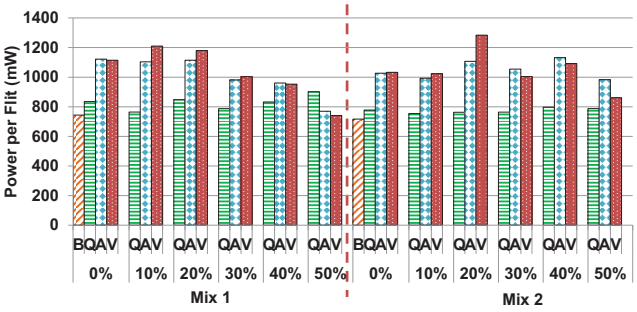


Fig. 13: Network power for different traffic mixes for BAR (B), QORE (Q), Ariadne (A) and Vicis (V).

area for power. For mix 1, QORE saves at least 15% power over Ariadne and Vicis on average. Additionally, QORE saves 25%, 22%, and 23% on traffic mixes 2, 3, and 4. The main contribution to the power savings is the link power. Ariadne and Vicis route around faulty links which many times leads to packets taking non-minimal paths to the destination. QORE can avoid this as long as there is one working link between routers. The only time QORE has the possibility to take a non-minimal path is when the backup ring is used which, in this simulation, only occurred when the fail percentage was 50%. As seen in Figure 13 for 50% faults, the power of QORE is higher than Ariadne because of the backup ring routes packets on non-minimal paths for this particular traffic mix. BAR has a power 9.5% less than QORE due to the backup ring in QORE which increases the crossbar size by one. However, when the number of faults increases, QORE cannot be compared to BAR since BAR is not a fault tolerant network. Therefore, QORE can save approximately 21% power on average while providing better fault coverage with a speedup 1.3× higher and improved throughput by 2.3×.

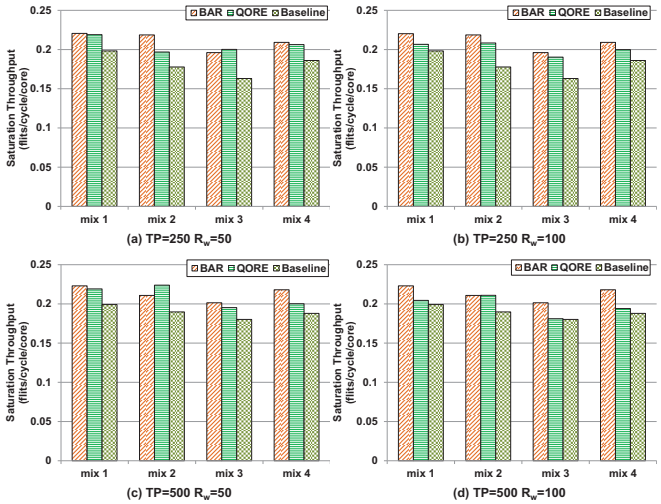


Fig. 14: Effect R_w on saturation throughput for varying traffic mixes.

F. Network Performance of Reversibility

We have shown that QORE can handle errors very well using reversibility. In this section, we will show that QORE can overcome faults while maintaining performance by comparing QORE to the non-fault tolerant, high performance BAR. In BAR, links are reconfigured every cycle ($R_w = 1\ cycle$). In this section, we evaluate the effect of a longer R_w on QORE as well as the difference between the two networks. Figure 14 shows the saturation throughput of QORE compared to BAR and a baseline network which is QORE without reversibility. In the first two parts of Figure 14, TP=250 is the same and R_w varies from 50 cycles in Figure 14(a) to 100 cycles in Figure 14(b). When $R_w = 50$ the saturation throughput of QORE is 2.5% less than BAR. When R_w increases to 100 cycles, QORE has less opportunities to reconfigure and the performance drop increases from 2.5% to 4.6%. In Figure 14(c) and 14(d) TP increases to 500 cycles and R_w changes from 50 cycles to 100 cycles again. In this case, the performance drop changes from 1.7% when $R_w = 50$ to 7.3% when $R_w = 100$. The uniform nature of mixes 3 and 4 give BAR a slight advantage over QORE since BAR reconfigures every cycle and at a finer granularity. Compared to the baseline, QORE can improve throughput by an average of 7.9% when $R_w = 100$ and the improvement can increase to 12.2% when R_w is changed to 50 cycles. Overall, when $R_w = 50\ cycles$ QORE performance is only 2.5-4.6% lower than BAR and 12.2% higher than the baseline, but has the additional benefit of being able to handle faults.

VI. Related Work

With the increase of soft and hard errors in NoCs due to decreasing technology sizes, much research has gone into the detection and handling of errors. Built-In Self Tests (BISTs) are commonly used to detect errors in systems. Recently, NoCAlert [14] was proposed which detected faults in real-time with 0% false negatives. Low overhead checkers were used to detect faults without the need of periodic or triggered-based testing.

As described in the introduction, the Ariadne [12] network uses up*/down* routing to move around faults. Each time a fault was detected, new routing paths were created by transmitting a series of flag broadcasts to all routers. This created a deadlock-free tree network for the irregular topology. The Vicis [13] network also changes its routing algorithm to move around faults when detected. To avoid deadlocks, turn restrictions are placed at certain routers. The Immunet [15] design avoids faults by adaptively routing packets while using escape VCs to avoid deadlocks. Our design differs from these previous works in that we try to avoid additional hops when possible by using reversible links. The implementation of reversibility eliminates the need for routing tables and multiple flag broadcasts to reconfigure the network as seen in Ariadne. Furthermore, in other networks, if one of the two unidirectional links fails then neither link can be used because this would create a one way path to a router. We mitigate this problem by using reversible links as opposed to unidirectional links. Therefore, as long is there is one good link to a router, communication will not be halted. The work in [36] uses bi-directionality in the two channels between routers to provide fault-tolerance to the links. QORE differs in that we move the buffers to the links and create reversible channel buffers to lower power and provide fault coverage to buffers. Additionally, we can provide higher fault coverage by implementing more than two links between each router.

In [22], a bandwidth-adaptive router (BAR) was created to increase channel utilization without affecting network latency. BAR increased channel utilization by using narrower channels while also improving performance through adaptive bidirectional channels. Our work also differs from BAR in that we reverse links as well as buffering by using reversible channel buffers. The reversing of buffers as well as links allows the downstream routers to store the increasing number of packets. Additionally, we reverse links/buffers at a coarser granularity to reduce serializer/deserializer overhead. Another reconfigurable design was proposed in BiNoC [21]. BiNoC dynamically reconfigured bidirectional channels to improve performance. We differ from BiNoC in that we reverse buffering as well as links to provide fault tolerance.

VII. Conclusions

With the decreasing technology sizes and increasing number of cores number integrated on a single chip, the design of fault tolerant NoCs that do not degrade performance is critical. In this paper, we propose QORE - a fault tolerant NoC architecture using reversible channel buffers. We use QORE's reversibility for increased performance and to overcome faulty links. Our results on real benchmarks (SPEC CPU2006, PARSEC, and SPLASH-2) show an increase in speedup of 1.3× and improved throughput by 2.3× on synthetic traffic compared to related work. Using the Synopsys design compiler, we show that QORE reduces network power by 21% while requiring minimal control overhead.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers as well as Prof. Savas Kaya for his feedback. This research was supported by the Stocker Research Assistantship and NSF awards ECCS-0725765, CCF-1054339 (CAREER), ECCS-1129010, ECCS-1342657, ECCS-1342702, CNS-1342984, CCF-0915537, and CNS-1318997.

REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on chips: A new soc paradigm," *IEEE Computer*, vol. 35, pp. 70–78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires," in *Proceedings of the Design Automation Conference (DAC)*, Las Vegas, NV, USA, June 18-22 2001.

[3] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, pp. 51–61, September/October 2007.

[4] J. Held, "Single-chip cloud computer: An experimental many-core processor from intel labs." Presented at Intel Labs Single-chip Cloud Computer Symposium, Santa Clara, California, Feb. 12, 2010.

[5] P. Kundu, "On-die interconnects for next generation cmps," in *2006 Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems*, Stanford, CA, USA, December 6-7 2006.

[6] G. Michelogiannakis, D. Sanchez, W. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, May 2010, pp. 9–16.

[7] A. K. Kodi, R. Morris, D. DiTomaso, A. Sarathy, and A. Louri, "Co-design of channel buffers and crossbar organizations in nocs architectures." IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2011.

[8] G. Michelogiannakis, J. Balfour, and W. J. Dally, "Elastic-buffer flow control for on-chip networks," in *Proceedings of the Fifteenth International Symposium on High-Performance Computer Architecture*, 2009, pp. 151–162.

[9] A. K. Kodi, A. Sarathy, and A. Louri, "ideal: Inter-router dual-function energy- and area-efficient links for network-on-chip (noc)," in *Proceedings of the 35th International Symposium on Computer Architecture (ISCA'08)*, Beijing, China, June 2008, pp. 241–250.

[10] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th annual International Symposium on Computer Architecture*, June 2007.

[11] M. Hayenga, N. E. Jerger, and M. Lipasti, "Scarab: A single cycle adaptive routing and bufferless network," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, December 2009.

[12] A. DeOrio, L.-S. Peh, and V. Bertacco, "Ariadne: Agnostic reconfiguration in a disconnected network environment," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2011, pp. 298–309.

[13] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009, pp. 21–26.

[14] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "Nocalert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *to appear in The 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012.

[15] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide, "Immunet: A cheap and robust fault-tolerant packet routing mechanism," *SIGARCH Comput. Archit. News*, vol. 32, no. 2, 2004.

[16] J. Kim, C. Nicopoulos, D. Park, N. Vijaykrishnan, and C. R. Das, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," in *Proceedings of the 33rd annual international symposium on Computer Architecture*, 2006, pp. 4–15.

[17] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Kilo-noc: a heterogeneous network-on-chip architecture for scalability and service guarantees," in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 401–412.

[18] S. Lin, J. Shi, and H. Chen, "Designing cost-effective network-on-chip by dual-channel access mechanism," *Journal of Systems Engineering and Electronics*, vol. 22, no. 4, pp. 557 –564, Aug. 2011.

[19] E. Carara, F. Moraes, and N. Calazans, "Router architecture for high-performance nocs," in *Proceedings of the 20th annual conference on Integrated circuits and systems design*, 2007, pp. 111–116.

[20] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006*, 2006, pp. 5–16.

[21] Y.-C. Lan, S.-H. Lo, Y.-C. Lin, Y.-H. Hu, and S.-J. Chen, "Binoc: A bidirectional noc architecture with dynamic self-reconfigurable channel," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 266–275.

[22] R. Hesse, J. Nicholls, and N. Jerger, "Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels," in *Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, May 2012, pp. 132–141.

[23] M. Hayenga and M. Lipasti, "The nox router," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 36–46.

[24] M. H. Cho, M. Lis, K. S. Shim, M. Kinsy, T. Wen, and S. Devadas, "Oblivious routing in on-chip bandwidth-adaptive networks," in *Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques*, 2009, pp. 181–190.

[25] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary, "Exploring concentration and channel slicing in on-chip network router," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pp. 276–285.

[26] S.-J. Chen, Y.-C. Lan, W.-C. Tsai, and Y.-H. Hu, *Reconfigurable Networks-on-Chip*. Springer, 2012.

[27] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, October 2008.

[28] J. L. Henning, "Spec cpu suite growth: an historical perspective," *SIGARCH Comput. Archit. News*, vol. 35, pp. 65–68, March 2007.

[29] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The splash-2 program: Characterization and methodological considerations," 1995, pp. 24–36.

[30] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "Vichar: A dynamic virtual channel regulator for network-on-chip routers," in *Proceedings of the 39th Annual International Symposium on Microarchitecture (MICRO)*, December 9-13 2006, pp. 333–344.

[31] J. H. Collet, A. Louri, V. T. Bhat, and P. Poluri, "Robust: a new self-healing fault-tolerant noc router," in *Proceedings of the 4th International Workshop on Network on Chip Architectures*, 2011, pp. 11–16.

[32] Y. Ho Song and T. M. Pinkston, "A progressive approach to handling message-dependent deadlock in parallel computer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 3, pp. 259–275, Mar 2003.

[33] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hllberg, J. Hgberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *Computer*, vol. 35, pp. 50–58, 2002.

[34] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Alameldeen, K. Moore, M. Hill, and D. Wood, "Multifacet's genreal execution-driven multiprocessor simulator (gems) toolset," *ACM SIGARCH Computer Architecture News*, no. 4, pp. 92–99, November 2005.

[35] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Proceedings of the 20th ACM International Conference on Supercomputing (ICS)*, Cairns, Australia, June 28-30 2006, pp. 187–198.

[36] W.-C. Tsai, D.-Y. Zheng, S.-J. Chen, and Y.-H. Hu, "A fault-tolerant noc scheme using bidirectional channel," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011.