Abstract— Network-on-Chip (NoC) architecture is considered to be an attractive solution to overcome the combined problems of limited bandwidth and scalability in multicores. Input buffering at the router allows the network to sustain the accepted throughput without performance degradation. However, the input buffers consume a substantial portion of the total power budget, and there have been proposals to reduce the size of these buffers. Eliminating buffers altogether can also reduce the power consumption at low network loads, however at higher loads when conflicts are frequent, deflecting or dropping packets can lead to higher power. In this paper, with both enhancing performance and decreasing power consumption as our goals, we propose a dual-input crossbar design called DXbar, which combines the advantages of bufferless networks to enable low-latency routing at low network load and limited buffering capability to handle excessive packets at high network load. Moreover, we also propose a unified dual-input crossbar that combines the bufferless and buffered approach in one integrated architecture. The dual crossbar network naturally provides fault tolerance and improves the reliability of the network. Dual-input crossbar architecture improves the area overhead, while providing similar performance as dual crossbar architecture. DXbar design not only has superior performance compared to the state-of-the-art designs based on similar motivation, but also achieves significant power savings. The simulation results of the proposed methodology show that DXbar achieves over 15-20% performance improvement and saves at least 15% power over the baseline design for synthetic and Splash-2 benchmarks. We further evaluated the performance by injecting varying percentage of faults into the network for both DOR and WF adaptive routing algorithms. Our results indicate that DOR outperforms WF adaptive routing algorithm at high network loads with increasing percentage of faults.

I. INTRODUCTION

Network-on-Chips (NoCs) are becoming a de facto standard for interconnecting multicores as they overcome the limited bandwidth and scalability of shared buses [1], [2]. NoCs are composed of links for communication and routers (buffers, crossbars) for switching and arbitration. It is well known that the input buffers contribute to a significant portion (~ 40%) of the total power budget in NoCs [3]. Recent research has proposed several techniques to reduce the power impact of input buffers which include re-organizing, contracting and eliminating the buffers altogether with either area overhead, performance penalty or increased complexity. Dynamically allocated virtual channel (VC) design as in ViCHAR [4] or on-link channel buffering as in iDEAL [3] provide similar performance with only half the number of input buffers (reduced power and area), but with increased complexity and latency penalty. Another approach utilizing channel buffering is the Elastic Channel flow control, which replaces all the repeaters with flip-flops, and eliminates the input buffers [5]. In this design, all packets are transmitted from one channel buffer to the next by a handshaking protocol, where the penalty is the link delays even if all the buffers are completely free between routers.

Bufferless routing is an unique approach which eliminates all input buffers without utilizing channel buffering. Flit-Bless [6] and the more recent CHIPPER [7], proposed a routing scheme to send all incoming packets to output ports, irrespective of the fact whether those output ports are productive. The age-based priority for arbitration indicates that the oldest incoming packet is guaranteed to be routed to its productive output port, while younger packets may be deflected to their non-productive output ports and take non-minimal numbers of hops before reaching their destinations. Another design which improves upon bufferless routing is SCARAB (Single Cycle Adaptive Routing and Bufferless Network), where packets are minimally-adaptively routed [8]. If none of the productive output ports are available, the packet will be dropped, and a NACK signal will be transmitted through a dedicated circuit-switched NACK network to trigger a retransmission, saving the bandwidth of the data network. Both designs can achieve substantial power saving and latency reduction at low network load. At higher network load, conflicts are more frequent and packets experience much more link and crossbar traversals, due to the increasing rate of deflections or retransmissions. This leads to significantly higher power consumption, and could exceed the network power budget. Recently, Adaptive Flow Control (AFC) [9] was proposed which dynamically switches between bufferless to buffered mode based on traffic load. At high load, the network behaves as buffered and at low load the network is bufferless. While this approach adapts based on traffic flow, it increases the design complexity as flow control should be adjusted on a per-router basis.

Clearly, bufferless networks are preferred at low loads using deflection and buffered network at high load to prevent re-routing of packets via deflection or re-transmission. Therefore, in this paper, we propose DXBar, a high-performance and power-efficient dual-input crossbar design targeting both performance enhancement and power reduction. In order
to take the advantage of power-efficient and low-latency bufferless networks for low network load, and cost-efficient conflict-handling capability at high network load, the design becomes a combination of a bufferless primary crossbar and a buffered secondary crossbar. At a low network load, almost all packets will only traverse through the primary crossbar and take minimal routes, experiencing similar delays of what the packets will experience in a bufferless network. Whenever there is a conflict, the losing packet will be buffered in the secondary crossbar, and wait until its desired output port is free. The introduction of the secondary crossbar enables blocked packets to be removed from the critical path of the primary crossbar, thus maintaining the continuous flow of packets and achieving low-latency. More importantly, with a secondary crossbar, the packets can be routed without blocking an incoming packet from the primary crossbar as a separate path is available for both packets. Further, we also propose a unified crossbar design that merges both primary and secondary crossbar while occupying lesser area than two crossbars and providing similar functionality. The major contributions of this work are as follow:

- We implement a bufferless network (primary crossbar) with deflection routing and a buffered network (secondary crossbar) with buffered routing. With bufferless network, packets take one cycle to be routed, thereby lowering the latency and without buffers, the power consumption is reduced. With the buffered network, we resolve conflicts by buffering at high network load, thereby eliminating extra hops and associated higher power via deflection.
- We also implement a dual-input single crossbar design that merges the advantages of bufferless and buffered architecture into a single unified architecture. Here, buffered as well as bufferless inputs can contend to the outputs simultaneously, unlike latches or bypass designs. Moreover, we ensure that the inputs can contend without conflicts to different output ports simultaneously.
- We test the reliability of the crossbar with varying percentage of faults and evaluate the throughput and power for both deterministic and adaptive routing algorithms.
- Cycle-accurate simulation on 8×8 2D mesh topology shows that DXbar has at least 20% performance improvement with both DOR (dimension-ordered) and WF (West-First minimal adaptive) routing; and 15% power reduction over the generic routers with synthetic traffic patterns and real-application traces from SPLASH-2 suite.

While [10] showed the power and performance saving using dual crossbar network, this work extends the design by unifying the buffer and bufferless crossbar microarchitecture into a single integrated NoC router. The area overhead of the proposed crossbar architecture is significantly reduced by combining the dual crossbar into a single NoC microarchitecture. Moreover, we also evaluate the reliability of the proposed dual crossbar network with different percentage of faults that could occur at the crosspoints connecting any input to output. Due to dual crossbars, we can tolerate almost 100% failure (equivalent to one crossbar failing at every router). We provide detail results with both DOR and WF adaptive routing algorithms and compare the throughput, power and latency for uniform traffic. This work differs from prior work in [10] as follows:

- We develop a dual-input single crossbar that provides similar performance as a dual-input crossbar with reduced area overhead. Moreover, our proposed conflict-free allocation permits packets from the same input ports to be simultaneously connected to the different outputs.
- We also evaluate the fault-tolerant design by implementing faults in the router microarchitecture. We test the throughput, latency and power consumption for the fault-tolerant crossbar architecture for both DOR and WF routing algorithms.

In section 2, we explain the DXBar dual crossbar architecture, the proposed single crossbar with dual inputs and fault tolerance; in section 3 we explain the performance of DXBar along with fault tolerance results and in section 4, we conclude the paper.

II. DXBAR: DUAL CROSSBAR ARCHITECTURE

Figure 1 illustrates the router architecture of DXbar, which is different from that of the baseline virtual channel (VC) based routers. There are two crossbars within each router, with the primary crossbar having four input ports, the secondary crossbar having five input ports, and both of them having five output ports. The four input links are connected to both the crossbars via de-multiplexers, and the injection port of the processing element (PE) is connected to the last input port of the secondary crossbar. All five output ports of the two crossbars are connected via multiplexers, which are then connected to the output links in the four cardinal directions and the receiving port of the PE. Four buffers are located in between the de-multiplexer and the first four input ports of the secondary crossbar. There are no buffers deployed between the injection port of the PE and the secondary crossbar. The buffer slots are connected serially, thus eliminating VCs and the corresponding virtual-channel allocator. Look-ahead signal, which carries the routing information of the incoming flit, is connected directly to routing computation, which is then connected to the switch allocator. Switch allocator is modified to control the de-multiplexers, the crossbars, and the multiplexers to maintain the correct packet flow in both crossbars.

A. Router Design

Figure 2(a) shows the 5-stage router pipeline of the baseline design. The stages of router pipeline are as follows: 1) buffer write (BW) and routing computation (RC) for head flits, 2) virtual-channel allocation (VA) and switch allocation (SA) which are performed in parallel, 3) switch traversal (ST). An extra link traversal (LT) stage is needed to transmit a flit to the downstream router. Figure 2(b) shows
the reduced router pipeline of the baseline design using look-
ahead routing. RC is done at the previous router, and the
result is written to the flit and transmitted to the current
router so that no RC is needed for switching the flit to the
next router, reducing the number of stages from 5 to 4. Figure
2(c) shows the further reduced router pipeline of the baseline
design via speculation. Speculative SA is performed in order
to finish both VA and SA in the same cycle. This reduces
the number of stages from 4 to 3, and this pipeline is used
as the baseline design in this article. Figure 2(d) shows the
pipeline of DXbar router. By using look-ahead signal, the
routing information can traverse the link one cycle ahead
of the flit, and routing decision can be made before the
flit actually arrives [11]. As a result, other than the source
node, RC stage does not occupy a dedicated cycle. Moreover,
the elimination of VCs and buffers enables the removal of
BW and VA stages. Therefore, SA/ST is the only stage in
the router pipeline of DXbar. The simplification of router
pipeline, which is possible by eliminating buffers in the
primary crossbar, reduces the number of cycles to switch a
flit from three to only one. This proposed pipeline is similar
to those used in Flit-Bless [6] and SCARAB [8]. These two
bufferless designs both use only ST stage to switch packets,
and use another LT stage to transmit packets.

The flits arriving at the de-multiplexers are called incom-
ing flits. When an incoming flit arrives, it would compete
with other incoming flits for the output ports, using the
routing decisions made by the router one cycle ahead. If
the flit wins in the arbitration, it could traverse the primary
crossbar and go to its designated output port. Otherwise,
it would be sent to the secondary crossbar and be buffered.
Figure 1 shows the timing of an incoming flit which wins in
arbitration and traverses the primary crossbar. Flits waiting
at the head of buffers and in the injection port are called
buffered flits. Those flits could compete in arbitration as well,
but they have lower priority than incoming flits. Once a flit
wins in arbitration, the look-ahead signal leaves the router
and is transmitted to the downstream router, while the flit is
traversing the crossbar. The de-multiplexers and multiplexers
are set to select the primary crossbar inputs/outputs by
default. Switch allocator uses the result provided by the
arbitration to not only set the two crossbars to connect
input ports and output ports correctly, but also control all
of the multiplexers and de-multiplexers to link each output
port to the correct crossbar, and to change the direction
of incoming flits between the primary and the secondary
crossbar, respectively. In our design, each flit of a packet
has to be a head flit as it is possible to receive out-of-
order flits. While this can require some extra computation
and power, the re-assembly of the flits can be accomplished
by the cache controller that contains a Miss Status Holding
Register (MSHR) which maintains an entry for each cache
miss and waits for the entire cache block to be received [7].

1) Walkthrough Examples:: Figure 3 shows several pos-
sible scenarios to illustrate how flits are switched in DXbar
design. Figure 3(a) shows the scenario when there is no
conflict among packets. Flits from the x+ and the x- want
to go to the x- output port and x+ output port, respectively.
At the same time, flits from the y+ and the y- want to go
to the y- output port and y+ output port, respectively. All of
the four incoming flits are switched simultaneously, and the
network operates similarly to a bufferless networks, because
no packet is buffered and latency is minimized. This is the
best case scenario. Figure 3(b) shows the scenario when
conflict occurs. A flit from the y- input port and another flit
from the x+ input port compete for the x- output port. Assume
the flit from x+ is older, then that flit is switched from
x+ input port to the x- output port through the primary
crossbar. The flit from the y- input port loses in arbitration,
and the y- input de-multiplexer receives a control signal from
switch allocator, routing the flit from the y- input port to the
secondary crossbar and buffering it. Figure 3(c) shows the
scenario when another flit arrives from the y- input port in
the next cycle. Because the flit arriving from the y- input port
one cycle ago is currently buffered in front of the secondary
crossbar, the path from the y- input port to the primary
crossbar is unoccupied. Therefore, the incoming flit in the
current cycle can be switched to the desired output port. Otherwise, it needs to wait in buffer until the previous packet is switched. The figure also illustrates how the injection ports work. As the injection ports have the same level of priority as buffers, a flit can be injected whenever the desired output port is not occupied. Figure 3(d) shows the scenario when no more flits from the input ports need the x-output port. The buffered flit can then traverse the secondary crossbar and be routed downstream. At the same time, a flit from the y-input port can traverse the primary crossbar to a different output port. This is not feasible in other designs, because there can be only one flit coming from one particular input port at any cycle.

DXbar uses buffered secondary crossbar to enable bufferless routing in the primary crossbar, without deflecting or dropping packets. More importantly, the buffered flits could use the secondary crossbar to go to any unoccupied output port. Being different from simple buffer bypassing, this dual-crossbar design does not have contention between bufferless routes and buffer slots for the same input ports of the crossbar. As a result, buffered flits increase the probability to win in arbitration. This design further facilitates the implementation of minimal adaptive routing to increase the output port occupancy. The removal of VCs could potentially decrease the performance, and prevent the network from providing Quality-of-Service (QoS) for different traffic classes, or utilizing some of the fully adaptive routing algorithms which need extra VCs to prevent deadlocks. However, the purpose of deploying buffer slots is to handle conflicts in a bufferless network without deflecting or dropping to save power. Moreover, the extra crossbar increases the speed of progressing buffered flits. Therefore, we believe that the proposed design captures the best of bufferless and buffered networks by dual crossbars.

2) Fairness Maintenance:: Because the arbitration is age-based, flits coming from the nodes on the edges of the mesh network will always have higher priority when they pass through the nodes in the center. As a result, the flits injected by center nodes are more likely to lose in arbitration and traverse the secondary crossbar. A fairness issue thereby arises, since flits can be stalled in the injection ports or buffers for long cycles, when the desired output ports are taken by other flits from the input ports. In order to maintain fairness between the primary crossbar and the secondary crossbar, a fairness counter is maintained in each router to count the number of times flits from the primary crossbar win consecutively in arbitration. The counter works only when there are flits waiting in the buffers or in the injection port, and it is reset every time a waiting flit wins. Once the number exceeds a threshold, the priority flips so that flits from the secondary crossbar and the injection port have higher priority, so that they will be assigned output ports ahead of the flits from the input ports. Although the continuous flow of flits on the primary crossbar is interrupted, the nodes in the center are able to inject new flits even if the network is heavily saturated, and the buffered flits are guaranteed to leave in a finite number of cycles. Setting the threshold too small can lead to difficulty covering the round-trip delay of credits, while setting the number too large does not help to solve the fairness issue. After testing with different traffic patterns, the threshold is set to four to obtain

Fig. 3. The illustration of dual xbar: (a) No conflict appears, the network operates as bufferless network. (b) Conflict appears, one flit is sent to the secondary crossbar and buffered first. (c) The flit arriving after the buffered flit sees no delay and can proceed without being buffered, thus ensuring no instant back pressure. The injection port can send a flit whenever the desired output port is not occupied. (d) Even though there is still a flit arriving at the same input port, the buffered flit can proceed when the desired output port is not occupied by flits from the input ports. The incoming flit can proceed simultaneously, encountering no delay.
the best performance.

B. Dual Input Single Crossbar

While dual crossbars allow multiple paths from the same inputs, they occupy more area. One approach is to design directional-based smaller crossbars for x and y directions as in [12], [13] by increasing the input speedup. One potential problem with these approaches is that the same input cannot be directed to different direction when the opportunity may arise. For example, if the routing algorithms decides the direction is x+ and the output port is occupied, then the packet will be buffered. However, similar situation occur in the next consecutive cycle, the packet will still be buffered as the crossbar directions are only x+ or x-. However, in our dual crossbar approach, we can implement adaptive routing by re-directing the buffered flit to another progressive direction y+ or y-, thereby improving performance. Therefore, in order to achieve similar flexibility, we design a unified single crossbar with conflict free allocation to reduce the area overhead.

Figure 4(a) shows the unified crossbar. This crossbar is constructed by placing transmission gates between output lines of a matrix crossbar. These transmission gates allow or block an electrical signal from crossing from one side to the other. For example, if a high voltage signal is placed on the transmission gate, there is a conduction path from one side to another. On the other hand, if a low voltage signal is placed on the transmission gate, the electrical current is blocked creating a segmentation of the crossbar input. By correctly controlling these transmission gates, it is possible to segment the matrix crossbar to allow for multiple flits from the same input port to traverse the crossbar. In Figure 4(b), an example of multiple flits traversing the crossbar at the same time is shown. From the figure, I0 (from primary bufferless input) has one flit traversing the crossbar to O2 and \( \Gamma_0 \) (from the secondary buffered input) also has another flit traversing the crossbar to O3. This is accomplished by having the transmission gate off between the two output ports and all the other transmission gates along the input to be on.

1) Switch Allocator Implementation: As each input port has the potential for two different packets traversing across a crossbar from buffered and bufferless inputs, the standard switch allocation found in most routers needs to be augmented. In a separable output-first switch allocator, flits will proceed through two stages of arbitration [14]. During the first stage, all output ports requested from the two input ports are combined together (OR logic) into a P bit value, where each bit corresponds to an output port. For example if the first input (I0) requires the first output port, it will place a high signal of the first bit. Then the P bits from each input port are routed to the correct P:1 arbiters. Next, each P:1 arbiter independently selects which input port is granted the right to traverse across the crossbar to the given output port. Afterwards one bit from each of the P:1 arbiters are combined together and move towards the second stage of each input port. In the second stage, the output ports from each input port who had previously won, compete among the multiple flits inside each input port to see who will traverse the crossbar. To accomplish this, the P bits are logically AND’ed and then OR’ed with the requesting input port to see if there is a match. If there is a match, the corresponding bit for the (selected) requesting flit will be high and will proceed to the V:1 arbiter. Finally, the V:1 arbiter will select a flit to traverse the crossbar. It should be mentioned from the figure, unified crossbar will have a value of 5 for p and a value of 2 for V as dxs has 5 input/output ports and 2 incoming flits (I0 and I0’). We add another V:1 arbiter in series with the first arbiter. This second V:1 arbiter is used to select an additional packet for a different output port if the given input port was granted to two or more output ports. The reason the second V:1 arbiter is designed in series and not in parallel with the first V:1 arbiter is to prevent the second arbiter from selecting the same input port as the first arbiter. By having the second arbiter in series, we can have the output from the first arbiter be a selection vector for the second arbiter which will prevent the second arbiter from selecting the same input port.

2) Conflict Free Allocator: Each of the two V:1 arbiters can select a combination of output ports that will cause a conflict. For example (Figure 4(c)), the first V:1 arbiter for input 1 can select output 4 and the second V:1 arbiter can select output 2. As this creates a conflict, only input 1 will have one packet that will traverse the crossbar. To compensate for these situations, we add extra logic after the switch allocation to detect if a conflict arises. If a conflict arises, we switch the two packets such that the packet originating from I0 is switched to \( \Gamma_0 \) input and vice versa, thereby enabling forward progress by both the packets.

Figure 4(c) shows the logic used to evaluate and detect a conflict between the two inputs from the same port. From the figure, the conflict detection logic is divided into two different stages. In the first stage, conflict detection takes place by having the two selected output ports from the two inputs (I0 and \( \Gamma_0 \)) enter the detection logic circuitry. During the detection logic circuitry, the I0 input bits are compared to the \( \Gamma_0 \) output bits. If the input I0 is requesting an output port that has a higher input port than the input from \( \Gamma_0 \) input
port, one of the AND gates will go high and will propagate through the OR gate to the switch logic. After the detection logic, the signal will be an input for four multiplexors which will select the correct conflict free combination. The single crossbar design has more overhead; power as well as latency for additional logic. However, single crossbar design with dual input can provide consistently better performance and different routing algorithms can be easily implemented due to full connectivity.

**C. Fault Tolerance**

In this section, we evaluate the fault tolerance of the proposed dual-crossbar network. While similar discussion could be possible for dual-input single crossbar, we limit our studies to understand the effect of failure of one crossbar within the router. The duplication of crossbar enables hardware-level fault tolerance. The router could avoid being completely disabled, even in case of a permanent crossbar failure. When a crossbar failure is detected, switch allocator signals all of the input de-multiplexers to direct incoming flits to the buffer slots. The router then functions similarly to the routers of a buffered network. Because both the primary and the secondary crossbar may fail, there should be a mechanism to direct the buffered flits to the working crossbar. Therefore, a set of $2 \times 2$ crossbars, controlled by the signals from the switch allocator, are deployed between the buffer slots and the crossbars. A 1-bit signal could control these small crossbars, which indicates that the required modification of switch allocator is minimal. As a result, flits could traverse the router through the buffer slots and the working crossbar, even if the other crossbar does not function. This functionality is enabled in the dual-crossbar architecture to switch the input from one crossbar to the other. It should be noted that the fault detection is not implemented along with fault tolerance in this work. Fault detection can be enabled by a BIST (built-in-self test) circuits that test the input/output periodically or at fixed intervals to determine if there are any faults [15], [16]. Such circuits will consume power, area and latency when enabled, however this work does not consider the working of the BIST circuit. In order to determine the performance degradation of DXbar when running with crossbar faults, the fault detection mechanism is assumed to be available and works without affecting performance.

**III. PERFORMANCE EVALUATION**

In this section, we evaluate the DXbar design in term of area overhead, timing, energy, and overall network performance.

**A. Simulation Methodology**

We compare DXbar to Flit-Bless and SCARAB designs as they resemble closely to our work with identical router pipeline latencies. Moreover, we also compare our design to a generic router with 4 flit buffers connected as FIFO called as Buffered 4 and two sets of 4 flit buffers called Buffered 8. The split design resembles DXbar only at the buffering and provides for a fair comparison by removing Head-of-Line (HoL) blocking. The pipeline stages for the baseline are 3 (RC, SA/ST and LT) and for our proposed design as well as deflection networks are 2 (SA/ST and LT) due to look-ahead signalling. Two different routing algorithm are considered for DXbar, dimensional-order routing (DOR) and west-first adaptive (WF) routing, which are called DXbar DOR and DXbar WF, respectively. The proposed crossbar designs, buffer and links were synthesized and optimized using the Synopsys Design Compiler tool using the TSMC 65 nm technology library at a nominal supply voltage of $1.0 V_{dd}$, and 128 bits flit size.

We use a cycle-accurate NoC simulator to perform the detailed evaluation of DXbar. The network load varies from 0.1 to 0.9 of the network capacity, and packets are injected according to the Bernoulli process based on the given network load. All the designs are simulated to run nine different synthetic traffic traces, which are Uniform Random (UR), Non-Uniform Random (NUR), Bit Reversal (BR), Butterfly (BF), Complement (CP), Matrix Transpose (MT), Perfect Shuffle (PS), Neighbor (NB), and Tornado (TOR). NUR creates hot-spot scenarios by injecting 25% additional traffic to a select group of nodes. Moreover, we collected SPLASH-2 traffic traces using the full system simulator Simics [17] with the GEMS memory module enabled for accurate cache coherence and memory access latencies [18]. The processor model used in our simulations is based on the Ultra SPARC architecture. Each processor can issue two threads and has it own 64 KB private L1 instruction and L1 data cache. Each processor has its own private 1 MB L2 cache and uses the MESI protocol to maintain cache coherence. The following benchmark and input parameters are used for the select SPLASH-2 applications: FFT (16 K), LU (512 × 512), radixsort (largeroom), Ocean (258 × 258), Raytrace (Teapot), Radix (1 M), Water (512), FMM (16K) and Barnes (16 K).

**B. Energy and Area Estimates**

Table 1 shows the area and buffer energy estimations for each design obtained from Synopsys. The $5 \times 5$ crossbar area is 0.033 mm$^2$, the area of four buffers is 0.0037 mm$^2$ and the area of four input links is 0.173 mm$^2$. DXbar occupies more area than the buffered 4 design because of

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>3</td>
</tr>
<tr>
<td>Issue</td>
<td>2, in-order</td>
</tr>
<tr>
<td>Retire</td>
<td>in-order</td>
</tr>
<tr>
<td>Ld/St units</td>
<td>1</td>
</tr>
<tr>
<td>Mul/Div units</td>
<td>1</td>
</tr>
<tr>
<td>Write-buffer entries</td>
<td>16</td>
</tr>
<tr>
<td>Branch predictor</td>
<td>13-bit GHR, 2,048 10-bit BHRs, 8,192-entry</td>
</tr>
<tr>
<td>(hybrid of GAg+SAg)</td>
<td>2,048/32</td>
</tr>
<tr>
<td>BTB/RAS/entries</td>
<td></td>
</tr>
<tr>
<td>IL1/DL1 size, associativity</td>
<td>64 KB, 4-way</td>
</tr>
<tr>
<td>IL1/DL1 access latency</td>
<td>2 cycles</td>
</tr>
<tr>
<td>IL1/DL1 block size</td>
<td>64 B</td>
</tr>
</tbody>
</table>
the secondary crossbar. However, DXbar consumes less area than the buffered 8 design because the buffers have a larger area than the crossbar. The unified crossbar design occupies less area than DXbar because the crossbar is smaller than two 5 x 5 crossbars. For each design, the crossbar energy is 13 pJ/flit and the link energy is 36 pJ/bit. The unified crossbar consumes a slightly higher energy of 15 pJ/flit due to transmission gates. The table shows the buffer energy for each design. Buffered 8 consumes the most energy due to more buffers whereas Flit-bless and SCARAB have no buffers, therefore consume no buffer energy. The critical path for each design is the link traversal (LT) stage which is 0.47 ns (obtained from Synopsys). The longest path in the switch traversal for the unified crossbar is when all 5 transmission gates must be switched which takes 0.27 ns. Both these values are under the targeted 1 ns clock cycle. The DXbar design occupies 33% more area than Flit-Bless and SCARAB due to additional crossbar and the unified crossbar design occupies 25% more area. However, both these designs are much closer to the baseline designs of buffered 4 and 8. It should be noted that the energy values reported below are only for the DXbar design as the unified design is marginally more.

C. Synthetic Traffic

Figure 5 shows the throughput plot for UR. DXbar is significantly better than bufferless designs and buffered networks. DXbar DOR brings the best performance in terms of throughput and latency, and has a saturation point at over 0.4. This has a 20% improvement over buffered 8, and a 40% improvement over buffered 4, Flit-Bless, and SCARAB. DXbar WF is slightly worse, but it is still able to completely outperform all other designs, and it has a saturation point at around 0.38. This equals a 15% improvement over buffered 8, and a 38% improvement over buffered 4, Flit-Bless, and SCARAB. Even though the buffer size is twice as many as DXbar, buffered 8 still cannot outperform. On the other hand, Flit-Bless and SCARAB perform similarly, but their saturation points are below 0.3, which are lower than DXbar.

Figure 6 shows the energy plot for UR. Even though buffered 4 has the same buffer organization as DXbar, it uses buffer every time and consumes more energy than DXbar. Buffered 8 has a buffer organization which consumes more energy. Therefore the energy requirement of buffered 8 is higher than buffered 4. Flit-Bless and SCARAB use as little energy as DXbar does at zero load. However, they consume higher energy when the offered network load goes near or beyond their saturation points. SCARAB has an increase of about 2X, and Flit-Bless even has an increase of about 3X. This indicates that at high offered network load, the way bufferless networks handle conflicts sacrifices energy consumption, and they actually consume more energy than the generic buffered router does when offered network load increases. Nevertheless, the energy consumption for DXbar
hardly changes when the offered network load increases, and the only little difference comes from the fact that a portion of packets are buffered, but only when they are relatively new. The result from simulation shows that the chance for the packets to be buffered while traversing through a router is only 1/6 after saturation point. This gives DXbar a huge advantage on energy saving, compared to not only bufferless networks, but also generic routers, which put packets in buffers every time.

Figures 7 and 8 show the throughput and power for all synthetic traffic patterns at an offered load = 0.5. It is clear that for UR, NUR, CP, and TOR, DXbar DOR performs the best. For BR, BT, MT, and PS, which all favors adaptive routing algorithms, DXbar WF is very competitive. One significant advantage of the proposed design which includes both dual and unified crossbar designs, is the ability to adapt the network dynamically and achieve higher performance for adverse traffic pattern. While other dimensionally split crossbars can achieve lower power consumption, dynamically changing the direction is difficult and will need more hardware. The proposed dual and unified crossbars can achieve adaptability on buffered flits which can provide higher performance. DXbar uses the least power, while Flit-Bless uses the most, SCARAB the second, and the generic routers lie in between.

D. Real-Application Traces

Figures 9 and 10 show the normalized execution time and energy for all SPLASH-2 traces. The results show that DXbar DOR performs better than DXbar WF. For most of the traces, DXbar can achieve the best performance. In this simulation, Flit-Bless and SCARAB can really keep up with DXbar for some of the traces, and can even do slightly better for FFT. Flit-Bess and SCARAB consume at least 16X and 2X higher than the energy consumption of DXbar, respectively. In order to let packets proceed, Flit-Bless deflects for an average of almost 50 times per packet transmitted, and SCARAB drops packets at a high rate. This results in a higher energy consumption, but also relieves congestion at some locations. Therefore, DXbar design is able to provide superior performance and decrease energy requirement simultaneously. It must be noted that both Flit-Bless and SCARAB designs have shown to have higher performance at lower network load for real applications as originally proposed. This is true as the processors do not generate significant L2 misses for most scientific applications. However, recently it has been shown that for web and server traffic, the workload will resemble a high workload traffic where deflection or dropping of packets will result in higher power consumption [9]. Therefore, the proposed design attempts in hardware to
Fig. 11. (a-b) Throughput and (c) latency for DOR and WF routing algorithms with varying percentage of router crossbar faults.

balance the energy consumption and improve performance simultaneously for all workloads without relying on adaptive flow control mechanisms. We believe that the adaptive flow control techniques are complementary to our techniques which can further reduce the energy consumption by merging deflection and buffered networks.

E. Performance Degradation with Crossbar Faults

We evaluate the performance degradation due to faults in the crossbar in the NoC microarchitecture. First, we discuss the origin and evaluation of faults. The faults are randomly generated at different crossbars with the same random seed but varying percentages of faults. Once the fault is developed, we predict that the fault will manifest and will be detected after several cycles. We assume that BIST circuit can detect the fault in five router clock cycles. In our proposed network, priority is given to the incoming flit more than the buffered flit. Therefore, if the incoming flit does not get the desired output port, we can test the desired output port to check whether it is busy or free. If it is busy, the router functions correctly. If it is free, and we are unable to connect the input to the output, we assume that a fault must have occurred. Similarly, when packet at the buffered input port cannot connect to the output on switch allocation, then we detect
a fault must have generated. The number of cycles for fault detection is optimistically assumed to be five, however with parallelized circuitry, we can reduce this even further. Figure 11 shows the results for both DOR and WF adaptive routing algorithms for uniform random traffic. From 11(a), we can see that the throughput of the network does not change much with increasing faults for uniform random with DOR routing algorithm. The degradation in performance is less than 10% for varying percentage of faults. This is primarily due to the fact that as faults increase, most of these networks become a buffered network with a single functioning crossbar, thereby tolerate the fault. However from 11(b), we can see that the throughput degradation is significant with WF adaptive routing. With 100% faults i.e. there is a fault in almost every router, the degradation in performance is almost 33%. The loss of performance can be attributed to the fact that the delay in detecting the fault (five router clock cycles) affects adaptive routing more as most paths are non-minimal and therefore, when a fault occurs, more packets try to adapt to the topology. As more packets try to adapt to changed topology, packet throughput takes a hit. Figures 11(b) and 11(c) show the average packet latency which reflects some of the comments from before. Figures 11(d) and (e) show the power dissipated due to increase in faults. The common trend is the increase in power consumption as more packets are buffered and the network accounts for the buffered power as well. Overall, the dual crossbar network as well as the single crossbar, dual inputs can overcome faults with minimum performance degradation.

IV. CONCLUSIONS

In order to decrease the power consumption and enhance power consumption of the NoCs under all network loads, we propose DXbar design which combines primary bufferless crossbar with a secondary crossbar with the buffers. The design is implemented to take advantage of low-latency switching of bufferless networks at low load and buffering capability of buffered networks at high network load. Moreover, we also propose and show a unified crossbar design that achieves identical functionality with reduced area. The simulation results show that by using the the dual crossbars to handle excessive packets when network load is high, DXbar not only is able to outperform the baseline design with the same buffer size per input port, but it is also better than the baseline when the buffer size is doubled. More importantly, DXbar achieves 15% energy savings and 15-20% performance gains over the baseline design. In general, DXbar design achieves the goal of improving performance and saving energy, with the tradeoff of area. Moreover, our fault tolerance mechanisms indicate that we can tolerate excessive router failures due to dual crossbar or dual input single crossbar architecture. Our results also indicate that DOR routing provides higher throughput with faults when compared to WF-based adaptive routing algorithm.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their feedback. This research was partially supported by NSF awards, CCF-0915418, CCF-1054339 (CAREER) and ECCS-1129010.

REFERENCES