ReliNoC: A Reliable Network for Priority-Based On-Chip Communication

Mohammad Reza Kakoee
DEIS
University of Bologna
Bologna, Italy
m.kakoee@unibo.it

Valeria Bertacco
CSE
University of Michigan
Ann Arbor, USA
valeria@umich.edu

Luca Benini
DEIS
University of Bologna
Bologna, Italy
luca.benini@unibo.it

Abstract—**The reliability of networks-on-chip (NoC) is threatened by low yield and device wearout in aggressively scaled technology nodes. We propose ReliNoC, a network-on-chip architecture which can withstand failures, while maintaining not only basic connectivity, but also quality-of-service support based on packet priorities. Our network leverages a dual physical channel switch architecture which removes the control overhead of virtual channels (VCs) and utilizes the inherent redundancy within the 2-channel switch to provide spares for faulty elements. Experimental results show that ReliNoC provides 1.5 to 3 times better network physical connectivity in presence of several faults, and reduces the latency of both high and low priority traffic by 30 to 50%, compared to a traditional VC architecture. Moreover, it can tolerate up to 50 faults within an 8x8 mesh at only 10 and 40% latency overhead on control and data packets for PARSEC traces [24]. Synthesis results show that our reliable architecture incurs only 13% area overhead on the baseline 2-channel switch.**

I. Introduction

It has been predicted that future designs will consist of hundreds of billions of transistors, with upwards of 10% of them being defective due to wearout and process variation. Consequently, we must soon learn how to design reliable systems from unreliable components, managing both design complexity and process uncertainty [9].

The interconnect architecture in a chip multiprocessor becomes a single point of failure as it connects all other components of the system together. A faulty processing element may be shut down entirely, but the interconnect architecture must be able to tolerate partial failure and operate with performance or latency overhead [4]. Networks-on-chip provide opportunities to address this issue, as redundant paths exist from point to point, potentially allowing for reconfiguration around failed components. A Network on Chip (NoC) is a high performance and scalable communication mechanism, for transferring data among the cores in a multi processor SoC [13]. The reliability of NoC designs is threatened by transistor wearout in aggressively scaled technology nodes. Wear-out mechanisms such as oxide breakdown and electromigration become more prominent in these nodes as oxides and wires are thinned to the physical limits. These breakdown mechanisms occur over time, so traditional post burn-in testing will not capture them. NoCs provide inherent structural redundancy and interesting opportunities for fault diagnosis and reconfiguration.

978-3-9810801-7-9/ DATE11/ ©2011 EDAA

On the other hand, NoCs significantly affect performance, latency, power and area of the chip. NoC latency reduction is essential for SoC performance as it is introduced to every communication stream within the SoC. Latency may become vital in the case of SoCs with critical timing demands (real-time SoCs). Priority-based NoC architectures are a well-known approach to provide quality of service (QoS) in networks and to reduce the latency of certain packet classes whose priorities is high [12], [14]. Virtual Channels (VCs) over a physical channel have statically assigned priorities [12], [14], [15]: high priority VCs are used for QoS traffic classes and the low priority VCs are used for normal traffic classes.

Even considering that a VC switch can implement the same amount of buffering resources of its VC-less counterpart by simply re-structuring them into multiple smaller VCs instead of a single queue, the incremental complexity when augmenting a baseline switching fabric with virtual channels is still severe [14], [15]. This is due to additional logic and components for the virtual channel allocator. To address this issue, the use of multiple physicals network on the same chip has been proposed to improve performance and keep traffic classes separate [17] instead of leveraging VCs. Gilabert *et al.* in [1] showed that by replicating switches as many times as the number of VCs, a simple yet efficient implementation can be achieved providing the same (or even better) cycle time at lower area and power.

Motivation and Contribution. The motivation behind this work is having a priority-based NoC architecture that can provide reliability in presence of many faults with minimum overhead on area, and latency. We provide two main contributions in this work: first, we propose a new router architecture which distributes the traffic to two different physical channels based on the class of the traffic, which can be either QoS or normal. We completely remove the overhead of VCs by replicating the components inside the switch as proposed in [1]. In addition to [1], we also replicate the links between routers and remove the hardware overhead at input ports. Note that, other complementary QoS techniques such as circuit switching can be applied on this 2-channel router for guaranteed throughput traffics.

Second, we propose ReliNoC switch which is based on the mentioned 2-channel switch architecture. In the ReliNoC switch, we take advantage of the redundant components in switches and use them as replacements in presence of faults. Our architecture can tolerate several faults in the network.

Experiment results show that our 2-channel switch architecture can decrease the latency of both QoS and normal traffic by 30 to 50 percent compared to the VC-based switch in an 8x8 NoC with synthetic traffic. In addition, ReliNoC can tolerate up-to 50 faults inside the 8x8 mesh with 10 and 40% overhead on control and data packets latencies, respectively, in real traffic. Moreover, it provides almost 1.5 to 3 times better network physical connectivity compared to a VC-based architecture. Synthesis results show that ReliNoC switch has only 13% area overhead with respect to the baseline 2-channel switch.

II. Previous work

A number of works has evolved the VC switch microarchitecture to optimize latency and to develop an infrastructure for QoS purposes in the Chip Multi-Processor (CMP) domain [12], [14], [15]. Virtual channel flow control was first proposed by Dally in [12] as an effective workaround for head-of-line blocking. Since its implementation requires extra buffers, the router power consumption will increase with the number of VCs [14]. Moreover, VCs impose performance overhead due to virtual channel allocators [16], [1].

S. Noh *et al.* in [16] proposed a multiplane virtual channel router which has multiple crossbar switches and a modified switch allocator. This is proposed as a way to increase the flit transfer rate between input and output queues. Unfortunately, their interesting finding ultimately implies increased switch complexity and a critical path degradation. Gilabert *et al.* in [1] proposed a simple yet efficient approach to VC implementation. Instead of replicating only buffers for VCs, they replicated the entire switch and proved that their solution is counter-intuitively more area/power efficient while potentially operating at higher speeds. Our 2-channel switch is inspired from their work; however, we also replicate the links and provide two physical channels instead of virtual channels. Moreover, we propose a reliable architecture integrated with our 2-channel switch to recover different hardware failures in the network.

Reliable network design was first introduced by Dally *et al.* in [2]. This network used link-level monitoring and retransmission to accommodate the loss of a single link or router anywhere in the network, without interruption of service. Constantinides *et al.* presented the BulletProof router, which efficiently used a combination of spares and component-level N Module Redundancy (NMR) techniques for router level reliability [3]. However, NMR approaches are expensive, as they require at least N times the silicon area to implement.

Fick *et al.* proposed Vicis, a NoC design that can tolerate the loss of many network components due to wear-out induced hard faults [4]. Each router has a built-in-self-test (BIST) that diagnoses the locations of hard faults and runs a number of algorithms like ECC, port swapping, and a crossbar bypass bus to mitigate them. However, they use routing tables which needs to be reconfigured after fault detection. This reconfiguration logic imposes area and power overhead. Moreover, they did not propose any technique to recover faults in routing tables, port swapper, and arbiters.

Routing algorithms for fault-tolerant networks have been extensively explored for network level reconfiguration [5], [6], [7], [8]. These algorithms direct traffic around failed network components in a way that avoids network deadlock. Although we must also accomplish this, we additionally reconfigures around faults using other methods as well. We adopt an implementation of the routing algorithm described in [7] for part of our network level fault recovery.

III. 2-channel switch

In this section we describe the architecture of our 2-channel switch. The switch has five directions: West, East, South, North, and Local. Each direction has two input and two output ports. Each pair of input/output port makes a channel, so each direction has two channels. Channel one is dedicated to QoS traffic, while channel two is shared between QoS and normal traffic. We call these channels channel-1 and channel-2, respectively. The switch is derived from a normal switch with 5 ports and 2 virtual channels. In the original switch each VC is dedicated to one class of traffic. As it is proposed by Gilabert *et al.* in [1], replicating the entire switch for each class of traffic leads to better area and power for a given performance compare to traditional VC-based switch where only buffers are replicated. This is because replicating the entire switch removes the need of virtual channel allocators and makes the entire switch much simpler. Therefore, the critical path of this switch is much shorter than the traditional VC-based switch. This helps the synthesis tool to optimize the area and power of the design for a given target frequency.

We replicate the internal components of each port as well as the links and make 2-channel ports. Replicating links removes the need of VC decoders at input ports. At the output ports we need only one level of multiplexing because of link replication. For each buffer at the input port we have a bit which shows the type of traffic inside the buffer, and the arbiters decide based on that bit and give the priority to the buffers which contain QoS traffic.

We note that, VC-based switches are better when links are off-chip, as they trade-off increased logic complexity for better utilization of physical channels. When physical links are expensive, e.g. for off-chip networks, this trade-off makes sense. Also, VC-based switches are better when the number of VC becomes very large and they are used rarely, but this is not our case, as we have only two classes of traffic and high utilization for both.

IV. ReliNoC Architecture

Utilizing the properties of the proposed 2-channel router, ReliNoC switch is introduced to ensure fault-tolerant operation of the NoC. In this section, we explore various possible failure modes within an NoC router, and propose detailed recovery schemes with minimum area cost. ReliNoC router architecture possesses some inherent fault-tolerance due to its replicated design. This additional operational granularity may be utilized to allow replacement of a faulty component by another one, thus allowing operation of the router with latency overhead instead of a complete breakdown. Five major components of the router i.e. routing computation units (RC), arbiters, buffers, output MUXes, and links, are susceptible to permanent faults. We propose ReliNoC switch to overcome faults in all these components. The proposed switch architecture is shown in Figure 1. Note that, only input channels of West port and output channels of East port are shown in the figure. Four types of components are added to the baseline 2-channel switch for fault tolerance purposes: 1) two MUXes per each input channel 2) 5-bit input status register per each channel 3) one control logic for both channels 4) 10-bit output status register for the entire switch.

As shown in Figure 1, two MUXes are added to each channel, one at the entry of buffer (MUXBUFF) and another one at entry of RC (MUXRC). The muxes are added to enable each channel to access both buffers and routing computation units. There is a 5-bit register for each input channel storing the faultiness status of five different components in that channel including: link, MUXBUFF, MUXRC, RC, and buffer. We call this register ISR (Input Status Register). One simple control logic reads the ISR and generates appropriate control signals to MUXBUFF and MUXRC based on the faultiness of each component. Finally, there is a 10-bit register for the entire switch tracing the faultiness of all output channels in that switch. We consider one output channel as a faulty channel if there is a fault in at least one of the following components of that channel: arbiter, out-multiplexer, link, and the input channel of the corresponding next router. We call this register OSR (Output Status Register).

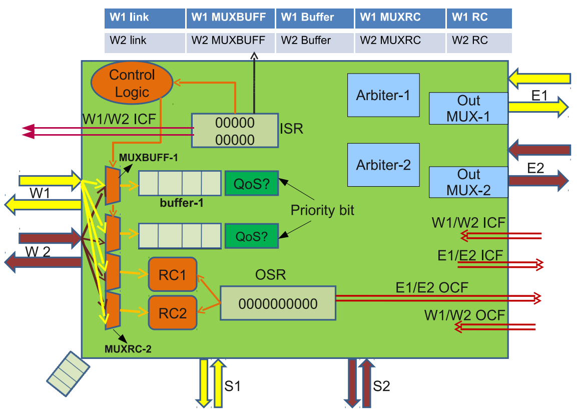


Fig. 1. ReliNoC switch. Each direction has two channels which are used for QoS purposes in normal operation and for reliability purposes in case of faults.

Every two connected routers in ReliNoC send the information of their corresponding ISR and OSR to each other by dedicated signals. Each router receives two signals from its neighbour per each channel. One shows the status of the neighbour's input channel and another one is for the output channel. We call these signals input-channel-faulty (ICF) and output-channel-faulty (OCF) respectively. ICF and OCF signals can affect on OSR and ISR registers. If router A receives an active OCF signal from router B for a specific channel, this means that router B never sends data to router A on that channel. This is equal to having faults on the input link of that channel in router A and, therefore, router A sets the "link" faultiness bit in the corresponding ISR to '1'. If router A receives an active ICF signal from router B on a specific channel, it means that router A should not send data on the corresponding output channel. This is equal to having faults on the output link of the related channel in router A and, therefore, router A sets the corresponding bit in its OSR to '1'.

Based on the values in ISR, control logics generates appropriate signals for MUXBUFF and MUXRC of each channel. The information in OSR is used by RC logic to specify the appropriate output channel for each packet. Depending on the status of ISR and OSR, several different cases exist for each channel and each direction of the router. In the following paragraphs we outline these cases for the East direction and its two channels (E1 and E2) and describe our fault recovery schemes for each situation. The mechanisms for other directions are exactly the same as those of East.

All bits in ISRs of E1 and E2 are '0': this means all the input components at East channels (E1 and E2) of the current router and all the output components at West channels (W1 and W2) of its related neighbour are non-faulty. In this case the router works in its normal operation.

The bit of buffer-1 in ISR is '1': this means buffer-1 is faulty and can not be used any more. In this case we consider channel-1 as faulty. As an ICF signal is already sent to the previous router, no data will arrive at channel-1 and channel-2 is utilized for both types of traffics. Thus, control logic does not change control signals for MUXes. In this case RC-1 also remains idle. The same scheme is applied when buffer-2 is faulty. Note that, even in these cases the QoS traffic has priority over normal traffic because of the priority bit.

The bit of MUXBUF-1 in ISR is '1': this means MUXBUF-1 is faulty. This case is similar to the previous one. No data will arrive at channel-1 and channel-2 is utilized for both traffics. The same mechanism is applied on channel-2 when MUXBUF-2 is faulty.

The bit of link-1 in ISR is '1': this means Link-1 or the output channel of previous router is faulty. Therefore, no data will arrive at channel-1 because of ICF signal which is already sent to the previous router. However, in this case if buffer-1 and MUXBUF-1 are non-faulty, they can be used by channel-2. Here, control logic generates appropriate signals so that channel-2 is connected to both buffer-1 and buffer-2. In this case buffer-1 is used for QoS traffic and buffer-2 for both QoS and normal traffic.

The bit of RC-1 in ISR is '1': this means RC-1 is faulty. Clearly, since RC and MUXRC are driven only by the header flit, their utilizations are relatively low compared to the flit-by-flit operation of per-flit components like buffers. Thus, RC of one channel can be shared during its unloaded periods with another channel. However, this needs a more complex control logic since two packets may arrive at the same time. To avoid this, we do not share RCs between channels and, therefore, consider channel-1 as faulty. The same mechanism is applied on channel-2 when RC-2 is faulty.

The bit of MUXRC-1 in ISR is '1': this means MUXRC-1 is faulty. This case is similar to the previous one and we consider channel-1 as faulty. The same mechanism is applied on channel-2 when MUXRC-2 is faulty.

The East-channel-1 bit in OSR is '1' and East-channel-2 bit is '0': this means one of the components (arbiter, link, multiplexer) at the output channel E1 is faulty or the input channel of the next router which is connected to E1 is faulty (ICF is '1'). As all the RCs take the OSR bits into account, in this case no traffic will go through E1, and E2 is used by both QoS and normal traffic.

Both East-channel-1 and East-channel-2 bits in OSR are '1': this means both output channels at East direction are faulty. In this case, the packet cannot go to the East direction and should be detoured to another direction. To do so, we use Logic-Based Distributed Routing (LBDR) which is a fault tolerant routing algorithm proposed by Rodrigo *et al.* in [7]. This routing is a logic-based mechanism which can be implemented on top of any distributed routing like XY to detour the faulty link. This logic eliminates the need of routing tables (either at routers or at end-nodes). Routing tables do not scale in terms of latency, power consumption, and area, thus being impractical for large NoCs [10]. As described in [7], the area overhead of the added fault tolerant logic on top of the traditional XY routing is around 4% which is much less than that of table based routings. This additional logic increases the timing path of RC by 20%. However, as the timing critical path of the router is not through the RC logic and this module (RC) is not the one setting the router frequency, this performance overhead on the RC does not have any effect on the router performance [7]. The added logic needs some status bits to find the detour port [7]. Eight of them which represent the faultiness of each output port are already in OSR. Three other bits should be added to each direction to implement the fault-tolerant logic on top of XY.

Since we do not recover faults in fault-status registers (ISR and OSR), the recovery techniques in our architecture rely on having robust design for these registers. Error correcting codes (ECCs) [18] are good robust mechanisms for registers. Various ECC mechanisms have been integrated within VLSI chips with large internal memories and cache units [19] as well as NoCs [4]. Also, TMR can be used for the reliability of ICF and OCF signals.

Our reliability mechanism is based on knowing precisely the location of faults in the switch. Built-in-self-test (BIST), a well-known approach to diagnose faults, has been extensively addressed as a post-silicon technique for fault detection during the NoC lifetime [4], [20], [11]. To utilize our recovery mechanisms, the network requires to go periodically into self-test in regular intervals and update ISRs and OSRs.

V. Experimental Results

To evaluate our ReliNoC architecture, we used Noxim [23] which is a cycle accurate NoC simulator implemented in SystemC. The switch model in this simulator has a 2-stage pipeline and therefore has 2 cycles minimum latency. We extended it to support both virtual and physical channels. We also extended it for our reliability techniques.

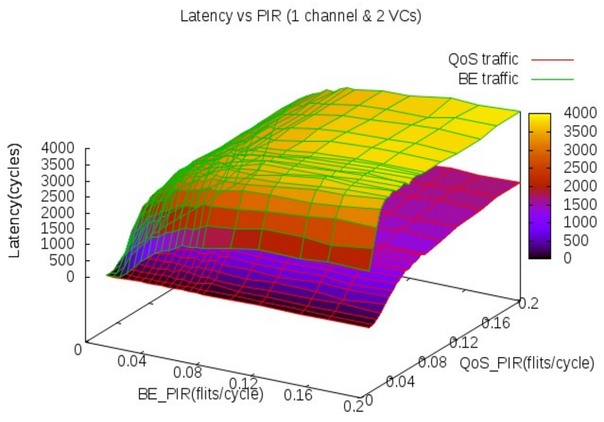
A. NoC latency evaluation

To see the effectiveness of ReliNoC router with respect to the VC-based router, we measured the average latency on 8x8 Mesh networks of both routers. We applied different synthetic traffic patterns including Unified Random, Butterfly, Transpose, and Bit complement on both networks. We injected two classes of traffic to the network including QoS and normal, and measured the latency of both traffic types as a function of packet injection rate of both traffic classes. The results for unified random traffic are shown in Figure 2. In this 3-dimensional plot, the X and Y axes represent the packet injection rate (PIR) of normal and QoS traffic respectively. The Z axis shows the latency. The surfaces in Figure 2 are the latency of QoS and normal traffic. As seen, in both routers the PIR of normal traffic does not have effect on latency of QoS traffic. This is because of the arbitration policy in the router which gives high priority to the QoS traffic. However as can be seen, the latency of normal traffic in ReliNoC is much less than that of NoC with 2-VC switch. As shown in the plots, increasing PIR of QoS traffic has much less impact on latency of both traffic types in ReliNoC compared to that of NoC with 2-VC switch. As seen, the latency of both QoS and normal traffic has been reduced by 30 to 50 percent.

B. Network connectivity

To see the effect of faults on the network, we injected faults on the entire 8x8 NoC. Our fault model is based on the area and covers seven different components (RC, MUXBUFF, MUXRC, buffer, arbiter, out-MUX, link) of each channel in each router. We injected faults on the network based on the area portion of each component in the network. Those components that have more portions of the total NoC's area have more probability to receive faults.

First, we compared the network connectivity of ReliNoC with that of a network comprising 2-VC switches in presence of different number of faults. In this experiment, we injected different number of faults on the network on different random components by giving higher probability to larger components. Then, we checked statically if the network is fully connected or not. We consider a network as fully-connected if there is at least one physical path from each source to each destination regardless of the routing algorithm. For any number of faults, we injected that amount of faults on the network with 1000 different random seeds, and calculated the average number of fully-connected networks out of 1000 possible fault distributions. We did this experiment for two types of networks: i) 8x8 ReliNoC and ii) 8x8 NoC with 2-VC switch. Figure 3 shows the plot of network connectivity of both cases in terms of number of faults. The Y axis in this chart is the probability of having a fully-connected network as a function of "number of faults". As seen, ReliNoC has much better connectivity in presence of faults. For example, in presence of 20 faults, ReliNoC has 90% probability of fully-connected network,



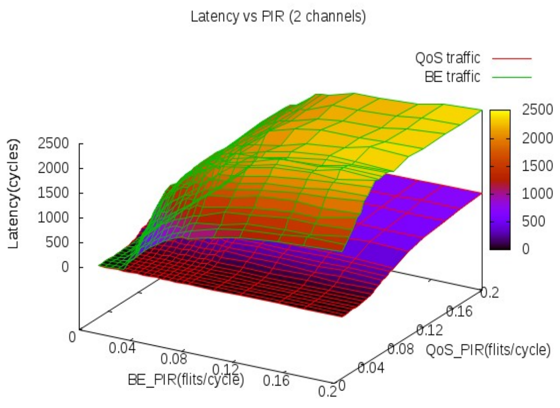


Fig. 2. Latency vs. packet injection rate (PIR).

while that of NoC with VC-based switches is 70%. For 40 faults, ReliNoC shows 2 times better physical connectivity compare to the VC-based architecture.

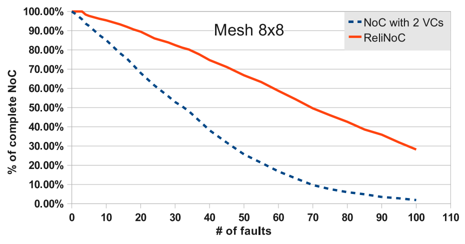


Fig. 3. Network Connectivity vs. number of faults.

C. NoC recovery evaluation

To see the effect of faults on the latency and to verify our recovery mechanism in ReliNoC, from previous experiments we selected those fault configurations that have fully-connected networks. Then we injected faults and ran the simulation for different synthetic traffics. We measured both average and worst-case latency among all configurations. Figure 4 shows the relation between average latency of both QoS and normal packets and number of faults for 2 synthetic traffics: random and butterfly. As can be seen, when the number of faults increases, the latency of both QoS and normal packets increases. However, in both benchmarks the QoS traffic pays less latency penalty compare to that of the normal traffic. Experiments showed that our recovery mechanism in ReliNoC could handle all the fully-connected networks for up-to 50 faults, and all packets reached the destination with a reasonable latency overhead.

We also calculated the worst-case latency overhead in all fault configurations having fully-connected networks. Experimentally, we found the worst-case latency overhead in Bit-Complement traffic. Our Experiments showed that even in the worst-case, the latency overhead on QoS packets was 30% for up-to 25 faults. However, in the worst-case configuration, the normal packets payed much more latency overhead in presence of many faults compared to the QoS packets. The latency

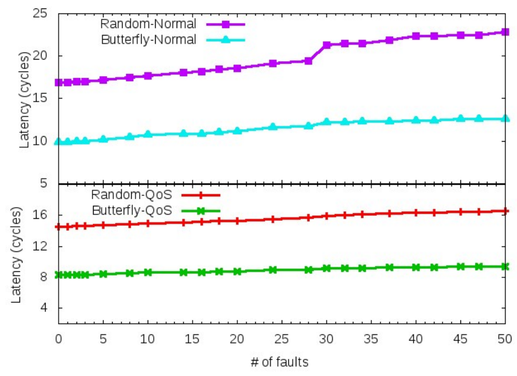
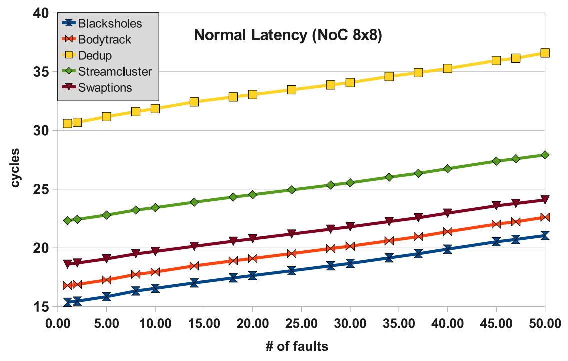


Fig. 4. Average latency vs. number of faults for synthetic traffics in 8x8 ReliNoC.

overhead in normal packets of Bit-Complement traffic for 15 faults in the worst-case configuration was 5X.

In addition to the synthetic traces, we also performed simulation to see the effect of faults on 5 real traffic traces from PARSEC benchmarks, a suite of next-generation shared-memory programs for CMPs [24]. The traces used are for a 64-node shared memory CMP arranged as a 8x8 mesh. Each processor node has a private L1 cache of 32KB and 1MB L2 cache (64MB shared distributed L2 for the entire system). There are 4 memory controllers at the corners. To obtain the traces, we used Virtutech Simics [25] with the GEMS toolset [26], augmented with GARNET [27], simulating a 64-core NoC. Like the previous experiments we chose those fault configurations that give us fully-connected networks, injected faults and ran the simulation for each benchmark trace. Each trace contains two types of packets: data and control. We considered control and data as QoS and normal packets respectively. Figure 5 shows the relation between latency and number of faults for each benchmark and for each packet type. As can be seen, in real traffics ReliNoC can recover up-to 50 faults with a very low penalty on latency in both traffic types. In all the benchmarks and for 50 injected faults, the maximum latency penalty on control and data packets is 10% and 40%



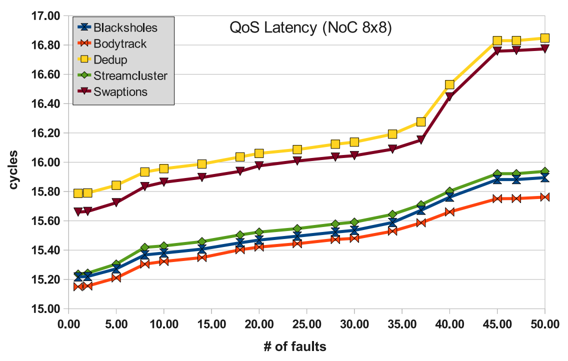


Fig. 5. Latency vs. number of faults for PARSEC benchmarks in 8x8 ReliNoC.

respectively.

D. *Cost of reliability hardware*

To see the area overhead of our reliable schemes on the original 2-channel switch, we synthesized the RTL version of the ReliNoC switch which is based on Xpipes router [21] in 65nm technology. We performed automatic BIST insertion using Synopsys Design Compiler to find the BIST overhead. We carried out this experiments for different size of input buffers with a flit width of 32 bits. The results are shown in Table I. As it can be seen, the area overhead is from 12.49% to 15.45% depending on the number of buffers. For a reasonable number of buffers, i.e. 8, the area overhead of all components including ISR, OSR, control logics, fault tolerant RC, and BIST chain is only 12.8%. To efficiently perform physical routing for the wires of 2-channel links, the bundled NoC global link routing technique proposed by Kakoee *et al.* in [22] can be utilized.

TABLE I
BIST AND OTHER LOGICS AREA OVERHEAD IN RELINOC SWITCH

# of buffers	*Area (no reliability)*	*Area (with reliability)*	*Overhead*
2	7567	8736	15.5%
4	13873	15961	15%
6	20421	23066	13%
8	27004	30458	12.8%
16	52426	58972	12.5%

VI. CONCLUSIONS

A priority-based switch architecture which can provide reliability in presence of several faults inside the NoC has been proposed. First, a 2-channel switch architecture which completely removes the control logic overhead of VCs was introduced. Then, we proposed ReliNoC architecture which utilizes the inherently redundant components inside the 2-channel switch as replacements for the faulty elements.

VII. ACKNOWLEDGMENTS

This work is supported by HiPEAC and JTI-ENIAC-Modern project.

REFERENCES

[1] F. Gilabert et al., “Improved Utilization of NoC Channel Bandwidth by Switch Replication for Cost-Effective Multi-Processor Systems-on-Chip,” Proc. ACM/IEEE *NoCS*, pp. 165-172, 2010.
[2] W. J. Dally et al. “The reliable router: A reliable and high-performance communication substrate for parallel computers,” Proc. *PCRCW*, pp. 241-255, 1994.
[3] K. Constantinides et al. “BulletProof: a defect-tolerant CMP switch architecture,” Proc. IEEE *HPCA*, pp. 5-16, 2006.
[4] David Fick et al., “Vicis: A Reliable Network for Unreliable Silicon,” Proc. ACM/IEEE *DAC*, pp. 812-817, 2009.
[5] D. Fick et al.,“A highly resilient routing algorithm for fault-tolerant NoCs,” Proc. ACM/IEEE *DATE*, pp. 21-26, 2009.
[6] M. E. Gomez et al., “An efficient fault-tolerant routing methodology for meshes and tori,” IEEE Computer Architecture Letters, vol. 3, No. 1, pp. 3-3, 2004.
[7] S. Rodrigo et al., “Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing,”, Proc. ACM/IEEE *NoCS*, pp. 25-32, 2010.
[8] C.-T. Ho et al., “A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers,” IEEE Trans. on Computers, Vol. 53, No. 4, pp. 427-439, 2004.
[9] S. Borkar,“Microarchitecture and design challenges for gigascale integration,” Proc. ACM/IEEE *MICRO*, keynote address, pp. 3-3, 2004.
[10] J. Flich et al., “An efficient Implementation of Distributed Routing Algorithms for NoCs,” Proc. ACM/IEEE *NoCS*, pp. 87-96 , 2008.
[11] A. Alaghi et al., “Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model,” Proc. IEEE *DFT*, pp. 21-29, 2007.
[12] W.J.Dally “Virtual-Channel Flow Control,” Proc. ACM/IEEE *ISCA*, pp. 60-68, 1990.
[13] T. Bjerregaard et al., “A survey of research and practices of network-on-chip,” ACM Computer Survey, Vol. 38, No. 1, 2006.
[14] N.Banerjee et al., “A Power and Performance Model for Network-on-Chip Architectures,” Proc. ACM/IEEE *DATE*, pp. 21250, 2004.
[15] A.Mello et al., “Virtual Channels in Networks-on-Chip: Implementation and Evaluation on Hermes NoC,” Proc. ACM/IEEE *SBCCI*, pp. 178-183, 2005.
[16] S.Noh et al. “Multiplane Virtual Channel Router for Network-on-Chip Design,” Proc. IEEE *ICCE*, pp. 348-351, 2006.
[17] Young Jin Yoon et al.“Virtual channels vs. multiple physical networks: a comparative analysis,” Proc. ACM/IEEE *DAC*, PP. 162-165, 2010.
[18] W. Peterson et al., Error-Correcting Codes, 2nd ed. Cambridge, MA: MIT Press, 1972.
[19] A. Agarwal et al., “Process variation in embedded memories: Failure analysis and variation aware architecture,” IEEE J. Solid-State Circuits, Vol. 40, No. 9, pp. 1804-1814, 2005.
[20] M. Hosseinabadi et al., “A Concurrent Testing Method for NoC Switches,” Proc. ACM/IEEE *DATE*, pp. 1171-1176, 2006.
[21] D. Bertozzi et al., “Xpipes:A network-on-chip architecture for gigascale system-on-chip,” IEEE Circuits Syst. Mag., Vol. 4, No. 2, pp. 18-31, 2004.
[22] M. R. Kakoee et al., “A new physical routing approach for robust bundled signaling on NoC links,” Proc. ACM/IEEE GLSVLSI, pp. 3-8, 2010.
[23] F. Fazzino et al, Noxim: Network-on-chip simulator [Online]. http://noxim.sourceforge.net
[24] Christian Bienia et al., “The PARSEC benchmark suite: characterization and architectural implications,” Proc. ACM *PACT*, pp. 72-81, 2008.
[25] P. S. Magnusson et al., “Simics: A full system simulation platform,” IEEE Computer, Vol. 35, No. 2, pp. 50-58, 2002.
[26] M. M. K. Martin et al., “Multifacet’s general execution-driven multiprocessor simulator (GEMS) toolset,” SIGARCH Computer Architecture News, Vol. 33, No. 4, pp. 92-99, 2005.
[27] L.-S. Peh et al., “GARNET: A detailed on-chip network model inside a full-system simulator,” Proc. IEEE *ISPASS*, pp. 33-42, 2009.