Comprehensive Online Defect Diagnosis in On-Chip Networks

Amirali Ghofrani*, Ritesh Parikh†, Saeed Shamshiri*, Andrew DeOrio†, Kwang-Ting Cheng*, and Valeria Bertacco†
*Department of Electrical and Computer Engineering, University of California, Santa Barbara, California 93106
Telephone: (805) 893–5678, Fax: (805) 893–3262
Email: {ghofrani, saeedshamshiri, timcheng}@ece.ucsb.edu
†Department of Electrical Engineering and Computer Science, University of Michigan at Ann Arbor
Email: {ritesh, awdeorio, valeria}@umich.edu

***Abstract*—We propose a comprehensive yet low-cost solution for online detection and diagnosis of permanent faults in on-chip networks. Using error syndrome collection and packet/flit-counting techniques, high-resolution defect diagnosis is feasible in both datapath and control logic of the on-chip network without injecting any test traffic or incurring significant performance overhead.**

***Keywords*-Network-on-Chip; permanent faults; online diagnosis; error correcting codes; fault-tolerant router;**

I. Introduction

To satisfy the perpetual need of increasing processor performance while maintaining manageable levels of power consumption, multi-core systems have become the prevalent architectures for high performance computing [2]. The number of processors on a single chip is increasing from few cores in Intel's quad-core processors [16] to thousands of simple processors in Adapteva's Epiphany [1]. Among various communication infrastructures, Network-on-chip (NoC) has become the dominant architecture to cope with the ever-increasing complexity of these multi-core systems [9].

Aggressive technology scaling has increased the number of transient and permanent faults due to increased process variation and reduced noise-margins [6], [13]. In a multi-core system, both the processor cores and the interconnecting network should be protected against such faults. Fault tolerance techniques for processors are well-studied in the literature [24], [17], [22], [19], [15] and are not the subject of this paper. In this paper, we address the reliability issues for the interconnect network by detecting the presence of permanent faults and then locating the defective component. This diagnostic information can then be used to reconfigure the system in a way that better utilizes the fault tolerance resources.

Defective routers and links in NoCs result in various erroneous behaviors that could cause performance degradation, data loss, and ultimately complete system failure [4]. These reliability concerns have motivated various innovative fault-tolerance methods that are discussed in the next section. These methods enhance reliability by detecting and correcting errors, and recover the system by either bypassing or replacing the defective parts [11], [12], [21], [10], [3]. However, they also impose area and performance overhead and/or fail to adequately address both data and control errors.

In this paper, we present a comprehensive online method to diagnose permanent faults in an NoC. The proposed method addresses both datapath and control-logic faults and incurs a low area overhead (less than 10% is our case study). The transient faults in the datapath are corrected using existing Forward Error Control (FEC) methods [20], and our passive diagnosis method runs in the background to locate the permanent faults, based on the observed FEC syndromes. For control faults, our counter-based diagnosis mechanism locates a permanent fault shortly after its occurrence. Complementary corrective methods such as Automatic Repeat reQuest (ARQ) can be employed to recover from data loss due to such faults, which imposes minimal corrective traffic overhead due to our method's short diagnosis time. For a relatively healthy network, the performance overhead of our method is negligible. In summary, the main contributions of this paper are:

- A comprehensive detection and diagnosis method that covers permanent faults in both datapath and control logic
- Passive and high-resolution diagnosis with low area and runtime overheads

The rest of this paper is organized as follows: Sec. II discusses the sources and effects of NoC faults and mentions some of the researches performed to address them. In Sec. III, our proposed passive error-detection and online diagnosis solution is introduced and statistical analysis are shown to demonstrate the accuracy of the proposed method. Simulation setup and experimental results are discussed in Sec. IV, and Sec. V concludes the paper.

II. Background and Related Work

There are many different causes of errors in NoCs: crosstalk, radiation-induced soft errors, voltage-induced delay errors, *etc.* Often these effects result in transient faulty behavior in the system [21], [14], while permanent faults are resulting from manufacturing defects, infant mortality, or hardware aging. Manufacturing testing and burn-in can help screen most manufacturing and infant mortality defects, but wear-out defects cannot be detected before shipment. These permanent faults need continuous in-field testing, and are the target of this paper.

These faults can result in erroneous NoC behaviors in the form of control errors or corrupted data [4]. Any permanent or transient fault in any of the components in the routing path of transmitted data (datapath), such as router external or internal wires, input FIFOs, or the cross-bars (as shown in Fig. 1), could manifest as a corrupted packet payload since it changes the transmitted data. It should be noted that an error in the header of the packet could also manifest as a control error, sending the packet to a wrong destination, or resulting in an incomplete or dropped packet.

On the other hand, if the fault site is in the control hardware of a router, it can result in dropped, spurious/duplicated, or misrouted packets rather than changing the packet payload [4]. For example, if any stuck-at fault happens in the routing table, the packet may be sent to an output port other than the one intended, resulting in a misrouted packet. Alternatively, a false "empty" signal from a router's input FIFO could lead to retransmitting the last packet in the FIFO, causing a spurious duplicated packet. A false "not-full" signal could result in dropped packets, since the previous router would keep sending packets to one with a full FIFO. For comprehensive diagnosis in NoCs, both data and control faults should be addressed.

Forward Error Control (FEC) methods use Error Correction Codes (ECC) in order to detect and correct data faults during transmission by adding data redundancy to the packets [11]. However, these methods lose their efficiency in the case of permanent faults since the correction strength of the ECC is frequently consumed by the permanent faults in the datapath.

To address such permanent faults, in [21] the authors proposed an online diagnosis mechanism that collects error syndromes in the system and locates the faulty link while end-to-end ECC is used to correct data errors in the network. However, the goal of [21] is to pinpoint defective components to the resolution of links between routers; defects in the router's datapath are not directly addressed. A higher-resolution diagnosis capable of locating the faulty part(s) within a router could enable replacing/bypassing the defective part(s), rather than disabling the entire router or link. Moreover, control errors such as spurious/dropped packets and misroutings are out of the scope of that paper.

978-1-4673-1074-1/12/$31.00

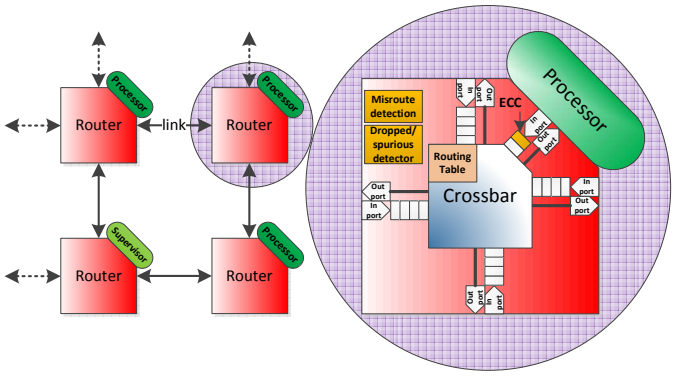

Fig. 1: **NoC and internal structure of the router.** Each router is augmented with end-to-end error correction code, misroute detection hardware, and counter-based dropped/spurious packet/flit detection, and one of the nodes also acts as a supervisor.

Addressing both data and control faults, in [7] the authors demonstrated a reliable router using N-Modular Redundancy (NMR) methods. However, NMR imposes a very high area overhead and is not applicable to all circuit components (e.g. clock tree). Other methods such as Automatic Repeat reQuest (ARQ) can address both data and control errors as well. ARQ methods ensure that each packet reaches its destination without error by monitoring and retransmission [25]. However, these techniques cannot effectively handle permanent faults. Permanent faults result in frequent erroneous traffic in the system, flooding the network with ARQ corrective packets and leading to significant performance degradation. This necessitates a fast diagnosis method for permanent faults, after which we can either bypass the faulty parts or replace them with spares.

A diagnosis method was introduced in [12] where the authors proposed embedding Built-In-Self-Test (BIST) error detection hardware followed by a rerouting mechanism to bypass the defective parts. The results of their case study showed that each router would incur a 43% area overhead. Moreover, whenever an error is detected, this method will need to interrupt the normal system operation and switch all routers into the test mode to diagnose the faulty component.

Our proposed approach is a passive diagnosis method that eliminates the shortcomings of the aforementioned solutions. The solution, imposing lower area and performance overheads in comparison with existing methods, addresses both data and control errors, enhances the diagnosis resolution to match available fine-grained repair options, and works in the background without affecting the normal system operation. Fig. 1 shows an overview of the system in presence of the method.

III. Online Error Detection and Diagnosis

Our diagnosis approach is based on observing regular traffic in the system for error detection, and consists of two main components: *1)* a globally centralized scoreboarding approach is utilized to collect the error syndromes of corrupted data and pinpoint the fault location of detected data errors, and *2)* a local flit/packet counting method, implemented in each individual router, is used to diagnose packet drop/duplicate and misroute resulting from control-logic faults.

A. Data Fault

Inspired by [21], we introduce a novel diagnosis method based on scoreboarding to pinpoint the fault location in the datapath. Locality-aware ECC [20] is added to packets protecting them against faults along the path. However, the method is independent of the used ECC. The ECC is decoded at the destination as the packets are received, which detects and corrects errors. In our method, the error syndromes produced after decoding the ECC are also used to differentiate permanent and transient faults based on how often errors are observed at the same bit position. For example, if a low percentage of the flits in a packet have an error on a specific bit position, it can be inferred that the errors are due to a transient fault. However, a high percentage of erroneous flits at the same bit position would indicate a permanent error in that bit position somewhere along the packet's path.

This threshold for declaring a fault as permanent is one of the diagnosis parameters. A high threshold results in missing the syndrome of some permanent errors, while a low threshold results in misdiagnosis of transient faults as permanent faults. The exact value of this threshold will depend on a number of other design and operational parameters, and determining it is not the subject of this paper. However, the repercussions of a poorly selected threshold are discussed in Sec. IV-B.

When a data error is detected at the destination and is declared as permanent according to the threshold, an error information packet is sent to a designated supervisor node responsible for scoreboarding and diagnosis. The supervisor node is one of the nodes of the network and thus protected by the same error correction mechanisms. The error information packet contains the source and the destination of the erroneous packet and the bit position at which the error occurred. Based on this information, a probabilistic method is used to calculate the fault probability for each component along the possible paths from the source to the destination. These probabilities are accumulated on the scoreboard. After observing sufficient traffic in the system and accumulating the fault probabilities from enough erroneous packets, we expect that the scoreboard entry with the highest accumulated fault probability is indeed defective. This argument is later supported by statistical analysis in Sec. IV-B and the method is clarified by an example.

The method can differentiate faults in components A and B only if there is a packet that goes through component A and not thorough component B, or vice versa. Accordingly, under a regular router architecture, faults in an external link, an input port and the input FIFO that connects them would not be differentiable. Collectively, we call these three components a "link" in the rest of the paper. However, faults within a router's crossbar are more diagnosable since different packets traveling through a crossbar could utilize different crossbar inputs/outputs. Based on this observation, this paper offers a diagnosis solution for locating a datapath fault to a unique link or to a unique pair of I/O ports on a router (hereafter referred to as a "turn").

The fault probability calculation for different components is done at the software level once an error information packet is received and thus imposes no area overhead. These calculations depend on the interconnect architecture and routing algorithm used in the network. In this study we focus on a mesh topology, which is one of the most frequent schemes used in the existing literature.

Our method can support a wide range of routing algorithms and the component-level fault probability is computed accordingly. The number of possible paths between the source and destination might be one (as in a deterministic X-Y routing algorithm) or several (as in a nondeterministic minimal-routing in which the route could be any minimal length route) and the probability of taking each of the components along those paths can be pre-calculated and stored. In case of observing an error, the component-level fault probabilities are calculated based on usage probability of each component and the number of components on each path and then are accumulated in the scoreboard to diagnose the defective component.

In the following, we show the step by step calculation of component-level fault probability once the supervisor node receives an error information packet. For the sake of better illustration, a 4-by-4 mesh is assumed as shown in Fig. 2 with minimal-routing algorithm. A permanent fault is present in the $left \rightarrow up$ turn inside router D that may cause an error in a packet sent from source A to destination B.

Step 1- Calculating the usage probability of each link: The use of a minimal-routing algorithm implies that a valid path is limited to the links and turns inside the gray rectangle, each of which has a different usage probability. At A, there are two output links x and y that can be used to transfer the packet. Their usage probability is equal (0.5) since all the packets will travel through one of them. At the next router, C, we again have two valid output links with equal probability. However, since only half of the packets are sent to the

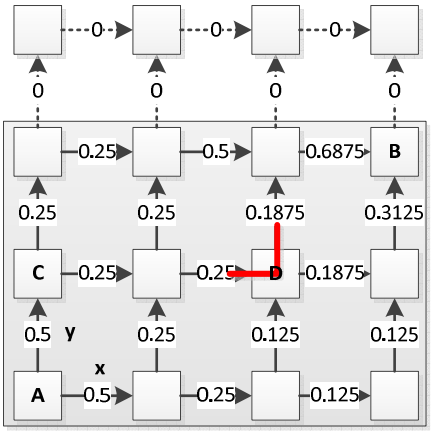

Fig. 2: **The usage probability of each link for a packet sent from source A to destination B. The red turn is defective.**

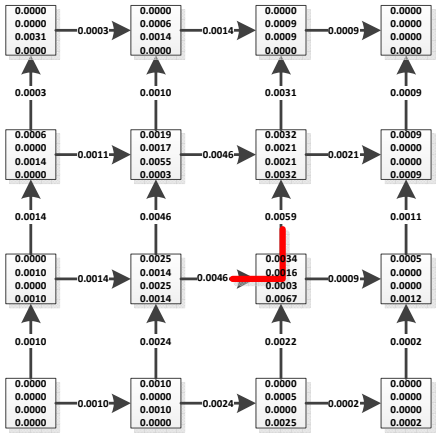

Fig. 3: **The expected value of different links and turns being faulty in a 4-by-4 minimal-routed mesh given that the red turn is defective. Values shown inside each router are the expected values for $bottom \rightarrow up$, $left \rightarrow right$, $bottom \rightarrow right$, and $left \rightarrow up$ turns respectively. values are truncated to four decimal digits.**

router C, the cumulative usage probability of each of the links is only 0.25. In this way, the usage probability can be calculated for all links in the gray region, according to Eq. 1:

$$Prob(output\ link\ at\ node\ X) = \frac{\sum Prob(input\ links\ to\ node\ X)}{number\ of\ valid\ output\ links} \quad (1)$$

The number of valid output links depends on the interconnect architecture and routing algorithm. The usage probability of different links assuming pair A, B as source and destination is shown in Fig. 2. Note that usage probability of the same component varies based on the selected source and destination pair.

Step 2- Assigning the usage probability of each turn: While the actual usage probability of each turn can be calculated by dividing the usage probability of the input link of the turn by the number of valid output links assuming that specific input link is taken, recall that these numbers are accumulated in a table in order to find the defective component. Whenever a turn is used, both links forming that turn are also used and their scores will increase at least as much as the turn. To avoid overshadowing by and misdiagnosis of the links, the usage probability recorded for turns should be assigned to the highest usage probability of its connected links multiplied by a small scaling constant. As we demonstrate in Sec. IV-B, this strategy will not affect our ability to detect faulty links if enough syndromes are collected.

Step 3- Calculating the fault probability of each part: An observed error could be caused by any of the links or the turns along the routing path and shorter paths, can let us know more about the exact fault site since there are less components on them. Hence, the fault probability for each component is defined to be the associated usage probability divided by the total number of links and turns along the path.

Step 4- Accumulating the fault probabilities in the scoreboard: The scoreboard is a table of floating point fault probability entries for each bit position in each link or turn of the network. Every time an error information packet is received by the supervisor, the fault probability for each link or turn that could be responsible is calculated and is accumulated in the scoreboard. Aging (i.e. gradual decrease of the scoreboard entries over time) can be incorporated to limit the repercussions of transient faults that were misdiagnosed as permanent faults.

At the end of this process, the entry with the highest fault probability is declared faulty if its score or the number of erroneous packets exceeds a preset threshold. To justify this conclusion, we consider the expected scoreboard value for each entry, assuming a specific fault and a random (uniformly distributed) packet source/destination.

Figure 3 shows the expected value of the scoreboard entries for a faulty turn (shown in red) on an 4-by-4 mesh. The four values shown inside each router are the probabilities for $bottom \rightarrow up$, $left \rightarrow right$, $bottom \rightarrow right$ and $left \rightarrow up$ turns, respectively. The other turns are omitted as their expected values are strictly zero. The values on each edge are the scores for the links, and again, the opposing link directions are omitted as their expected values are strictly zero. As we can see, the faulty turn indeed has a higher expected value of being faulty than the others.

Our experimental results in Section IV-B indicate that just a few tens of erroneous packets are sufficient for an accurate diagnosis of the defective part. It should be emphasized that ECC corrects the errors caused by the permanent faults, so the system operates normally during the diagnosis process.

B. Control Faults

A fault in a router's control logic may lead to many different failure scenarios. Such faults may lead to data corruption and/or to complete flits/packets being dropped or spurious ones generated. In addition, such faults could lead to packets being misrouted, potentially resulting in deadlock or livelock. Other errors that inhibit forward progress, such as starvation, can be due to faults in the arbitration logic rather than misrouting. Hence, we try to detect various classes of control faults and accurately diagnose the faulty port using the following techniques:

- Dropped or spurious flits: by counting the number of flits per packet received by a router's input port.
- Dropped or spurious packets: by keeping a total count of packets currently inside any router.
- Misrouted packet: by detecting misrouting at the receiving router
- Starvation: using time-outs

These symptoms are monitored on a per-router or per-port basis and any diagnosis information is relayed back to the supervisor node that maintains the scoreboard. In addition to maintaining a scoreboard for diagnosing datapath faults, this node keeps a "status table" reflecting the state (working or faulty) of each link and router in the network. This table is updated upon receiving the diagnosis information from various nodes.

Note that the goal of diagnosis is not to find the exact cause of the faulty behavior, but to find the faulty component with sufficient resolution and accuracy to guide reconfiguration or repair. In the following subsections, each diagnosis scheme is discussed in detail. All of these schemes introduce minimal area overhead, using simple counters and tables, and thus to safeguard the detection hardware we can use established reliability mechanisms such as NMR [7], without introducing significant overhead.

1) Dropped or Spurious Flits: In an NoC, packets are typically broken down into fragments called "flits". The number of flits per packet can be extracted from the header flit or is implied in the case of fixed packet size. We use this information to detect dropped or spurious/duplicated flits by counting the number of flits sent from each input port (towards the output port), starting with the header flit

and stopping at the tail flit. If this count differs from its expected value, then either a flit was dropped or created within this input port or the connecting output port of the upstream router. Thus, to enable this scheme we only need one counter per input port bounded by the maximum size of any packet serviced by the network.

There are three cases in which this simple detection scheme might fail: First, when an entire packet is dropped or created. To cover this case, separate counters for counting packets are provided which will be discussed in Sec. III-B2. Second, if a tail flit is missing, the counter would never be compared against its expected value. For this case, we provide a timeout counter per input port that counts the number of cycles from the departure of the header flit and resets on departure of the tail flit. If this counter exceeds a threshold, then we infer that the tail flit was dropped. Finally, if the packet length information within the header is corrupted due to data corruption, then it could lead to incorrect diagnosis. When the supervisor node is notified about any data corruption, all the fault diagnosis information relayed from the nodes in the path of the corrupt packet is ignored. This avoids any bogus update to the "status table" due to incorrect diagnosis caused by data errors in the header flit.

2) Dropped or Spurious Packets: In order to detect a dropped or duplicated/spurious packet, we maintain a packet counter in each router, incrementing the counter at each packet tail flit received and decrementing the counter at each packet tail flit sent. This counter will be used to detect dropped and spurious packets.

Dropped packet: Packet drop inside a router means that there are more packets coming into the router than going out of it. Therefore, a dropped packet can be identified by checking whether the packet counter reaches zero at any point during a specified period of time or check window: if zero is never reached, it indicates that there has been a dropped packet. This can be implemented with a sticky flag register that is reset at the beginning of every checking window. This scheme eventually avoids false negatives, but may cause false positives in a fault-free router due to heavy traffic. We experimentally show in Sec. IV-D that choosing a suitable check window size can reduce the false positive rate to an acceptable level.

Spurious packet: A duplicated/spurious packet error is detected when the counter reaches a negative value. A negative value in the packet counter means that more packets have been sent out of the router than have entered the router. Note that this scheme is somewhat susceptible to false negatives: even if one or more packets are created within a router, the corresponding counter could remain at a non-negative value if the faulty router is congested and filled with packets for an extended period of time. However, the results in Sec. IV-E show that such false negatives are extremely rare for realistic traffic loads, and detection will eventually occur in the long term.

In summary, implementation of this scheme requires a counter of size bounded by the packet storage capacity of all buffers within a router, a timer to track timing windows for zero-checking, and a zero-observed storage bit, per router.

3) Misrouted packets: Misrouted packets are caused by faults in the control logic of the router, sending the packet to a port other than the one intended. This condition is often recoverable: a misrouted packet can be sent along to the correct destination even if it makes an unanticipated detour. However, a permanent misrouting fault can result in livelocks/deadlocks and should be avoided.

Detection of a misroute is highly dependent on the interconnect architecture and routing algorithm. In case of a deterministic routing scheme like X-Y routing, a lookup table or simple hardware assertion in the receiving router could be used to identify all misroutes. In the case of nondeterministic minimal-routing, misroutes which cause the packet to take a longer-than-minimal path can be similarly detected.

In order to differentiate transient and permanent misrouted faults, each router should keep track of the number of misrouted packets received by each port. If the number exceeds a threshold, the port will be declared defective (i.e. having a permanent fault) and rerouting/reconfiguration schemes can use the information to avoid the associated link. Aging can be used on the number of observed faults to further decrease the effect of transient faults.

4) Starvation: Starvation causes a flit or packet to wait indefinitely for access to an output channel. Thus, starvation can be detected using a timeout counter per input port. This counter, which counts the number of cycles that no output channel is granted to each input channel, is different from the one discussed in Sec. III-B1. In addition, the diagnosis information provided by this counter will indicate a specific faulty input port.

IV. Experimental Results

In this section, we present our experimental results to evaluate the proposed method.

A. Experimental Setup

Data corruption and misrouting diagnosis accuracy were assessed by Monte Carlo simulation using a C++ simulator supporting various routing algorithms on an 8-by-8 mesh. For dropped and spurious packet detection (which require a cycle-accurate model due to the use of counters), we used a cycle-accurate C++ NoC simulator based on [8]. The baseline system is an 8-by-8 XY-routed mesh network; routers are a simple 2-stage pipeline with no virtual channels and five entry deep buffers per input port. The NoC is augmented with the detection and diagnosis capabilities described in the previous sections. This framework was analyzed with two different types of workloads: random traffic and applications from the PARSEC suite [5]. A separate Verilog implementation of the same system was used to estimate the area overhead of the detection scheme. Synthesis was performed using Synopsys Design Compiler targeting the Artisan 45nm library.

B. Corrupted Data

It should be intuitive that the location of the faulty component is diagnosable with high confidence if a sufficiently large number of erroneous packets is observed. However, a key question is how sensitive the diagnosis accuracy is to the quantity of observed erroneous traffic.

Fig. 4 shows the diagnosis accuracy of our method for a selected faulty turn and link based on the total number of erroneous packets observed for both XY and nondeterministic minimal-routing. In all the cases, over 98% diagnosis accuracy is achievable after observation of fewer than 15 erroneous packets with XY-routing and around 40 packets with minimal-routing. This discrepancy is due to the fact that XY routing specifies the exact path, whereas in minimal-routing less information about the path is known.

Recall from Sec. III-A that not all permanent faults will be correctly identified, and some transient faults will be mistakenly classified as permanent. These scenarios are called a miss and a mistake, respectively. Fig. 5 shows the effect of the miss and mistake rates on the diagnosis accuracy. Observe that the accuracy is not affected severely even in the presence of many missed or mistakenly collected syndromes: since the accuracy has an asymptotic behavior with the count of observed erroneous packets as shown in Fig. 4,

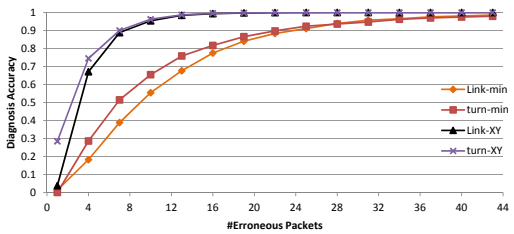

Fig. 4: **Diagnosis accuracy vs. erroneous packet count.** Our method is able to accurately diagnose the faulty turn/link after the transmission of limited number of erroneous packets, while the errors are corrected by ECC. More erroneous packets are required for diagnosis in case of having minimum-routing algorithm rather than XY-routing, since less information on the exact path taken by each packet is provided.

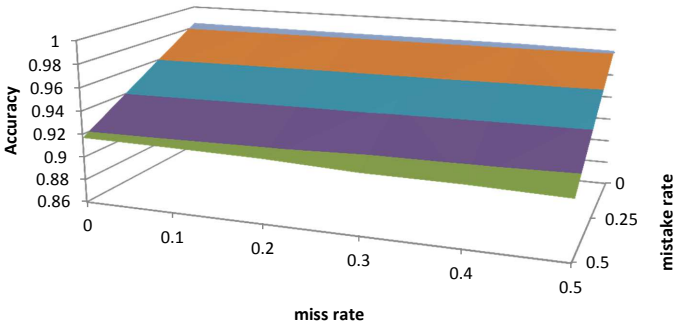

Fig. 5: **Effect of miss rate and mistake rate on diagnosis accuracy of a faulty turn in an 8-by-8 mesh using minimal-routing after sending 40 erroneous packets.**

a few missing fault reports will not severely decrease the accuracy. Mistakenly collected syndromes does not significantly affect accuracy either since they are scattered evenly on all the 64 bit positions while all the correct error syndromes are focused on the erroneous bit position.

C. Misrouted Packets

One of the diagnosis parameters is the number of observed misrouted packets from each port before declaring that port faulty: setting a higher threshold results in higher diagnosis confidence but requires more overall traffic to reach that decision. Fig. 6 shows the number of total packets required to have the desired number of erroneous packets activating the misrouting fault with a confidence of 99.7%. In larger networks, more total traffic is needed to ensure receiving enough packets activating the misrouting fault since each router has a lower probability of receiving a packet in general. Similarly, routers on the borders of the mesh need more total traffic. However, it should be noted that the performance overhead is regardless of the total traffic since only the erroneous packets need corrective measures.

D. Dropped Packet

As discussed in Section III-B2, our detection scheme for dropped packets can exhibit false positives. The false positive rate of the detection scheme depends on the duration of the *check window* and traffic conditions. False positives are triggered when the packet counter is non-zero for an entire *check window*; a heavily loaded network will see more false diagnoses as packets accumulate at router buffers due to congestion. Intuitively, a longer *check window* will reduce the false positive rate by allowing more time for packets to clear a routers buffers.

Fig. 7 shows the decrease in false positive rate with increasing *check window* size. It can be seen that a heavily loaded network exhibits a higher false positive rate than a moderately loaded network, and hence requires a larger *check window* size to limit false positives. Additionally, networks operating with smaller packets, exhibit a greater false positive rate as more number of packets can reside at

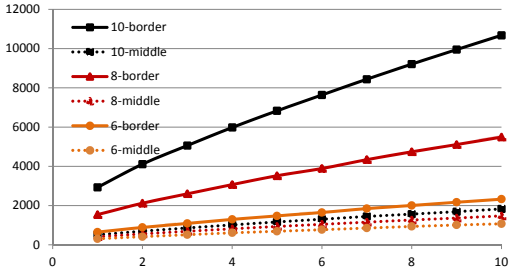

Fig. 6: **The number of total packets (Y-axis) Vs. number of erroneous packets activating misrouting fault (X-axis).** Results are shown for different mesh sizes (6-by-6, 8-by-8, and 10-by-10) and different places in the mesh (in the middle and on border). Routers in larger meshes and those on the borders need higher number of total packets since they have a lower probability of receiving a packet activating the misroute fault.

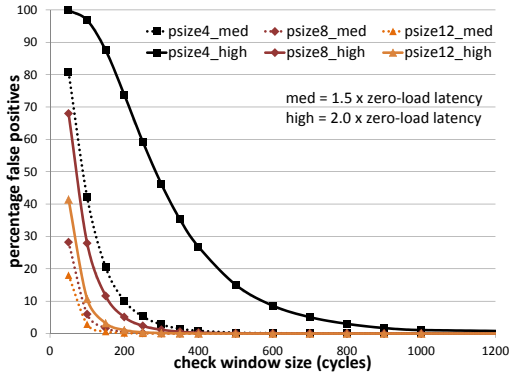

Fig. 7: **Dropped packet detection scheme under uniform random traffic.** The false positive rate with increasing *check window* size is plotted for various packet sizes (4, 8, 12 flit-per-packet) at medium (1.5 x zero-load latency) and high (2 x zero-load latency) injetion loads. False positive rate drops rapidly with larger *check window* sizes and decreasing network load. Additionally, networks with shorter packets exhibit more false positives.

any router's buffer for extended periods of time, causing the packet counter to not touch zero values. In all cases, the false positive rate drops to a negligible value beyond a certain *check window* size ($Window_{min}$).

To calibrate the *check window*, we ran rigorous simulations, operating the network normally (*i.e.* fault free) for a preset length of time and calculating the false positive rate for a range of *check window* values. Fig. 8 plots $Window_{min}$ necessary to eliminate false positives and the average network latency as network load is increased, under uniform network traffic. $Window_{min}$ exhibits a slow increase with rising injection rate up to network saturation, and a steep rise afterwards. From the plot, a worst-case $Window_{min}$ of 1K cycles is sufficient to eliminate all false positives when the network is in deep saturation, operating at an average latency of well over 3 times the zero-load latency. A similar result was observed for nine different PARSEC benchmark traces (1 million instructions each), where a *check window* size of 400 cycles was sufficient to eliminate all false positives for every benchmark. Our simulations indicate that $Window_{min}$ rises to high values only when the network is operated at loads well past saturation. Such a scenario is unlikely: NoC workloads are characterized by the self-throttling nature of the applications, which prevents them from operating past saturation loads [18].

E. Spurious Packet

As discussed in Section III-B2, our detection scheme for spurious packets can exhibit false negatives, while false positives are not possible. Hence a different methodology is required to evaluate this scheme: after operating the network fault-free for a preset length

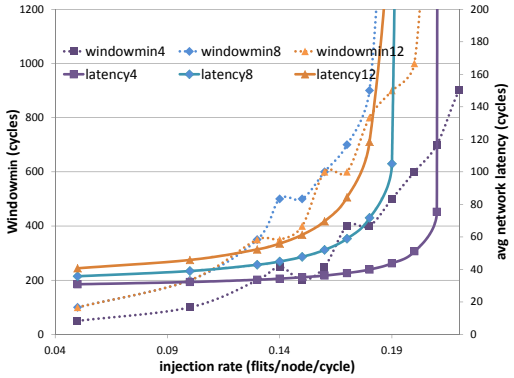

Fig. 8: **Dropped packet detection scheme under uniform traffic.** The variation of $Window_{min}$ and latency with increasing network load is shown for various packet sizes (4, 8, 12 flit-per-packet). A high $Window_{min}$ is required only when the network is deeply saturated.

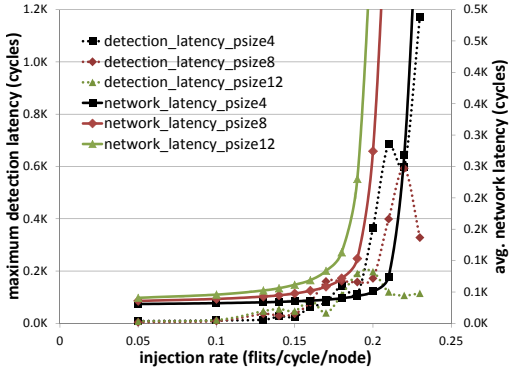

Fig. 9: **Created packet detection scheme under uniform traffic.** Figure shows The variation of maximum detection latency (create packet) and average network latency with increasing network load. Detection latency is within tolerable limits for deeply saturated networks.

of time, a spurious packet is intentionally created with a certain probability and written to the buffer of a randomly chosen router. Detection latency is defined as the time required to detect the error after the packet was created. A spurious packet is flagged as false negative if the detection latency exceeds 10K cycles. To gain statistical confidence, we inject 10,000 such faults one after another and repeated each simulation 10 times for different randomization seeds. Fig. 9 shows the results of such a study. Maximum detection latency and average network latency are plotted for different packet sizes as we increase the network load. Again, maximum detection latency increases slowly up to a certain network load after which there is a steep increase. As we can see, worst-case detection latency is within 1.2K cycles for networks operating at average latency of more than 10 times the zero-load latency, and recall that on-chip networks do not operate at deep saturation points [18]. Similarly, for the nine PARSEC benchmark traces, a maximum detection latency of 302 cycles was observed. No false negatives were observed for the network loads shown in Fig. 9.

F. Area Results

In this paper, our main goal is to provide a high diagnosis resolution while keeping the area overhead at a minimum. Our diagnosis scheme is based on a software implemented scoreboard, end-to-end ECC, and low-overhead per-router counter-based diagnosis hardware and misrouting detectors described in previous sections. In addition, faults inside these additional detection and diagnosis hardware are detected using Double modular redundancy (DMR); a viable solution due to our low-overhead implementation. Our synthesis results show that the overall area overhead after adding DMR is less than **10%**, as shown in Table I. Same baseline router as Vicis [12] is used here which is consistent with the Intel Polaris router [23]. As it is shown in Table I, the majority **7%** area overhead is due to DMR enabled ECC encoders and decoders necessary to detect data corruption. Note that, the area overhead of our scheme is less than **5%**, if DMR is not employed.

Design	**Area** (um^2)	**%**
Baseline Router	115898	100.00
Datapath ECC Encoder/Decoder (DMR)	8026	6.92
Control-logic Diagnosis Hardware (DMR)	3168	2.73
Total Overhead	11194	**9.65**

TABLE I: **Area overhead (per router).**

V. Conclusion

In this paper, a comprehensive method was introduced to detect and diagnose permanent faults in datapath and control logic of an on-chip network. ECC is utilized to detect and correct data errors. In case of an error detection, an information packet is sent to a supervisor node that is used to measure the fault probability of different components and pinpoint the defective one to the resolution of a specific port, link, or turn. Control errors such as dropped/spurious packets/flits are diagnosed via per router counters measuring the number of flits/packets passing through every port/router in a predefined activity window, and misrouted packets are found according to the routing algorithm.

Our method imposes no performance overhead on the system in the case of an error-free network. There is no system down time to test the system and no specific traffic is sent to test different components. The high resolution provided by our method can be utilized to improve the efficiency of reconfiguration/rerouting schemes.

Acknowledgments

The authors acknowledge the support of the National Science Foundation Center for Domain-Specific Computing (CDSC) and the Gigascale Systems Research Center (GSRC), one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

References

[1] Adapteva. *Adapteva Epiphany multi core architecture*.
[2] V. Agarwal, M.S. Hrishikesh, S.W. Keckler, and D. Burger. Clock rate versus IPC: The end of the road for conventional microarchitectures. In *Proc. ISCA*, 2000.
[3] K. Aisopos, A. DeOrio, Li-S. Peh, and V. Bertacco. ARIADNE: Agnostic reconfiguration in a disconnected network environment. In *Proc. PACT*, 2011.
[4] A. Alaghi, Karimi. N., M. Sedghi, and Z. Navabi. Online NoC switch fault detection and diagnosis using a high level fault model. *Proc. DFT*, 2007.
[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proc. PACT*, October 2008.
[6] S. Borkar. Design challenges of technology scaling. *Micro, IEEE*, 1999.
[7] 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 *Proc. HPCA*, 2006.
[8] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.
[9] G. de Micheli and L. Benini. Networks on chip: A new paradigm for systems on chip design. In *Proc. DATE*, 2002.
[10] A. DeOrio, K. Aisopos, V. Bertacco, and Li-S. Peh. DRAIN: Distributed recovery architecture for inaccessible nodes in multi-core chips. In *Proc. DAC*, 2011.
[11] A. Dutta and N. A. Touba. Reliable network-on-chip using a low cost unequal error protection code. In *Proc. DFT*, 2007.
[12] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester. Vicis: A reliable network for unreliable silicon. In *Proc. DAC*, 2009.
[13] International Technology Roadmap for Semiconductors (ITRS), 2010.
[14] A. P. Frantz, F. L. Kastensmidt, L. Carro, and E. Cota. Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk. In *Proc. ITC*, 2006.
[15] S. Gupta, S. Feng, A. Ansari, J. Blome, and S. Mahlke. The stagenet fabric for constructing resilient multicore systems. In *Proc. MICRO*, 2008.
[16] Intel Corporation. *Intel Core2 Extreme Quad-Core Processor QX6000*.
[17] M. Mehrara, M. Attariyan, S. Shyam, K. Constantinides, V. Bertacco, and T. Austin. Low-cost protection for ser upsets and silicon defects. In *Proc. DATE*, 2007.
[18] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu. Next generation on-chip networks: What kind of congestion control do we need? In *Proc. Hotnets*, 2010.
[19] A. Pellegrini and V. Bertacco. Application-aware diagnosis of runtime hardware faults. In *Proc. ICCAD*, 2010.
[20] S. Shamshiri and K.-T. Cheng. Error-locality-aware linear coding to protect multi-bit upset in srams. In *Proc. ITC*, 2010.
[21] S. Shamshiri, A. Ghofrani, and K.-T. Cheng. End-to-end error correction and online diagnosis for on-chip networks. In *Proc. ITC*, 2011.
[22] S. Shyam, K. Constantinides, S. Phadke, V. Bertacco, and T. Austin. Ultra low-cost defect protection for microprocessor pipelines. 2006.
[23] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 2008.
[24] N.J. Wang, J. Quek, T.M. Rafacz, and S.J. Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Proc. DSN*, 2004.
[25] F. Worm, P. Thiran, G. de Micheli, and P. Ienne. Self-calibrating networks-on-chip. In *Proc. ISCAS*, 2005.