Notice: This paper was not presented by one of the authors in ASP-DAC 2011.

# NS-FTR: A Fault Tolerant Routing Scheme for Networks on Chip with Permanent and Runtime Intermittent Faults

Sudeep Pasricha, Yong Zou

Colorado State University, Fort Collins, CO
{sudeep, yong.zou}@colostate.edu

**Abstract – In sub-65nm CMOS technologies, interconnection networks-on-chip (NoC) will increasingly be susceptible to design time permanent faults and runtime intermittent faults, which can cause system failure. To overcome these faults, NoC routing schemes can be enhanced by adding fault tolerance capabilities, so that they can adapt communication flows to follow fault-free paths. A majority of existing fault tolerant routing algorithms are based on the turn model approach due to its simplicity and inherent freedom from deadlock. However, these turn model based algorithms are either too restrictive in the choice of paths that flits can traverse, or are tailored to work efficiently only on very specific fault distribution patterns. In this paper, we propose a novel fault tolerant routing scheme (NS-FTR) for NoC architectures that combines the North-last and South-last turn models to create a robust hybrid NoC routing scheme. The proposed scheme is shown to have a low implementation overhead and adapt to design time and runtime faults better than existing turn model, stochastic random walk, and dual virtual channel based routing schemes.**

## I. Introduction

Chip multiprocessors (CMPs) with multiple processing and memory cores integrated on a single chip are today at the heart of many computing devices. As digital convergence continues to increase application complexity, aggressive CMOS technology scaling has allowed CMP designs to keep pace by allowing more and more cores to be integrated in the same area budget with each successive technology generation. However, with scaling heading towards nanometer geometries, a major challenge that has appeared on the horizon is the increased likelihood of failure due to permanent, transient, and intermittent faults caused by a variety of factors that are becoming more and more prevalent. *Permanent* faults occur due to manufacturing defects, or after irreversible wearout damage due to electromigration in conductors, negative bias temperature instability (NBTI), dielectric breakdown, etc [1][2]. *Transient* faults (also called soft errors) occur when an event such as high-energy cosmic neutron particle strike, alpha particle strike due to trace uranium/thorium impurities in packages, capacitive and inductive crosstalk, electromagnetic noise, etc. causes the deposit or removal of enough charge to invert the state of a transistor, wire, or storage cell [3][4]. These errors occur for a very short duration of time and can be hard to predict. A third class of faults, called *Intermittent* faults, occurs frequently and irregularly for several cycles, and then disappears for a period of time [5][6]. These faults commonly arise due to process variations combined with variation in the operating conditions, such as voltage and temperature fluctuations.

On-chip interconnect architectures are particularly susceptible to faults that can corrupt transmitted data or altogether prevent it from reaching its destination. Reliability concerns in sub-65nm nodes have in part contributed to the shift from traditional bus based communication fabrics to network-on-chip (NoC) architectures that provide better scalability, predictability, and performance than buses [7][8]. The inherent redundancy in NoCs due to multiple paths between packet sources and sinks can greatly improve communication fault resiliency. Several CMP designs are emerging that make use of NoCs as interconnection fabrics [9]-[12]. To ensure reliable data transfers in these communication fabrics, utilizing fault tolerant design strategies is essential. Traditionally, error detection coding and retransmission has been a popular means of achieving resiliency towards transient and intermittent faults [13][14]. Alternatively, forward error correction (FEC) schemes can provide better resiliency against these faults, but usually at a higher performance and energy overhead [15]. Circuit and layout optimizations such as shield insertion and wire sizing to reduce crosstalk induced transient faults have also been proposed [16][17]. To overcome permanent faults in NoCs, fault tolerant routing schemes are a critical requirement and have been the focus of several research efforts over the last few years [20]-[38]. In the presence of intermittent or permanent faults on NoC links and routers, routing schemes can ensure error free data flit delivery by using an alternate route that is free of faults.

In this paper, we propose a new fault tolerant routing scheme (NS-FTR) for 2D mesh NoCs that combines the North-last and South-last turn models together with opportunistic replication to increase successful packet arrival rate while optimizing energy consumption. For the first time, we explore the performance of fault tolerant routing schemes in the presence of both runtime intermittent and design time permanent faults. Compared to our recently proposed fault tolerant routing algorithm (OE+IOE [39]), the proposed NS-FTR algorithm has much lower implementation overhead. Our extensive experimental results indicate that NS-FTR outperforms OE+IOE as well as other state-of-the-art NoC fault tolerant routing schemes based on single and hybrid turn models, and stochastic random walk heuristics.

## II. Related Work

NoC routing schemes can be broadly classified as either static (also called deterministic or oblivious) or dynamic (also called adaptive). While static routing schemes [18][19] use fixed paths and offer no fault resiliency, dynamic (or adaptive) routing schemes [20]-[38] can alter the path between a source and its destination over time as traffic conditions and the fault distribution changes. The design of adaptive routing schemes is mainly concerned with increasing flit arrival rate, avoiding deadlock, and trying to use a minimal hop path from the source to the destination to decrease transmission energy. Unfortunately, these goals are often conflicting, requiring a complex trade-off analysis that is rarely addressed in existing literature. In general, fault tolerant routing schemes can be broadly classified into three categories: *(i)* stochastic, *(ii)* fully adaptive, and *(iii)* partially adaptive.

Stochastic routing algorithms provide fault tolerance through data redundancy by replicating packets multiple times and sending them over different routes [20]. For instance, the probabilistic gossip flooding scheme [21] allows a router to forward a packet to any of its neighbors with some pre-determined probability. Directed flooding refines this idea by preferring hops that bring the packet closer to its addressed destination [22]. N-random walk [22] limits the number of packet copies by allowing replications at the source only. In the rest of the network, these copies stochastically take different routes without further replication. The major challenges with these approaches are their high energy consumption, strong likelihood of deadlock and livelock, overhead of calculating, storing, and updating probability values, and poor performance even at low traffic congestion levels.

Fully adaptive routing schemes make use of routing tables in every router or network interface (NI) to reflect the runtime state of the NoC and periodically update the tables when link or router failures occur to aid in adapting the flit path [23]-[25]. However, these schemes have several drawbacks, including *(i)* the need for frequent global fault and route information updates which can take thousands of cycles at runtime during which time the NoC is in an unstable state, *(ii)* lack of scalability – table sizes increase rapidly with NoC size, increasing router (or NI) area, energy, and latency, and *(iii)* strong possibility of deadlock, unless high overhead deadlock recovery mechanisms such as escape channels are used.

Partially adaptive routing schemes enable a limited degree of adaptivity, placing various restrictions on routing around faulty nodes, primarily to avoid deadlocks. Turn model based routing algorithms such as negative-first, odd-even, north-last, and south-last [26]-[29] are examples of partially adaptive routing, where certain flit turns are forbidden to avoid deadlock. [30] combines the XY and YX schemes to achieve fault resilient transfers. [31] combines routing tables with a customized turn model to avoid deadlock. Unfortunately, the degree of adaptivity provided by these routing algorithms is highly unbalanced across the network, which in some cases results in poor performance.

A few works have proposed using convex or rectangular fault regions with turn models [32]-[38]. Routing detours around these regions can be selected so that deadlock-prone cyclic dependencies are impossible, such as by using turn models or cycle free contours. In general, fault tolerant routing schemes that make use of fault regions are either too conservative (e.g., disabling fully functional routers to meet region shape requirements), have restrictions on the locations of the faults that can be bypassed, or cannot adapt to runtime intermittent/permanent faults.

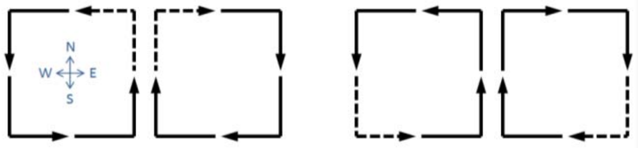
Our recent work [39] combined the OE and IOE turn models to create a hybrid fault tolerant routing scheme (OE+IOE) that achieved high flit arrival rates. In contrast, this work proposes a new fault tolerant routing algorithm (NS-FTR) that combines the North-last (NL) and South-last (SL) turn models to create a hybrid routing scheme with better fault tolerance and lower energy, latency, and area overhead. NS-FTR balances data replication and resource and path redundancy to optimize power dissipation while ensuring increased tolerance to design time permanent and runtime intermittent faults. Data replication is limited to implementations with high fault rates, which ensures a low cost operation in the absence of faults (or for scenarios with very few faults). Path diversity as a result of utilizing dual turn models ensures that even if one packet hits a dead end, the other has a possibility of arriving at the destination. Our experimental results (Section IV) that compare NS-FTR with several state-of-the-art fault tolerant routing schemes make a strong case for considering NS-FTR as a viable fault tolerant routing scheme for emerging CMPs.

## III. NS-FTR Routing Scheme

In this section we present an overview of our proposed fault tolerant routing scheme (NS-FTR) for 2D mesh NoCs. Section *A* introduces north last (NL) and south last (SL) turn model routing. Section *B* presents details of the implementation and operation of the proposed routing scheme that combines the NL and SL turn models. Finally, Section *C* describes the router architecture and control network that supports our NS-FTR scheme in a NoC fabric.

### *A. NL and SL Turn Models*

The north last (NL) turn model for deadlock-free routing was first introduced by Glass et al. [26] for large scale off-chip 2D mesh networks. A turn in this context refers to a 90-degree change of traveling direction for a flit. There are eight types of turns that are possible in a 2D mesh based on the traveling directions of the associated flits. A deadlock in wormhole routing can occur because of flits waiting on each other in a cycle. In the NL turn model deadlock-free routing is achieved by prohibiting two out of the eight possible turns, as shown in Fig. 1(a) where dashed arrows indicate prohibited turns. A flit in the NL turn model is initially routed in the E, W, or S directions before finally turning in the N direction, after which it cannot make further turns. In a similar manner, the south last (SL) turn model ensures deadlock-free routing by prohibiting two out of the eight possible turns as shown in Fig. 1(b). A flit in the SL turn model is initially routed in the E, W, or N directions before finally turning in the S direction, after which it cannot make further turns.



(a) (b)

Fig. 1: Turns allowed in the (a) North-last (b) South-last algorithms

### *B. NS-FTR Routing Scheme: Overview*

To ensure robustness against faults, redundancy is a necessary requirement, especially for environments with high fault rates and unpredictable fault distributions. As the level of redundancy is increased, system reliability improves as a general rule. However, redundancy also detrimentally impacts other design objectives such as power and performance. Therefore in practice it is important to limit redundancy to achieve a reasonable trade-off between reliability, power, and performance. Unlike directed and random probabilistic flooding algorithms that propagate multiple copies of a packet to achieve fault tolerant routing in NoCs, the proposed NS-FTR scheme sends only one redundant packet for each transmitted packet, and only if the fault rate is above a replication threshold δ. The original packet is sent using the north last (NL) turn model while the redundant packet is propagated using south last (SL) turn model based routing scheme. The packet replication happens only at the source. Two separate virtual channels (VCs), one dedicated to the NL packets and the other for the SL packets ensure deadlock freedom. If the fault rate is low (i.e., below δ), replication is not utilized and only the original packet is sent using the NL scheme while power/clock gating the SL virtual channel to save power. The value of δ is a designer-specified parameter that depends on several factors such as the application characteristics, routing complexity, and power-reliability trade-off requirements.

The NS-FTR routing algorithm prioritizes minimal paths that have higher probabilities of reaching the destination even if faults are encountered downstream. Minimal paths and the replication threshold ensure low power dissipation under a diverse set of fault rates and distributions. No restriction on the number or location of faults is assumed, but the routers must know which of their adjacent (neighbor) links/nodes are faulty, which is accomplished using control signaling. The proposed routing approach can be combined with error control coding (ECC) techniques for transient fault resiliency and optimizations such as router buffer reordering [40] and router/NI backup paths [41] to create a comprehensive fault tolerant NoC fabric. In the following subsections, we describe the implementation of the proposed NS-FTR scheme.

#### *B.1 Turn Restriction Checks*

Whenever a packet arrives at a router in the NS-FTR scheme, three scenarios are possible: *(i)* there is no fault in the adjacent links and the packet is routed to the output port based on the NL (or SL) scheme that selects a minimal path to the destination, *(ii)* there are one or more faults on adjacent links that prevent the packet from propagating in a valid direction (an output port with a fault-free link that does not violate turn model routing rules) in which case the packet must be dropped as a back turn is not allowed, and *(iii)* one or more adjacent links have faults but a valid direction exists to route the packet towards, based on the NL (or SL) rules. A packet traversing an intermediate router must choose among four output directions (N, S, W, E). However, choosing some directions may lead to a violation of a basic NL (or SL) turn rule immediately or downstream based on the relative location of the destination.

Fig.2 shows the SystemC [44] pseudocode of the turn restriction check phase for the NL turn model in our router model to aid in detecting an invalid routing direction. The input to the turn restriction check phase is a potential output port direction for a header flit currently in the router input buffer, and the output of the phase indicates whether the output port direction is valid or not.

First, we check whether the output port direction has a fault in its attached adjacent link (steps 3-4) or is a back turn (steps 5-6), in which case this is an invalid direction. Next, we check for scenarios where choosing the given output port direction may lead to violations of the NL turn model rules immediately or eventually at some point downstream. In steps 7-8, we check for the scenario where the packet arrives in the same column as its destination which is located in the N direction. If the packet attempts to make a turn to E, W, or S, it will violate the basic NL turn model rules, and this is not allowed. Steps 9-12 test for two cases when the packet cannot be routed in the N direction without violating the NL turn model rules: if the current router node is already to the north of the destination code, or if the current node is not on the same column as its destination node. Finally, in steps 13-18, we check for a particular scenario in which the current node is on the southern border of the mesh, and with the packet attempting to go in the opposite direction to where the destination is located (e.g., going W when the destination node is in the E direction; steps 14-15). As the packets can go north only once they arrive at the column of their destination nodes, going in the opposite direction will eventually violate the NL turn model rules and is thus not allowed.

**Pseudocode:** Turn Restrictions for North-last Implementation

```
1:  RESULT   check_neighbor(UI ip_dir, UI direct, ULL cur_id, ULL dest_id)
        //ip_dir: direction where the packet comes from
        //direct: direction we want to check
        //cur_id: current node position
        //dest_id: destination node position.
        //dest_yco,dest_xco: y,x coordinates of destination node
        //cur_yco,cur_xco: y,x coordinates of current node
        //dif_yco=dest_yco-cur_yco
        //RESULT: LINK_FAULT, TURN_FAULT, BACK_TURN, OK
2:  {
3:      if (check_linkfault(direction)==LINK_FAULT)
4:          return LINK_FAULT;
5:      if (ip_dir==direct)
6:          return BACK_TURN;
7:      if(((cur_yco==dest_yco)&&(cur_xco>dest_xco))&&(direct!=N))
8:          return TURN_FAULT;
9:      if((cur_xco<dest_xco)&&(direct==N))
10:         return TURN_FAULT;
11:     if((cur_yco!=dest_yco)&&(direct==N))
12:         return TURN_FAULT;
13:     if(borderS(cur_id)) {
14:         if((dest_yco>cur_yco)&&direct==W)
15:             return TURN_FAULT;
16:         else if((dest_yco<cur_yco)&&direct==E)
17:             return TURN_FAULT;
18:     }
19:         return OK;
20: }
```

Fig.2. SystemC pseudocode for NL turn restriction checking

The turn restriction check phase described above is invoked in parallel for all four possible directions in an intermediate node for a header flit. Fig. 3 shows the circuit level diagram for the NL turn restriction check phase, implemented in the route compute unit of a NoC router. The circuit is replicated at each input port of the router. At the TSMC 65nm technology node, the circuit dissipates 1.34 µW of power on average, and has a critical path latency of 0.48 ns. The pseudocode and circuit level diagrams for the SL turn model are similar and not shown here for brevity. In contrast, the circuit level implementation overhead for the turn restriction check phase in our previous work that combined the OE and IOE turn models [39] was much greater, with 2.03 µW of power dissipation on average, and a critical path latency of 0.54 ns at the 65 nm node. Section IV presents results to compare the fault tolerance capabilities of these two schemes, as well as several other schemes proposed in literature.

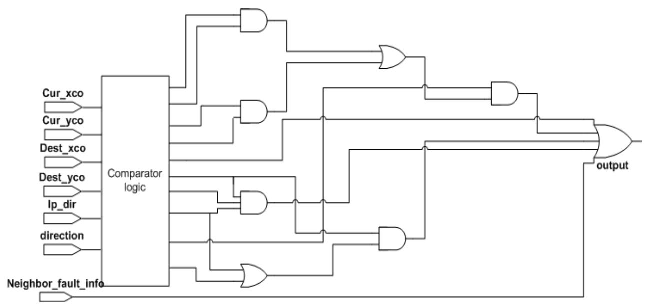


Fig.3. NL turn restriction check phase circuit diagram

### B.2 Prioritized Valid Path Selection

After the turn restriction checks, it is possible for a packet to have multiple valid directions to choose from. In the NL scheme, we identify two scenarios where such a choice between valid paths exists and show how NS-FTR selects among them. Again, we omit describing analogous scenarios for the SL scheme for brevity.

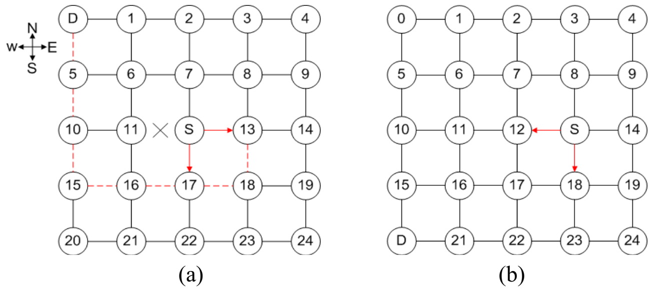


Fig.4. Path selection scenarios for NL turn scheme

Fig. 4(a) demonstrates an example of the first scenario. “S” and “D” are the source and destination nodes. With no faults, the NL minimal route would take the path S-11-10-5-D. Suppose that the link between S and node 11 has a fault. Then two choices exist: sending the packet to node 13 or 17. In this situation, we should choose node 17 as the resulting path is shorter by two hops than if node 13 is selected. A shorter path is preferred as it has lower probability of encountering a fault and consumes lower energy. This example can be generalized as follows: if a packet is to the SE or SW of its destination node, and the minimal path cannot be chosen due to a fault, then we give the S direction higher priority.

Fig. 4(b) shows the second scenario. At source S, a packet intended for destination D can be sent along two minimal paths. We observe that if the packet is sent along the path with node 18, the only paths to D that are available are S-18-17-16-15-D, S-18-17-16-21-D, S-18-17-22-21-D, S-18-23-22-21-D, and S-18-19-24-23-22-21-D. In contrast, if the packet is sent along the path with node 12, many more paths exist. This greater path diversity is critical in the presence of faults, as it can ensure a better probability of packet arrival at D. Again we can generalize the example as follows: if a packet is at the NE (or NW) of the destination and if multiple valid minimal paths exist, then we prioritize the E (or W) directions.

### C. Router Architecture and Control Network

We make use of a dual virtual channel (VC) router architecture. Packets traversing the network have a header flit with a destination ID field that is used to determine the output port at each router. The router datapath consists of the buffers and the switch. The input FIFO buffers store up to 16 flits waiting to be forwarded to the next hop. There are two input FIFO buffers each dedicated to a VC, with one VC for the NL routed packets and the other for the SL routed packets. When a flit is ready to move, the switch connects an input buffer to an appropriate output channel. To control the datapath, the router contains three major control modules: a route compute control unit (RC_CTRL), a virtual-channel (VCA) allocator, and a switch allocator (SA). These control modules determine the next hop direction, the next virtual channel, and when a switch is available for each packet/flit. The routing operation takes four phases: route computation (RC), virtual-channel allocation (VCA), switch allocation (SA), and switch traversal (ST). When a header flit (the first flit of a packet) arrives at an input channel, the router stores the flit in the buffer for the allocated VC and determines the next hop for the packet using the checks and priority rules described in the previous sections (RC phase). Given the next hop, the router then allocates a virtual channel (VCA phase). Finally, the flit competes for a switch (SA phase) if the next hop can accept the flit, and moves to the output port (ST phase). In our scheme, a packet carries its virtual channel (NL or SL) information in a bit in the header flit, therefore the output port and the destination VC can be determined early, at the end of the RC stage.

To reduce energy consumption, if the fault rate is below the replication threshold δ, the RC control unit shuts down the SL virtual channels in the router. We assume that dedicated control signals connected to a lightweight fault detection unit (FDU) can allow estimation of the fault rate at runtime in the NoC, and the decision to change the replication status (initiate or cut-off) can be communicated to the nodes and implemented with a delay of at most a few hundred cycles. Typically, such a change in status would be a rare occurrence as intermittent or permanent faults do not appear at runtime as frequently as transient faults.

An important requirement for any fault tolerant NoC fabric is a control network that can be used to send acknowledgement signals to inform the source whether the packet successfully arrived at the destination. We assume a lightweight and fault-free control network that is topologically similar to the data network for the purpose of routing ACK signals from the destination to the source node on successful packet arrival, or NACK signals from an intermediate node to the source node if the packet must be dropped due to faults that make it impossible to make progress towards the destination. In our implementation, a source can re-send a packet at most twice after receiving NACK signals, before assuming that the packet can never reach its destination. While this can signal unavoidable failure for certain types of applications, other applications may still be able to operate and maintain graceful degradation. For instance, a multiple use-case CMP can shut down certain application use-cases when their associated inter-core communication paths encounter fatal faults, but there might be other use cases that can still continue to operate on fault-free paths. We assume that a higher level protocol (e.g. from the middleware or OS levels) can step in to initiate thread migration and other strategies to ensure useful operation in the presence of unavoidable system faults.

## IV. Experimental Studies

### *A. Evaluation Setup*

We implemented and evaluated our proposed algorithm using a SystemC based cycle-accurate 2D mesh NoC simulator that was created by extending the open source Nirgam simulator [42] to support faults and different fault tolerant routing schemes. In addition to communication performance estimates, we were interested in NoC energy estimation as well. For this purpose we incorporated dynamic and leakage power models for standard NoC components from Orion 2.0 [43] into the simulator. We also incorporated the power overhead of the circuits in the routers (obtained after synthesis) required to realize the fault tolerant routing algorithms. The NoC fabric was clocked at 1 GHz. The target implementation technology was 65nm, and this node was used to determine parameters for delay and energy consumption. A 9×9 (i.e., 81 cores) CMP with a mesh NoC was considered as the base system in our experiments. The stimulus for the simulations was generated for specific injection rates using a uniform random traffic model. A fixed number of flits (3000/core) were transmitted according to the specified traffic pattern in the simulations, to enable comparisons for not only successful arrival rates, but also for overall communication energy and average packet latency.

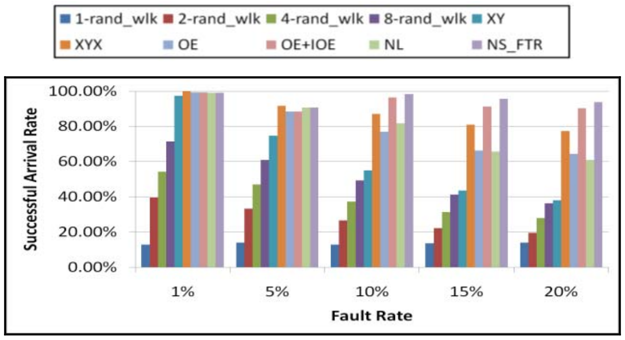
We modeled two types of faults in the NoC: *(i)* design time permanent faults that are randomly distributed link-level hard failures, and *(ii)* runtime intermittent faults, which are also randomly distributed and appear due to reasons such as periods of thermal hotspots and high crosstalk noise during system operation. For intermittent faults, we assumed a fault duration of 5000 cycles, with the fault location being randomly generated, as for permanent faults. Ten different random fault distributions were generated and analyzed for each fault rate to obtain a more accurate estimate of fault resiliency for the routing schemes. To ensure fair comparison, the randomized fault locations and durations were replicated across simulation runs for different fault tolerant routing schemes, to study the performance of the schemes under the same fault conditions.

### *B. Comparison with Existing FT Routing Schemes*

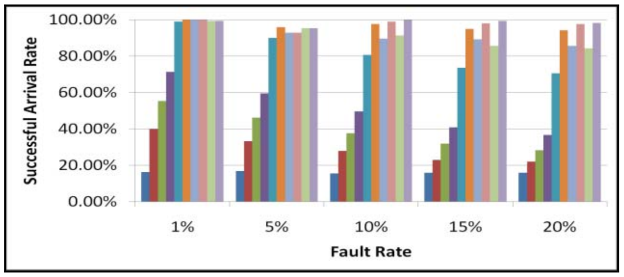
Our first set of experiments compared the packet arrival rates for the proposed NS-FTR fault tolerant routing scheme with several fault tolerant routing schemes from literature. The following schemes were considered in the comparison study: *(i)* XY dimension order routing [18] (although not inherently fault tolerant, it is considered here because of its widespread use in 2D mesh NoCs), *(ii)* N-random walk [22] for N = 1,2,4,8 *(iii)* adaptive OE turn model [29], *(iv)* adaptive North-last (NL) turn model [26], *(v)* adaptive OE+IOE [39] which combines replication with the OE and IOE turn models on two VCs, and *(vi)* XYX [30] which combines replication with the XY and YX routing schemes on two VCs. Other fault tolerant routing schemes such as probabilistic flooding [21] and fully adaptive table based routing [24] were considered but their results are not presented because these schemes have major practical implementation concerns - their energy consumptions is an order of magnitude higher than other schemes and frequent deadlocks hamper overall performance. For our NS-FTR scheme, we used a prioritized valid path selection introduced in Section III.B.2 and a replication threshold value of δ = 6%, based on our extensive simulation studies that showed a good trade-off between arrival rate and energy for 8% ≥ δ ≥ 2%.

Fig. 5 (a)-(c) compare the successful packet arrival rate for the fault tolerant routing schemes described above under different fault rates and types. Results are shown for a 20% injection rate (0.2 flits/node/cycle). The arrival rates are shown for permanent faults (Fig. 5(a)), intermittent faults (Fig. 5(b)), and an equal distribution of permanent and intermittent faults (Fig. 5(c)). Not surprisingly, arrival rates for all routing schemes in general are higher under intermittent faults than under permanent faults, as intermittent faults only last for a short period of time. On analyzing individual routing scheme performance, it can be immediately seen that the N-random walk schemes have a low arrival rate even in the absence of faults. This is due to frequent deadlocks during simulation which the scheme is susceptible to, causing packets to be dropped. The best arrival rate for the N-random walk scheme is achieved for a value of N=8. Out of the turn/dimension-order based routing schemes, XY not surprisingly performs worse than the other schemes due to the lack of any built in fault tolerance capability. The arrival rates for the NL and OE routing schemes are comparable. Among the dual VC schemes, OE+IOE outperforms XYX due to greater path diversity with the less restrictive OE and IOE turn model schemes than with XY or YX schemes. However, our NS-FTR scheme has a higher successful packet arrival rate than OE+IOE and every other scheme we compared. For fault rates under 6%, replication in NS-FTR is disabled to save energy (discussed later), which explains the slight dip in successful arrival rate for low fault rates. However, with replication enabled, NS-FTR outperforms every other scheme at low fault rates as well. NS-FTR scales particularly well under higher fault rates that are expected to be the norm in future CMP designs.

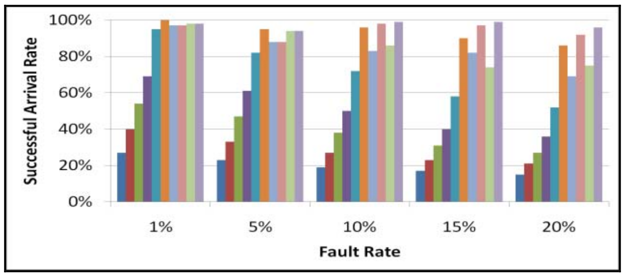
In addition to arrival rate, an increasingly important metric of concern for designers is the energy consumption in the on-chip communication network. Fig. 6 shows the energy consumption for the different routing schemes under the same fault rates and types, and injection rates as in the previous experiment. The energy consumption accounts for circuit level implementation overheads as well as acknowledgements and retransmission in the schemes. Energy consumption under intermittent faults is lower than under permanent faults as intermittent faults last for only a short amount of time, incurring lower retransmission and non-minimal path selection overheads in the schemes. It can be seen that the N-random walk fault tolerant routing scheme has significantly higher energy consumption compared to the other schemes. This is due to the high level of replication with increasing values of N, as well as because of the non-minimal paths chosen by default to route the packets on. For single VC schemes (OE, NL, XY), the XY scheme has the lowest energy dissipation which explains its popularity in NoCs today. But as it is lacking in any fault tolerance, the scheme may not be a viable option for future CMPs.



(a)



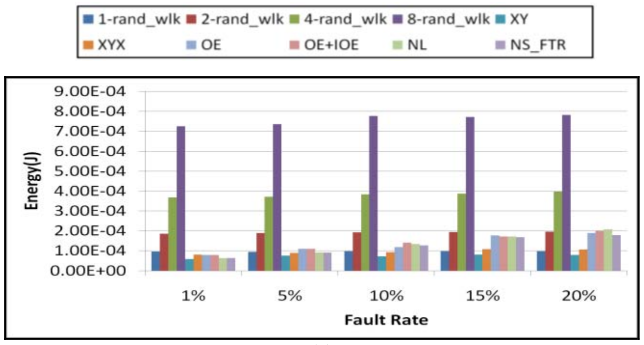
(b)



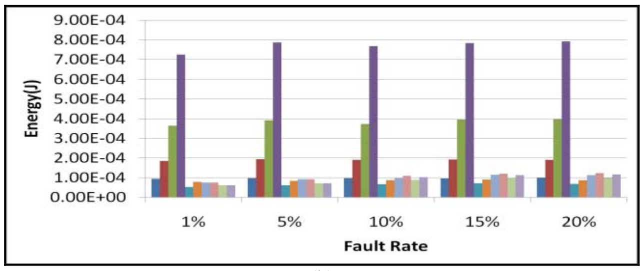
(c)

Fig. 5 Packet arrival rate for 9×9 NoC with (a) permanent faults, (b) intermittent faults, (c) both permanent and intermittent faults

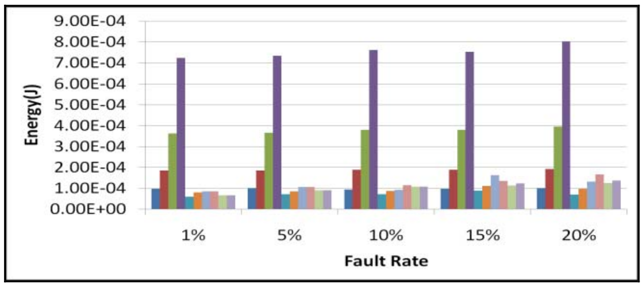
The energy consumption for the NL and OE schemes is comparable and higher than the XY scheme because they use non-minimal paths at times. But both NL and OE schemes offer higher successful packet arrival rate due to greater path diversity compared to the XY scheme, with the gap increasing especially under higher fault rates. For high fault rates, the XYX scheme has lower energy consumption than NS-FTR, OE+IOE, and even the single VC schemes (NL and OE) because it only uses minimal paths, much like the XY scheme. However, the XYX arrival rates are lower than that for NS-FTR and OE+IOE. Under low fault rate conditions (< 6%), both NS-FTR and OE+IOE disable replication, and therefore have lower energy consumption than XYX. In some cases, such a replication threshold based mechanism can be extremely useful to trade-off energy with packet arrival rate. For instance, for multimedia applications, occasionally dropped pixel data may not be perceivable by viewers, and in such scenarios the resulting lower energy consumption may lead to a much longer battery life and better overall user experience. Conversely, if fault resiliency is paramount, the value of δ can be reduced to close to 0, to enable the highest possible successful packet arrival rate.



(a)



(b)

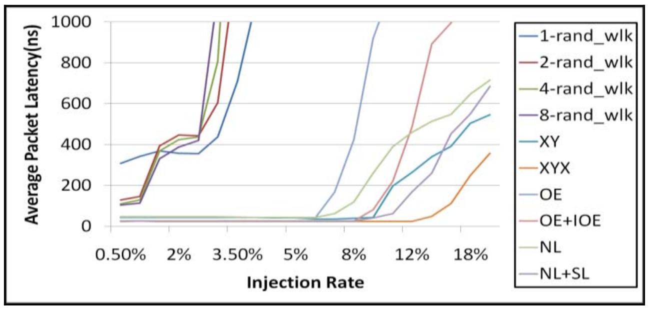


(c)

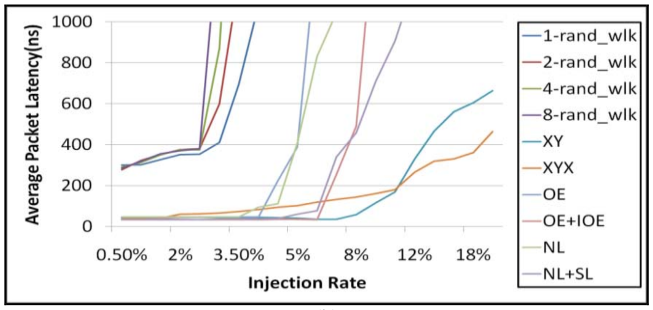
Fig. 6. Communication energy for 9×9 NoC with (a) permanent faults (b) intermittent faults, (c) both permanent and intermittent faults

Fig.7 (a)-(b) summarizes the average packet latency for the routing schemes for various injection rates, with a low fault rate (Fig. 7(a); 1% permanent faults and 1% intermittent faults) and a high fault rate (Fig. 7(b); 10% permanent faults and 10% intermittent faults). The N-random walk scheme has significantly higher packet latencies than other schemes due to its stochastic non-minimal path selection process. For low fault rates (Fig. 7(a)), low injection rates show comparable packet latencies for the rest of the schemes. However, as injection rates increase, the minimal path schemes (XY, XYX) have lower average latencies. But the latency figure becomes irrelevant for the XY and XYX schemes as their packet arrival rates suffer under high fault rates (Fig. 5). Among schemes that support adaptivity in path selection, the NS-FTR has the lowest packet latencies even for high injection rates.

For high fault rates (Fig. 7(b)), at low injection rates, both XY and XYX schemes surprisingly have higher packet latencies than turn model schemes such as NS-FTR, OE+IOE, OE, and NL. This is because while turn model based schemes such as NS_FTR or OE may not always use minimal paths (as XY and XYX schemes do), they have greater adaptivity to avoid faults and thus avoid retransmission, which can increase packet latencies directly and also indirectly (e.g., by increasing congestion). At higher injection rates, XY and XYX again have lower packet latencies, but suffer from lower packet arrival rates than the NS-FTR and OE+IOE schemes. Out of the two, NS-FTR again shows lower packet latencies than OE+IOE, even as injection rates increase.



(a)



(b)

Fig. 7. Average packet latency for 9×9 NoC with both permanent and intermittent faults (a) 2% fault rate, (b) 20% fault rate

To summarize, from the results it is quite apparent that the proposed NS-FTR scheme possesses high fault tolerance that scales well with rising (permanent and intermittent) fault rates and injection rates. Compared to the XYX scheme, NS-FTR has much higher successful packet arrival rates (>10%) especially when fault rates are high. Compared to the OE+IOE scheme, NS-FTR in general has lower implementation overhead, higher packet arrival rates, lower energy consumption, and lower packet latencies. These results strongly motivate the use of NS-FTR in future CMP designs to ensure fault tolerant on-chip communication.

## V. Summary and Conclusions

In this paper, we propose a novel fault tolerant routing scheme (NS-FTR) for NoCs that combines the North-last and South-last turn models to create a robust hybrid NoC routing scheme. The proposed scheme is shown to have a low implementation overhead and adapt to design time and runtime faults better than existing turn model, stochastic random walk, and dual virtual channel based routing schemes such as XYX and OE+IOE. Our ongoing work is developing a theoretical formulation to characterize the coverage and fault tolerance capabilities of various NoC routing schemes.

## References

[1] C. Constantinescu, “Trends and challenges in VLSI circuit reliability”, IEEE Micro, 23(4):14-19, July-Aug 2003.
[2] S. Nassif, “Modeling and analysis of manufacturing variations,” Proc. Custom Integrated Circuits Conference, May 2001.
[3] S. S. Mukherjee, J. Emer, S. K. Reinhardt, “The soft error problem: An architectural perspective,” Proc, HPCA, 2005.
[4] E. Normand, “Single event upset at ground level,” IEEE Transactions on Nuclear Science, 43 (6):2742–2750, Dec 1996.
[5] S. Pasricha and N. Dutt, On-Chip Communication Architectures, Morgan Kauffman, Apr 2008.
[6] C. Constantinescu, “Intermittent faults in VLSI circuits,” Proc. IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE), 2007.
[7] L. Benini and G.D. Micheli, “Networks on chips: a new SoC paradigm,” IEEE Computer, pp. 70-78, Jan. 2002.
[8] W. J. Dally, B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” Proc. DAC, pp. 684-689, 2001.
[9] D.C. Pham, et al., “Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor,” Proc. IEEE Journal of Solid-State Circuits, 41(1):179-196, 2006.
[10] Intel Teraflops, http://download.intel.com/research/platform/terascale/terascale_overview_paper.pdf
[11] Picochip PC102. http://www.picochip.com/highlights/pc102.
[12] S. Bell et al., “TILE64 processor: A 64-core SoC with mesh interconnect,” Proc. ISSCC, 2008.
[13] D. Bertozzi, L. Benini, and G. De Micheli, “Error Control Schemes for On-Chip Communication Links: The Energy–Reliability Tradeoff,” IEEE Trans. CAD, 24(6), pp. 818-831, 2005.
[14] S. Murali, et al., “Analysis of Error Recovery Schemes for Networks on Chips,” IEEE Design & Test of Computers, 22(5): 434-442, 2005.
[15] S. Lin and D. J. Costello, “Error Control Coding: Fundamentals and Applications,” Englewood Cliffs, NJ: Prentice-Hall, 1983.
[16] D. Bertozzi, L. Benini, and G. De Micheli, “Low power error resilient encoding for on-chip data buses,” Proc. DATE, pp. 102-109, 2002.
[17] M. Lajolo, “Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses,” TVLSI, 9(6), 2001
[18] W.J. Dally, B. Towles, “Principles and Practices of Interconnection Networks,” Morgan Kauffman, San Francisco, CA, 2004.
[19] M. Dehyadgari, et al., “Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC,” Proc. MICRO, 2005.
[20] H. Zhu, et al., “Performance evaluation of adaptive routing algorithms for achieving Fault Tolerance in NoC Fabrics,” Proc. ASAP 2007.
[21] T. Dumitras and R. Marculescu, “On-chip stochastic communication,” Proc. DATE, 2003.
[22] M. Pirretti, et al., “Fault Tolerant Algorithms for Network-On-Chip Interconnect,” Proc. ISVLSI, 2004.
[23] Y. B. Kim, Y.-B. Kim, “Fault Tolerant Source Routing for Network-on-chip,” Proc. DFT, 2007.
[24] T. Schonwald, et al., “Fully Adaptive Fault Tolerant Routing Algorithm for Network-on-Chip Architectures,” Proc. DSD, Aug. 2007.
[25] T.Schonwald, O.Bringmann, W.Rosenstiel, “Region-Based Routing Algorithm for Network-on-Chip Architectures,” Proc. Norchip 2007.
[26] C. J. Glass, L. M. Ni, “The turn model for adaptive routing,” Proc. ISCA, pp. 278 – 287, 1992.
[27] C. J. Glass and L. M. Ni, “Fault-tolerant wormhole routing in meshes without virtual channels,” IEEE Transactions of Parallel and Distributed Systems, Vol.7, pp. 620-635, No. 6, 1996.
[28] C. M. Cunningham, D. R. Avresky, “Fault-tolerant adaptive routing for two-dimensional meshes,” Proc. HPCA, 1995.
[29] G.-M. Chiu, “The Odd-Even Turn Model for Adaptive Routing,” IEEE TPDS, Vol.11, No.7, pp.729–738, 2000.
[30] A.Patooghy, S.G.Miremadi, “XYX: a power & performance efficient fault-tolerant routing algorithm for network on chip,” Proc. ICPDNP, 2009
[31] D.Fick, et al., “A Highly Resilient Routing Algorithm for Fault-Tolerant NoCs,” Proc. DATE 2009.
[32] R.V. Boppana and S. Chalasani, “Fault-tolerant wormhole routing algorithms for mesh networks,” IEEE Trans. on Computers, 44(7), 1995.
[33] Jie Wu, “A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model,” IEEE Transactions on Computers, Vol. 52, No. 9, pp. 1154-1169, Sept. 2003.
[34] A. Rezazadeh, et al., “If-cube3: An Improved Fault-Tolerant Routing Algorithm to achieve less latency in NoCs,” Proc. IACC, 2009.
[35] S.Jovanovic, et al., “A new deadlock-free fault-tolerant routing algorithm for NoC interconnections,” Proc. FPLA, pp. 326 – 331, 2009.
[36] M. Andres, et al., “Region-Based Routing: A Mechanism to Support Efficient Routing Algorithms in NoCs,” IEEE TVLSI, 17(3), 2009.
[37] J. Hu and R. Marculescu, “Dyad - smart routing for networks-on-chip,” Proc. DAC, 2004.
[38] Z. Zhang, A. Greiner, and S. Taktak, “A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip,” Proc. DAC, 2008.
[39] S. Pasricha, Y. Zou, D. Connors, H. J. Siegel, "OE+IOE: A Novel Turn Model Based Fault Tolerant Routing Scheme for Networks-on-Chip", Proc. CODES+ISSS, Oct 2010.
[40] W.-C. Kwon, S. Yoo, J. Um, S.-W. Jeong, “In-network reorder buffer to improve overall NoC performance while resolving the in-order requirement problem,” Proc. DATE, pp. 1058-1063, 2009.
[41] M. Koibuchi, et al., “A Lightweight Fault-Tolerant Mechanism for Network-on-Chip,” Proc. NOCS 2008.
[42] Nirgam simulator, http://nirgam.ecs.soton.ac.uk/.
[43] A. Kahng, et al, “ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration,” Proc. DATE, 2009.
[44] SystemC initiative, http://www.systemc.org.