Hermes: Architecting a Top-Performing Fault-Tolerant Routing Algorithm for Networks-on-Chips

Costas Iordanou[†], Vassos Soteriou[†], Konstantinos Aisopos[‡]
[†]Department of Electrical Eng., Computer Eng. and Informatics
Cyprus University of Technology
cc.iordanou@edu.cut.ac.cy, vassos.soteriou@cut.ac.cy
[‡] Microsoft Corporation
WA 98052, USA
kaisopos@microstoft.com

***Abstract*—Networks-on-Chips (NoCs) are experiencing escalating susceptibility to wear-out and reduced reliability, with the risk of becoming the key point of failure in an entire multicore chip. Aiming towards seamless NoC operation in the presence of faulty communication links, in this paper we propose Hermes, a highly-robust, distributed and lightweight fault-tolerant routing algorithm, whose performance degrades gracefully with increasing faulty link counts. Hermes is a deadlock-free hybrid routing algorithm, utilizing load-balancing routing on fault-free paths to sustain high-performance, while providing pre-reconfigured escape path selection in the vicinity of faults. Additionally, Hermes identifies non-communicating network partitions in scenarios where faulty links are topologically densely distributed. An extensive experimental evaluation, including utilizing traffic benchmarks gathered from full-system chip multi-processor simulations, shows that Hermes improves network throughput by up to $3\times$ when compared against prior-art.**

Keywords - Network-on-chip, chip multi-processor, fault-tolerance, routing algorithm, reliability

I. Introduction

Multicore chips such as Chip Multi-Processors (CMPs) [30] employ Networks-on-Chips (NoCs) as their interconnect architecture of choice to provide efficient on-chip communication. Unfortunately, at deep sub-micron scales on-chip components become increasingly unreliable and susceptible to permanent faults, with the International Technology Roadmap for Semiconductors (ITRS) [27] projecting a 10-fold increase in CMOS wear rate in a 10-year span. While device failure rates will keep increasing at future CMOS technologies, multi-billion transistor chips containing faulty components will be expected to operate transparently as being reliable and fault-free [8], [22].

In CMPs, transistors found in processing cores, cache memory, and on-chip network routers may equally fail permanently due to time-dependent physical wearout effects such as hot-carrier degradation and oxide breakdown [5], that may in turn quickly manifest to architectural-level failures. Individual core wear-out or failures in on-chip memory modules, however, may not be unavoidably disastrous to the CMP's full-system functionality as cores and memory cells are inherently redundant to a particular extent [25]. With some cores or memory cells failing, the multicore system may continue operation with a preserved correct functionality, albeit at a degraded performance mode, given that error detection and recovery techniques, such as fault-tolerant task migration, core- and cache-level error isolation and masking, and relevant OS support, are applied [22].

NoC routers, however, enjoy crucially less redundancy vs the rest of the multicore chip components. Even an isolated intra-router or communication link failure can turn a static regular topology into an irregular one with an unplanned geometry. Hence, either physical connectivity among network nodes may not exist at all due to the presence of faulty links and/or routers, that may even cause entire sub-network areas to detach from each other forming isolated partitions, and/or the associated routing protocol may not be able to advance packets to their destinations due to protocol-level breakdown. Traffic-induced back-pressure, then, causes accumulated congestion, possible traffic deadlocks, and even the entire multicore system to stall indefinitely rendering it inoperable. Hence an NoC failure, can become the entire multicore system's *single* fatal failure.

A. Hermes: Synopsis and Contributions

A key operational stress-induced wear mechanism in current and future CMOS technologies is electromigration (EM). EM causes material deformations and consequently the loss of connections in a circuit [4], assessed by the ITRS 2009 Interconnect Report [27] as the main cause of on-chip metal interconnect reliability loss. Enormous data transfer rates, as in the case of high speed/high-throughput metal wires in NoC links, cause thermal stressing which amplifies EM.

In an effort to establish inter-router communication resilience and hence to sustain seamless NoC operation in the presence of faulty links, we propose Hermes[1], a top-performing and highly robust distributed fault-tolerant routing algorithm that bypasses faulty network paths/regions/areas at a gracefully performance-degrading mode with increasing faulty link counts. Hermes employs a dual-strategy routing function that is agnostic of the current topological region's state: in the region(s) where only healthy links exist, by default, Hermes utilizes either XY Dimension-Order Routing (DOR) or load-balancing O1TURN routing [28] to sustain high throughput, while it provides pre-calculated escape path selection in the vicinity of faults, with routing information distributed in tables at each router. To achieve the latter, it employs up*/down* routing, a topology-discovering strategy that can be tailored to be deadlock-free, ideal for achieving fault-tolerance in NoCs. Further, works which propose intra-router redundancy-based wear resilience, such as in buffers [12] and router ports [14], equally pivotal to an NoC's operational robustness [22], are complementary and can be orthogonally applied to Hermes' scheme to further extend an NoC's operational lifetime.

To guarantee deadlock-freedom, no switching from up*/down* routing to either DOR or O1TURN routing is allowed, where the latter two are the default routing choices upon packet injection. With new faulty link(s) appearing in the topology, the up*/down*-based routes are no longer valid and new ones have to be reconfigured; the network is frozen, and using topology-scanning flags that spread synchronously from the new faulty point, analogous to the Ariadne re-configuration scheme's approach [1], Hermes discovers the latest inter-connectivity form and updates each table with relevant route information. Once completed, the NoC resumes operation until a new faulty link(s) appears in the topology, where the process is repeated.

[1]In Greek mythology Hermes was an Olympian God, who among many roles, he was protector and patron of travelers.

Hermes is geared towards 2D mesh-interconnected NoCs, and while its up*/down* faulty path discovering/bypassing reconfiguration process is based on that of Ariadne's elegant approach [1], it encompasses a number of extensions and distinct contributions:

1) Its *two synergistic routing functions* achieve significantly higher and linearly-degrading performance with increasing faulty link counts (Section IV) vs prior-art [1], [23].
2) It illustrates that *Virtual Channel (VC) classification* with regard to the underlying state of the routing path, i.e., encountering healthy vs faulty links, offers far-improved performance than in non-classified use of VCs, i.e., VCs being used without restraint by a single routing scheme whether traversing fault-free or faulty paths (Section IV).
3) It is a *purely hardware-based approach* and does not employ non-deterministic execution time iterative software kernels to discover fault-free routes, during which the entire multicore chip is temporarily inoperable [23].
4) Hermes *identifies network segmentations* using network-spreading topology-scanning flags. The CMP operating system may then utilize this information to mark sub-network borders, enabling the assignment of independent processes and threads to each network partition [8] (Section II-D).

B. Hermes' Major Features and Attributes

Hermes meets all design *challenges* and satisfies the *ideal objectives* of a well-designed fault-tolerant routing protocol [32]. First, Hermes establishes *fault-tolerance* bypassing a *high number of faulty links* that can form any topological fault region with no faulty link spatial placement restrictions and healthy link victimization to achieve deadlock-freedom vs works in [6], [10], [32]. Given a *realistic* minimally-connected path scenario, Hermes maintains *feasibility* in packet delivery given that *any* physical connectivity exits (Section II). Hence, Hermes *adapts* to the state of the topology, where any spatial permutation of healthy and faulty links can exist. Second, it is *deadlock-free*, where no packets can be involved in a deadlocked situation which can halt the flow of packets and stall the CMP's operation indefinitely (Section II-E), while establishing *short routes*, devoid of *livelocks*. Third, Hermes is *distributed*, where each node individually directs its packets towards their next-hop router, with no global information maintenance concerning the number and spatial distribution of faulty links (Section II). Fourth, it supports *load-balancing* in fault-free regions so as to maintain improved performance, detailed in Section II. Next, Hermes is *lightweight*, demanding *reasonable area and power overheads*, while keeping the base pipelined wormhole router's *critical path unaffected* (Section IV-D), and finally, Hermes can handle *dynamically-occurring (run-time) faults*. Although in most previous works some of the aforementioned objectives can conflict each other [6], [10], [32], to our best knowledge Hermes may be the *first* FT routing algorithm for NoCs to *satisfy all of them*.

Next, Section II details Hermes' routing algorithm and sub-network detection mechanism, while Section III presents its micro-architecture. Following, Section IV evaluates Hermes, while Section V presents related work. Finally, Section VI concludes this paper.

II. Hermes Routing Algorithm

With any NoC link becoming faulty, Hermes' reconfiguration process is initiated to discover and mark faulty links, then update routing tables distributed at each router so that they can steer traffic towards their destinations in an irregular interconnect geometry. Following Ariadne's reconfiguration scheme [1], route discovery depends on *atomic* flag broadcasts conforming to up*/down* routing rules, originally utilized in topology-irregular networks of workstations [16], [26], which we adopt to 2D meshes common in silicon-planar NoCs [30]. Up*/down* was chosen as it is topology-agnostic and ensures acyclic path formations devoid of deadlocks. Once the tables are updated, with the entire path discovery process guaranteed to complete in N^2 cycles in an N-node network, NoC operation resumes until a new faulty link(s) appears. Any faults in a router's datapath which can block access to links even when being healthy, including upstream and downstream buffers, control logic, and intra-router crossbar connections, are regarded as an extension to the link(s) connected to them, inevitably designating that link(s) as also being non-healthy and unusable [23]. Hermes' reconfiguration process is specifically geared for 2D mesh-based NoCs, where up*/down* optimizations were appropriately inherited from tactics found in [26].

A. Routing in a Non-Faulty Topology Region

Hermes combines two network routing strategies. In a completely fault-free topology, or fault-free region encountered at packet injection, Hermes uses either DOR-XY, dubbed as *H-XY*, or partially adaptive O1TURN [28] routing, named *H-O1T*. Under H-O1T, upon network ingress, a packet is granted a 50% probability of utilizing either XY or YX routing as a means of balancing network load. H-XY uses 2 Virtual Channels (VCs) at minimum, while H-O1T uses 3 VCs at minimum. Either of these routing schemes is followed until a faulty link is encountered, where routing switches to pre-configured up*/down* rules *exclusively* until network traffic egress.

Hermes utilizes two variants of up*/down* routing: bidirectional up*/down* routing as used in Ariadne [1], where even when one of the two unidirectional links in a link-pair is faulty then both are considered faulty, and the upgraded up*/down* scheme of uDIREC [23] that marks faulty links unidirectionally in a link-pair, hence incurring milder healthy link victimization. The uDIREC-based Hermes variants are equivalently dubbed *H-uXY* and *H-uO1T*. However, uDIREC requires iterative software kernels to form unidirectional up*/down* paths, and since Hermes is purely hardware-based, following we focus on utilizing Ariadne's up*/down* scheme [1]; H-uXY and H-uO1T are solely employed in our performance tests in Section IV.

B. Routing in a Faulty Topology Region: Reconfiguration Algorithm

A packet switches to up*/down* routing (from XY or O1TURN) and occupies its associated VC, *only when* it encounters a faulty link in its path, and keeps following its rules until its ejection to ensure deadlock-freedom (Section II-E). The reconfiguration process begins upon the detection of a new faulty link(s), at which point the router to which this link is connected becomes the root node (initiator). It begins broadcasting Direction-Recoding Flags (*DRF*) to all of its output ports connected to a healthy link that lead to associated next-hop routers to *discover the topology's connectivity*. Flags are spread using a *2-bit overlay control network*, which sits atop the data network, to ensure all-to-all node flag transmission/reception. The concept is that, upon reception of these DRF flags, the next-hop neighboring routers record the healthy input port through which the local DRF had arrived into their routing tables so as to designate the direction (port) that leads them back to the root (broadcasting) node. Simultaneously (where applicable), Hermes broadcasts Alert Flags (*AF*) using the same control network to *discover network partitioning*, detailed next. Once a root node completes reconfiguration, the remaining $N-1$ nodes individually become root nodes, so that all nodes will eventually discover how to reach each other. Each node carries the

following five-stage process (Fig. 1), requiring control information bookkeeping and handling of reconfiguration orchestration among routers (stage actions 1 and 5, and AF flags are unique to Hermes, while stage actions 2-4 are inherited from Ariadne [1]):

- **Action 1. Flag/sub-network detection:** Identifies whether a node receives a DRF or an AF flag.
- **Action 2. Entering Recovery:** A network node enters into recovery mode, invalidates its existing routing tables, freezes its pipeline and stops injecting traffic into the network.
- **Action 3. Tagging Link Directions:** Up*/down* marks all adjacent links of a node as either "up" or "down."
- **Action 4. Routing Table Update:** Information regarding which port can be used to reach the current broadcasting node is recorded into the current DRF-receiving node's routing table.
- **Action 5. Flag Forwarding:** Forwards the reconfiguration flag according to the link status between each node-pair (healthy=DRF flag, faulty=AF flag). When a node is in recovering mode the AF flag is always ignored since the reconfiguration process has already begun.

When the root node finishes broadcasting, the remaining $N-1$ nodes broadcast individually. All five actions are carried out during the first broadcast, while only actions 4 and 5 are carried out during the remaining $N-1$ broadcasts. Action details follow next.

Flag/Sub-Network Detection - Action 1: The 1-bit DRF and 1-bit AF flags are broadcasted starting from the root; DRF flags are propagated *only atop healthy links*, while AF flags are transmitted *only atop faulty links* connecting two routers via the 2-bit overlay network (assumed to be always functional with triple modular redundancy). The use of DRF flags is to discover connected paths among all source-destination router pairs, while the AF flags mark possible boundaries of network partitions/segmentations. The essence here is that AF flags always follow *minimal* paths, while DRF flags may follow *non-minimal* paths due to the presence of faulty links; hence AF flags may arrive earlier at a router, alerting the router that it may possibly belong to a physically disconnected network segment. If a router eventually receives a DRF flag from the root, while it had received an AF flag earlier, this ensures that a physical path does exist starting from the root, and the AF flag is ignored canceling the previously set network segmentation alert. Otherwise, the node remains in its network segmentation alert mode (Section II-D).

Entering Recovery - Action 2: Upon DRF flag reception, propagated from the current root node, the receiving node invalidates its routing table, freezes flit injection, stops its pipeline, and enters recovery. The state is set back to "normal" after N^2 cycles when the reconfiguration process is completed. Each subsequent DRF flag reception will only invoke Actions 4 and 5. However, in case the router had received an AF flag and is in its alert state and not in recovery mode, by the end of the current N cycles it will be designated that this node belongs to a network partition. Since all nodes in all possible partitions will eventually broadcast, all boundaries among all partitions will eventually be discovered (Section II-D).

Tagging Link Directions - Action 3: During this step, with the use of topology-propagating DRF flags, ports at a router are marked as either "up" or "down." This ensures that all source-destination router pairs will eventually mark their routing tables with next-hop routes to be used to reach each other. The walkthrough example in Section II-D shows how DRF flags are propagated according to pre-specified rules which in addition maintain deadlock-free routing (Section II-E).

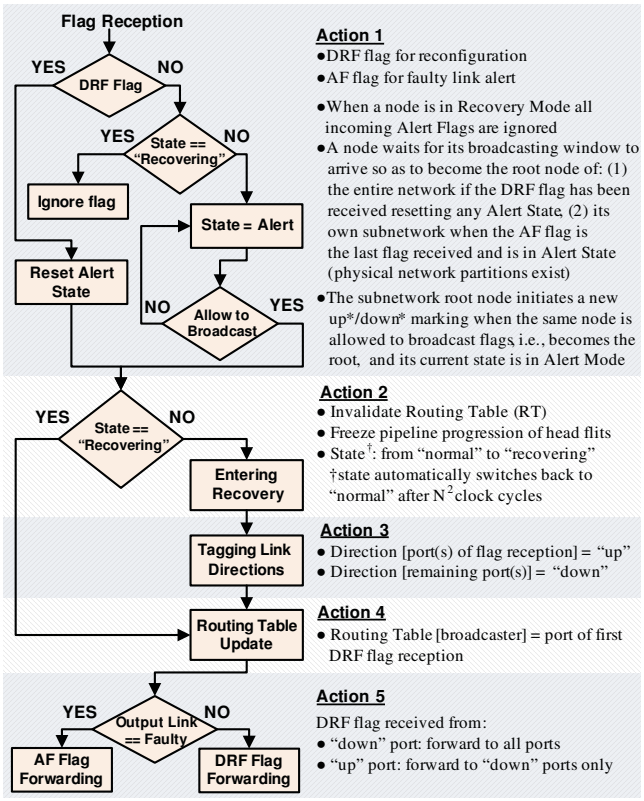

Fig. 1. Hermes reconfiguration algorithm.

Routing Table Update - Action 4: During each broadcast, a root node basically informs how it can be reached from all other nodes via the spreading of DRF flags that follows legal up*/down-based turns (see Action 5); these DRF flag broadcasting patterns trail relevant paths starting from this root that are recorded, in a mirror-reflected mode, into each nodes's routing table. Any node other than the root then uses these pre-recorded table paths to reach that root node.

Flag Forwarding - Action 5: After the routing table of an intermediate node is updated, the same node broadcasts its DRF flag through those ports (via the 2-bit overlay control network) where the associated link is healthy, that it had not received a DRF flag from during the previous cycle, and it broadcasts an AF flag through the port(s) where the associated link is faulty. In addition, the up*/down* turn restrictions are also taken into consideration, that is, a DRF flag received from an "up" link is never sent to an "up" link; only "up" to "down" and "down" to "down" broadcasts are legal. In case where the link is faulty, no turn restrictions are applied, and the AF flag is sent atop the faulty link using the 2-bit overlay control network to the neighboring node. If the receiving node of the AF flag is already in its recovering state then the AF flag is ignored (Fig. 1).

C. Timing and Synchronization

Single-cycle flag forwarding among node pairs ensures that the broadcasting window of each node *deterministically* takes N clock cycles to complete, in an N-node topology. This considers the worst-case, but unrealistic, scenario of having a *minimum spanning tree*-like faulty topology, where N cycles are needed for a node's complete broadcast. Since each node broadcasts in a non time-overlapping *atomic* mode, reconfiguration takes N^2 cycles to complete.

A broadcaster can be easily identified by all nodes, exactly because only this node is expected to broadcast during a specific time

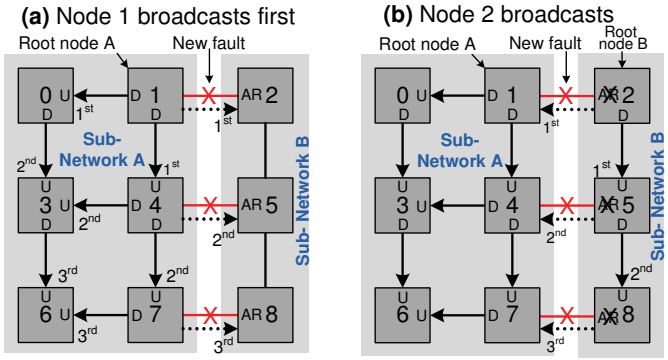

Fig. 2. A walkthrough of up*/down* marking and the sub-network detection mechanism using a 3×3 mesh consisting of subnetworks A and B. AR denotes "Alert Register" is on, red lines with an "X" are broken (i.e., faulty) links, solid and dotted lines show the broadcasting of the DRF and AF flags respectively at specified broadcast cycle times.

window, based on the system's global clock which acts a common reference point. Hence, no ID of any node is forwarded since the current root broadcaster is known to all nodes. The Node ID Extractor (Section III), a piece of hardware that resides in all nodes, identifies the current broadcasting node. *Atomic* broadcasts [1], where only one node is allowed to broadcast based on a simple formula which correlates the global clock to a node's unique ID, are used. This mathematical formula states that the first $log_2(N)$ LSBs of the global clock designate the broadcasting cycle of each node, while the next $log_2(N)$ higher bits are used to identify the broadcasting node; the latter ensures exactly N repetitions (i.e., one for each router node) using modulo arithmetic as $node((ID) mod(N))$ for node with identity ID, then $node((ID+1) mod(N))$ for node with identity $(ID+1)$ and so on until all N become non time-overlapping roots and broadcast during their assigned time slot. Assuming an 8×8 network with 64 nodes, as in our experimental evaluation of Section IV, the first $log_2(64) = 6$ LSBs and the next higher $log_2(64) = 6$ bits of the global clock are used to identify the broadcasting cycle of each node, and the broadcasting node's ID, respectively. In case multiple faults occur, one of the nodes which detected a fault will become root first, which is arbitrarily based on the received modulo-based slot; eventually all nodes become roots and broadcast their flags. The direction of the received DRF flags at a router, each designates the cardinal direction at the respective port that they are received, with the information marked into the routing table as 4-bit one-hot records (Section III).

D. Up/Down* Marking and Sub-Network Discovery Walkthrough*

Fig. 2 walks through up*/down* marking and our Sub-Network Detection Mechanism (SNDM) that can detect individual physical network-dividing non-communicating partitions, using a 3×3 mesh. The former is served by the network-spreading single-bit DRF flags (Direction Recording Flag), and the latter by the single-bit AF flags (Alert Flag), both which span the network on a cycle-by-cycle basis using the 2-bit overlay control network (Section II-B).

Fig. 2 emphasizes the usage of AF flags used by SNDM as they are a unique feature of Hermes, that traverse the topology along with DRF flags as in Ariadne's scheme [1]. In our example, links $4 \leftrightarrow 5$ and $7 \leftrightarrow 8$ are initially broken, while link $1 \leftrightarrow 2$ currently breaks down, partitioning the topology into sub-networks A and B. Instantly, node 1 acts as the root, initiates the reconfiguration process, enters recovery mode, and invalidates its current routing table entries. The broadcast cycle times of DRF and AF flags are shown at the side of each node, which begin spreading towards the vicinity of node 1 (Section II-B). The purpose of using the AF flags is to set a virtual border at the links residing at the physical edges of sub-networks, so that the topological partitions can be marked and hence identified.

As Fig. 2-(a) shows, node 1 marks its output ports connected to links $1 \rightarrow 0$ and $1 \rightarrow 4$ as "down" (D), and sends DFR flags to nodes 0 and 4. This sets their Status Register (SR) in recovering state (Section III), where their respective input ports are set as "up" (U), with these port directions recorded into their respective routing tables. Nodes 0 and 3 are aware that node 1 is the root as the node ID is extracted from the global clock based on modulo arithmetic (Section II-C), and hence all nodes are synced (Node ID Extractor of Section III). Node 2 only receives an AF flag, via the control network, since link $1 \leftrightarrow 2$ is now faulty, setting its Alert Register (AR) in alert state. Following up*/down* rules, in cycle 2 node 0 marks link $0 \rightarrow 3$ as D and U, and node 4 marks links $4 \rightarrow 3$ and $4 \rightarrow 7$ both as D and U, all at their two respective ends. Nodes 3 and 7 are set in recovery state, while node 4 sends an AF flag to node 5 atop $4 \leftrightarrow 5$ faulty link, via the 2-bit control network, setting node 5 in alert state. Finally, in cycle 3 the north and east ports of node 6 are marked U, setting node 6 in recovery state, while node 8 receives an AF flag from node 7 atop $7 \leftrightarrow 8$ faulty link setting node 8 in alert state. Now subnetwork A has completed both its up*/down* marking for all of its member nodes and recording of all paths which lead towards root node 1. Note that AF flags obviously cause no up*/down* marking at the receiving nodes, since they merely designate the possibility of a network disconnection at the receiver node, and that no U $\rightarrow$ U marking is allowed to ensure acyclic behavior [1], [16], [23], [26]. Also, no node that receives a DRF or AF flag from one of its ports can later send any other flag using that same port to avoid erroneous port marking synchronization; flag reception time is tied to the global clock dictated by the Node ID Extractor (Section III).

With $N = 9$ cycles elapsed, where N is the node count, node 2 broadcasts next, designated by the global clock utilized by the Node ID Extractor (Section II-C), initiating a new up*/down* marking chain of events, shown in Fig 2-(b). The reception of an AF flag and no DRF flag by node 2 from the previous broadcasting cycle, points towards its separate sub-network presence. Node 2 broadcasts its DRF and AF flags accordingly, following the flag-based up*/down* marking rules. The south port of node 2 is set as D while the north port of node 5 is set to U in cycle 1, and so on. However, when nodes 1, 4 and 7 receive an AF flag from corresponding nodes 2, 5 and 8 residing in sub-network B, they ignore this flag since they are already in recovering mode. Nodes 3, 4, 5, 6, 7, 8 and 0 will broadcast in series as roots when their broadcasting turn arrives, as dictated by the global clock, where each one informs its sub-network-residing nodes how they can be reached by broadcasting DRF flags.

The process completes in N^2 cycles with node 0 ending its broadcast. By then, *only valid escape paths* are recorded into the routing table of each router, reflecting upon routes that *are valid only within the same sub-network*, i.e., either sub-network A or sub-network B. This information may subsequently be provided to the operating system to mark the borders of these sub-networks, enabling the assignment of independent processes and threads to each such sub-network, being serviced by a sophisticated algorithm such as [8].

E. Hermes Deadlock Freedom

The DOR-XY or O1TURN [28] routing schemes used in Hermes are derived form the Turn Model [18] which ensures that channel

dependencies do not form loops, guaranteeing deadlock-freedom [10]. In up*/down* routing, the unique node order assignments to each router, where all increasing-order turns (down links) are disabled, followed by decreasing-order turns (up links), or vice-versa, ensure a unique visiting order where this decreasing to increasing node visiting order ensures the absence of routing cycles. Hence, the up*/down* routing function is acyclic and by definition deadlock-free, as Silla and Duato elaborately prove in [29]. Further, Aisopos et al. [1] prove that up*/down* is also deadlock-free in a faulty NoC environment.

The essence is to prove that the co-existence of DOR-XY and up*/down*, and O1TURN (DOR-XY + DOR-YX) and up*/down*, under H-XY and H-O1T respectively, are also deadlock-free. XY and up*/down* each require a single VC to be deadlock-free, while O1TURN requires 2 VCs, one for XY and the second for YX [28]. In both H-XY and H-O1T, a packet initially routes using a VC dedicated to the DOR function(s): XY (VC0) in the former, and either XY (VC0) or YX (VC1) in the latter. In both H-XY and H-O1T, a packet only switches to up*/down* (VC1 in H-XY and VC2 in H-O1T) when it encounters a faulty link in its path, and never switches back to either XY or YX. Hence, cyclic dependencies among VC0 and VC1 under H-XY, and between VC0 to VC2 and between VC1 to VC2 in H-O1T, are avoided, guaranteeing that H-XY and H-O1T are devoid of routing loops, i.e., acyclic, and by definition deadlock-free.

III. Hermes Micro-Architecture

Hermes uses a 4-stage pipelined wormhole NoC router with input Virtual Channels (VCs), comprising (1) routing computation, (2) VC arbitration, (3) switch allocation, (4) crossbar traversal and 1-cycle physical link traversal, along with some additional circuitry required for Hermes' operation as the upper part of Fig. 3 shows[2].

Hermes' Routing Logic unit shown in Fig. 3 is divided into three blocks, comprising up*/down*, DOR-XY and DOR-YX routing, each utilizing its own independent VC. H-XY utilizes the first two blocks, while H-O1T utilizes all three blocks where XY or YX routing are each used with equal probability when routing in a fault-free region. The up*/down* block is directly connected to Hermes' logic unit which provides the routing information, contained in the routing tables, and is utilized when the current packet being routed has encountered a faulty link in its progressive path. These routing tables are updated every time a new faulty link(s) appears in the network, dictated by said reconfiguration process (Section II-B). The VC Allocator is responsible in assigning a packet to the VC governed by up*/down* when a packet is being routed in a faulty region; otherwise XY's VC is assigned under H-XY, or either of XY or YX VCs are assigned in H-O1T, when routing in a healthy region.

Fig. 3 also shows Hermes' remaining logic blocks: the 1-bit Status Register (SR) with "normal" and "recovering" states, the 1-bit Alert Register (AR) with 'normal" and "alert" states, four 1-bit registers to designate "up" or "down" routing (one for each port) when up*/down* is in use, the logic to update the SR and AR registers accordingly (Action 3), to fill the routing tables (Action 4), and to forward the DRF or AF flags (Action 5). In a router there exist sixteen 1-bit width wires used for flag reception and forwarding, shown as wires $flag_{0-3}(in)$ and $flag_{0-3}(out)$ for flag in and out signals, respectively. Each $flag(in)$ and $flag(out)$ pair at each port consists of two 1-bit wires, each for DRF and AF flag forwarding.

[2]Hermes is orthogonal to other existing pipelined NoC router architectures such as the speculative wormhole architecture in [24].

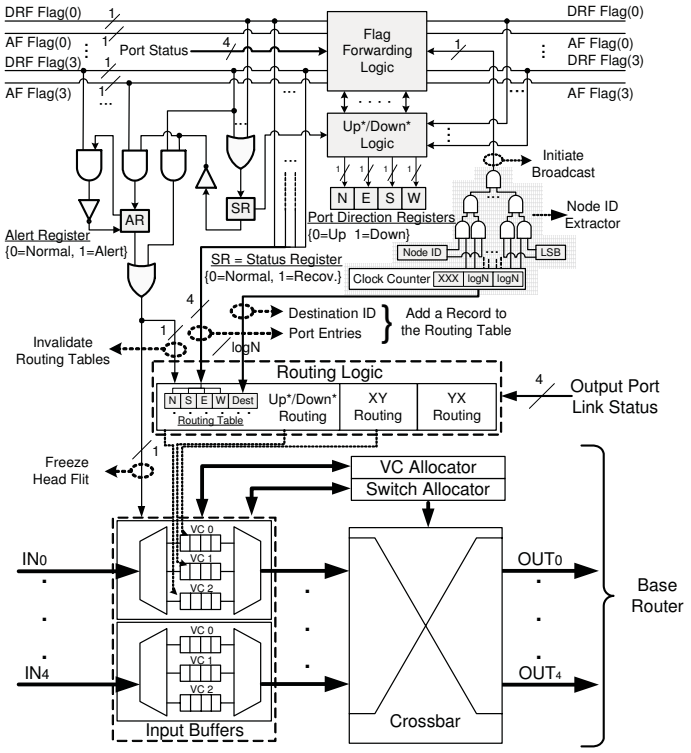

Fig. 3. Schematic of an input-buffered wormhole flow-control router architecture with virtual channels incorporating Hermes components.

The SR holds the current state of the network, where "recovering" implies that the network is currently stalled to reconfigure the routing tables, and where "normal" implies that the network is operating normally or has resumed into normal operation after a recovery from fault(s) and completed table reconfiguration. Under the normal state, packets can be injected into the network since valid routing paths exist in the routing table of each NoC router. The AR is updated as follows: if a new AF flag is received and the SR is in the "normal" state, then the AR register is set to the "alert state." If an AF flag is received under a current recovering state, then the AF flag is ignored. If the AF flag had already been received and the AR register is currently in the "alert" mode, then if a DRF flag is also received, the AR is reset to the "normal" state (Section II-B and Section II-D).

When the SR is in recovering state, the signal which freezes head flits and invalidates the routing tables is set to "on." This informs the up*/down* routing table block to discard the information currently held in it. The input buffers and injection ports are also stopped from forwarding/injecting packets into the network. During this recovering state, and the initial DRF flag reception at each node in the network, the up*/down* logic block is utilized by the root node and the subsequent nodes to mark their output and input ports as either "up" or "down" according to the rules detailed in Section II-D. The flag forwarding logic and output port status signals co-operate to decide when and what kind of flag (DRF or AF) needs to be forwarded to each direction according to (1) the up*/down* logic values and marking scheme (Section II-D), and (2) link statuses (Section II-B).

The four input 1-bit flag wire signals each direct the up*/down* marking in their respective cardinal direction (N, S, E, or W). The output port status identifies the state of health of each output port; in case the port is non-healthy, i.e., faulty, then the AF flag is forwarded to the next neighboring node, otherwise the DRF flag is forwarded (Section II-D). The DRF flags are received by the routing table filling

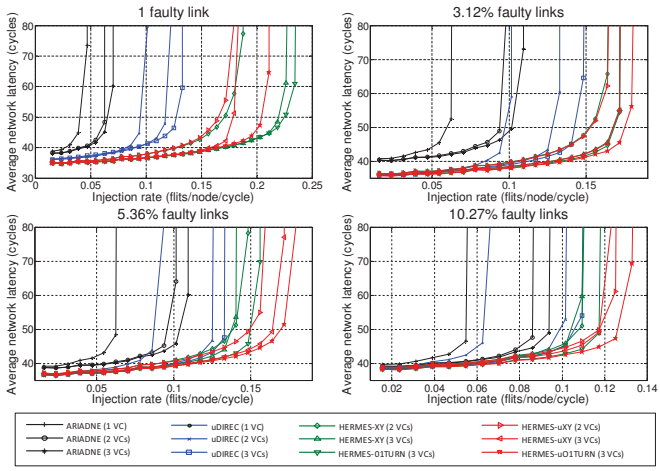

Fig. 4. Latency-throughput curves using a fully random faulty link placement scenario under synthetic uniform random traffic.

logic, and along with the extraction of the broadcasting node's ID and the direction of the input port(s) from where the DRF flag(s) was received, it updates the corresponding entry in its routing table using one-hot encoding to designate the cardinal direction. The table consists of N entries, 4 bits each. Once a table entry is recorded, it cannot be re-written at a later cycle; a table entry can be re-recorded only when a new reconfiguration process is initiated. Finally, the hardware which identifies the broadcasting node uses a set of $2log_2(N)$ bits, where N is the number of nodes in the NoC, with the N first LSBs used to compare to the global clock to designate the broadcasting cycle of each node, while the next $log_2(N)$ higher bits are used to identify the flag-broadcasting node (Section II-C).

IV. Experimental Setup and Results

To evaluate Hermes' performance we implemented a detailed cycle-accurate simulator that supports 2D meshes with four-stage pipelined routers (Fig. 3), each with 1 to 3 VCs per input port each consisting of 6-flit buffers. Our framework utilizes (a) synthetic Uniform Random (UR) traffic where all nodes have an equal probability of sending and receiving a packet per unit time, (b) Transpose (TR) traffic where packet source and destination coordinates matrix-alternate, an adversarial NoC-stressing form of traffic, and (c) real application workloads gathered from multi-threaded application execution in a full-system simulation environment, specifically from the Netrace benchmark suite, with their packets maintaining dependencies among them, tracked by our simulator for accuracy and fidelity. Synthetic traces, with six-flit 128-bit packets, are run for a million clock cycles, while in Netrace applications results were gathered within a 150-million cycle "region of interest." All experiments utilize an 8×8 mesh-interconnected NoC.

All link faults are topologically assigned in a purely random manner, but maintaining full topology connectivity, with 50 experiments repeated at each point to even-out the idiosyncrasy of each individual spatial faulty link placement, with results averaged. We compare against Ariadne [1] and uDIREC [23] which also utilize up*/down* routing. Further, no packets are segmented across a chain of routers in a path at the time of fault occurrence, as no retransmission mechanism, which can be orthogonally added [6], [10], is present.

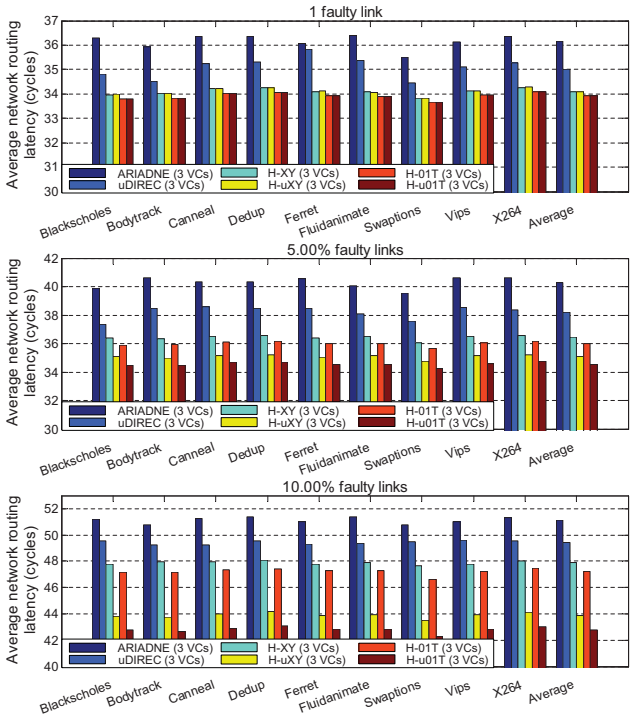

Fig. 5. Network routing latency using the Netrace benchmarks.

A. Results With Synthetic Uniform Random Traffic

Latency-throughput results under UR traffic are presented in Fig. 4, using 1 to 3 VCs per port for all considered routing schemes, with four faulty link intensities. Under all fault scenarios, and under any VC count, all Hermes variants consistently outperform Ariadne due to its heavy victimization of healthy bidirectional links in satisfying the up*/down* routing rules in pursue of deadlock-freedom.

With up to 5.36% faulty links all 2-VC Hermes variants outperform uDIREC with 3 VCs. This shows that *VC classification* in terms of intended usage, i.e., encountering faulty (up*/down*) vs non-faulty (XY or O1TURN) links in a routing path, is *superior to the plain use of VCs being used without restraint by a single routing scheme (up*/down*) irrespective of the health status of links*. Under 10.27% faulty links, H-XY (2 VCs) has the same performance as uDIREC with 3 VCs. Evidently, under UR traffic, the performance of all four Hermes variants degrades gracefully as faulty link counts increase. Additionally, with just one faulty link, the negative performance impact with Hermes is limited, as opposed to the cases of using Ariadne or uDIREC. Indicatively, H-XY and H-uXY outperform Ariadne and uDIREC with 2 VCs, and Ariadne and uDIREC with 3 VCs, by 28.2% and 23.9%, and 16.7% and 14.3%, respectively.

B. Realistic Full-System Workload Results

Realistic workload traces were captured from an 8×8 mesh-interconnected CMP running all the multithreaded PARSEC v2.1 suite benchmarks [3] executed onto the M5 simulator [2]. The Netrace infrastructure [20] was used to track and relay dependencies among packetized messages as expressed in a true CMP system. Each of the 64 tiles contains an in-order Alpha core at 2 GHz, each containing separate 32 KB 4-way set associative L1 I&D caches with 3-cycle access latency with coherency maintained via a MESI protocol, and a 16 MB L2 shared 8-way set associative 64-bank fully-shared S-NUCA with 64 B lines with an 8-cycle access time. Packets consist of 64 and 576 bits for miss request/coherence traffic and cache

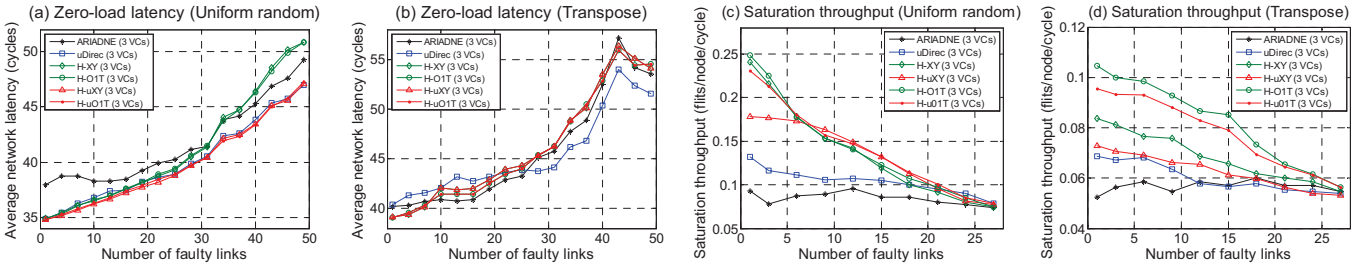

Fig. 6. Zero-Load latency under (a) Uniform Random (UR) and (b) Transpose (TR) traffic, and saturation throughput under (c) UR and (d) TR.

line transfers, respectively. All FT mechanisms use 3 VCs per input port, run under three random faulty link NoC topology placement scenarios: one faulty link, 5.0% and 10.0% NoC faulty links.

Fig. 5 presents average routing latency results for all Netrace benchmarks. Ariadne and uDIREC are increasingly outperformed by Hermes' four variants with increasing faulty link counts, proving Hermes' higher sustainable performance and robustness. The incorporation of the uDIREC scheme in hybrid H-uXY and H-uO1T shows performance improvements against H-XY and H-O1T respectively, due to the better utilization of unidirectional links and smaller healthy link victimization, with this gap widening under the severe case of 10% faulty links. Indicatively, under 5% faulty links, on average H-XY outperforms Ariadne and uDIREC by 9.60% and 4.71% respectively, H-O1T outperforms Ariadne and uDIREC by 10.66% and 5.83% respectively, H-uXY outperforms H-XY and H-O1T by 3.69% and 2.55% respectively, while H-uO1T outperforms H-XY and H-O1T by 5.12% and 3.99% respectively.

C. Zero-Load Latency and Saturation Throughput

An injection rate of 0.01 flits/node/cycle is used to emulate zero-load, where 3 VCs per port are used in all tests. Fig. 6-(a) shows that under the use of UR traffic H-uXY and H-O1T perform identically to uDIREC, as the opportunities for route optimization with O1TURN routing are non-existent at zero load. All these three, however, outperform Ariadne, H-XY, and H-O1T, as the last three victimize a greater number of healthy links due to their bidirectional use of links vs unidirectional under uDIREC. With one faulty link Ariadne is outperformed by 9.0% by all the other protocols. Under TR traffic, in Fig. 6-(b), interestingly at mid to high fault counts, uDIREC outperforms the remaining schemes due to less switching among VCs, when encountering faults, versus all Hermes' variants. With 43 faulty links the performance of all algorithms begins to improve due to a more optimal selection of shortest paths derived from the up*/down* scheme, despite having fewer healthy links available.

Saturation throughput is the point where the average NoC latency is $3\times$ its zero-load latency. All tests utilize 3 VCs per port. Under UR and TR traffic patterns of Fig. 6-(c) and Fig. 6-(d), respectively, Ariadne performs almost steadily, albeit at lowest sustainable throughput vs all other schemes, due to the complete dominance of up*/down* routing which victimizes some links as being bidirectionally faulty. uDIREC is the second worst-performing scheme across both UR and TR traffic patterns, and performs slightly better than Ariadne due to its milder link victimization under unidirectional up*/down* routing. Interestingly, and consistently, under TR traffic, H-XY outperforms H-uXY, while H-O1T outperforms H-uO1T, as under uDIREC the relatively heavier reliance on up*/down* routing, as fewer links are victimized, creates higher congestion than alternatively using XY or O1TURN routing. Overall, under both UR and TR, the four Hermes variant schemes are performance-superior to Ariadne and uDIREC.

D. Hardware Synthesis Results

We implemented and synthesized *all the Hermes hardware blocks* illustrated in Fig. 3 and described in Section III using the Synopsys Design Compiler, targeting a commercial 45 nm CMOS technology library at 1 V. We consider three-stage speculative pipelined virtual-channel routers [24] with credit-based wormhole flow-control with a 64-entry SRAM-based routing table, 128-bit links, 6-flit buffers, 2 GHz clock rate, at 50% switching activity, all for an 8×8 mesh topology. The 2-bit overlay network described in Section II and Section III possesses Triple Modular Redundancy (TMR). Ariadne [1] was also implemented, however the uDIREC reconfiguration scheme [23] is not compared against Hermes as it is heavily dependent on software strategies to form routing paths; in addition, the authors do not specify the implementation of the required end-to-end ECC blocks, and only report area overheads. It is, however expected, that once the reconfiguration hardware, the routing tables and ECC blocks are taken into account, that uDIREC will surpass Hermes' overheads.

Table I reports power-area overheads for the base speculative NoC router, as well as Ariadne, H-XY and H-O1T with 1 up to 3 VCs per input port. H-XY (2 VCs per port) presents 5.31% area and 9.81% power overheads respectively as compared to the base 2 VC per-port NoC router, while the corresponding comparisons against Ariadne with 2 VCs are 0.94% and 4.95%. H-O1T (3 VCs per port) presents 4.47% area and 5.91% power overheads respectively as compared to the base 3 VC per-port NoC router, while the analogous comparisons against Ariadne with 3 VCs are 1.51% and 0.47%.

V. Background and Related Work

Fault-Tolerant (FT) approaches applicable to NoCs have been inspired from macro-level Interconnection Networks (INs), where [6], [10] provide a comprehensive coverage. In both domains the scope of all FT approaches is analogous: to sustain seamless communication among all interconnected entities in the presence of *faulty links*, *nodes*, or *regions*. Duato's landmark work [9] is conducive in developing a theory for FT routing in INs, and consequently NoCs. Basically, as long as FT routing provides full connectivity devoid of cyclic channel dependencies in a sub-connected (faulty) topology, then the FT function guarantees deadlock- and livelock-freedom.

Most FT approaches are categorized as: (1) FT routing algorithms that bypass link/router failures, (2) logical/architectural redundancy within the routers to improve architectural resilience, and, (3) hybrid approaches that combine (1) and (2). Next, category (1) is further sub-classified as: (a) FT routing with bounded fault counts, (b) FT routing with unbounded faulty link counts but with spatial placement/pattern limitations, and, (c) unbounded faulty link counts and no spatial placement restrictions. FT schemes in the latter category are the most flexible, however the hardest to devise, in which Hermes belongs to.

The restrictive approaches outlined in categories (1-a) and (1-b)

TABLE I. OVERHEADS OF VARIOUS SYNTHESIZED NOC ROUTERS.

Router Virtual Channel Per-Port Count	Router Architecture (128bits)	Area (gates, 1000s)	Power (mW)
1 VC	Base Router	41,98	21.79
	Ariadne	44,64	23.54
2 VCs	Base Router	83,44	33.24
	Ariadne	87.05	34.78
	Hermes-XY	87.87	36.50
3 VCs	Base Router	135,44	56.95
	Ariadne	139,39	60.05
	Hermes-O1T	141,50	60.33

above dominated the initial research attempts in FT INs. Under (1-a), Dally's early reliable router was a 1-FT architecture [7], while the work by Glass and Ni only allowed (n-1) faults in an n-dimensional topology (just one in a 2D mesh) [19]. Many related early works can be found in [6], [10]. Under category (1-b), in an effort to define permissible fault patterns in the topology to avoid deadlocks and simplify FT routing, researchers utilized block-structured shapes, such as convex and/or concave regions of faults, at the expense of victimizing healthy links and routers as being unhealthy so as to form such regions. Significant works in [17], [21], [32] fall in this class.

Under category (1-c), [15] proposes distributed routing algorithms that re-configure the network to avoid faulty components. The Vicis router [14] employs specialized BIST testers in each router to detect faults, and then leverages extensive re-configurability and an appropriately-designed routing algorithm to perform port-swapping and crossbar bypassing. Next, Ariadne [1] uses up*/down* routing to reconfigure routing tables when faults in links occur, while uDIREC [23] reconfigures routing paths in a faulty NoC also using up*/down* routing, where an iterative software-based spanning tree kernel utilizes NoC path diversity to optimize routes and achieve smaller link victimization vs Ariadne using unidirectional links.

Under category (2), Bulletproof [5] analyzes the reliability vs. area tradeoffs of various NoC router designs and proposes run-time repair and recovery methodologies at the system-level. Next, [31] makes use of partially faulty links, while [13] converts a uni-directional link to work in a full-duplex mode in case the node-pairing link fails. Lastly, under category (3), stochastic communication techniques [11] use probabilistic packet broadcasting schemes to handle faults.

VI. CONCLUSIONS

This paper presented Hermes, a top-performing, distributed, and deadlock-free FT routing algorithm with high robustness and graceful performance degradation with increasing faulty link counts. Hermes is a hybrid routing scheme: it balances traffic to sustain high performance onto fault-free paths, while it provides pre-configured escape path selection in the vicinity of faults. Hermes improves throughput by up to 3×, and is able to identify network segmentations.

REFERENCES

[1] K. Aisopos et al. *ARIADNE: Agnostic Reconfiguration in a Disconnected Network Environment.* Proc. ACM PACT, pp. 298–309, Oct. 2011.

[2] N. L. Binkert et al. *The M5 Simulator: Modeling Networked Systems.* IEEE Micro Magazine, Vol. 26, No. 4, pp. 52–60, July/Aug. 2006.

[3] C. Bienia et al. *The PARSEC Benchmark Suite: Characterization and Architectural Implications*. Proc. ACM PACT, pp. 72–81, Oct. 2008.

[4] J. R. Black. *Electromigration Failure Modes in Aluminum Metallization for Semiconductor Devices*. Proc. of the IEEE, Vol. 57, No. 8, pp. 1587–1594, Sept. 1969.

[5] K. Constantinides et al. *BulletProof: A Defect-Tolerant CMP Switch Architecture*. Proc. IEEE HPCA, pp. 5–16, Feb. 2006.

[6] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc. ISBN: 9780122007514, 2004.

[7] W. J. Dally et al. *The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers*. Proc. Int'l Parallel Computer Routing and Communication Workshop, pp. 241–255, May 1994.

[8] A. DeOrio et al. *DRAIN: Distributed Recovery Architecture for Inaccessible Nodes in Multi-Core Chips*. Proc. DAC, pp. 912–917, June 2011.

[9] J. Duato. *A Theory of Fault-Tolerant Routing in Wormhole Networks*. IEEE TPDS, Vol. 8, No. 8, pp. 790–802, Aug. 1997.

[10] J. Duato et al. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann. ISBN: 1558608524, 2002.

[11] T. Dumitras and R. Marculescu. *On-Chip Stochastic Communication.* Proc. DATE, pp. 790–795, Mar. 2003.

[12] M. Evripidou et al. *Virtualizing Virtual Channels for Increased Network-on-Chip Robustness and Upgradeability*. Proc. IEEE VLSI, pp. 21–26, Aug. 2012.

[13] M. A. Al Faruque et al. *Configurable Links for Runtime Adaptive On-Chip Communication*. Proc. DATE, pp. 256–261, April 2009.

[14] D. Fick et al. *Vicis: A Reliable Network for Unreliable Silicon*. Proc. DAC, pp. 812-817, July 2009.

[15] D. Fick et al. *A Highly Resilient Routing Algorithm for Fault-Tolerant NoCs*. Proc. DATE, pp. 21–26, April 2009.

[16] J. Flich et al. *Improving Routing Performance in Myrinet Networks.* Proc. PDPS, pp. 27–32, May 2000.

[17] J. Flich et al. *An Efficient Implementation of Distributed Routing Algorithms for NoCs*. Proc. ACM/IEEE NoCS, pp. 87–96, May 2008.

[18] C. J. Glass and L. M. Ni. *The Turn Model for Adaptive Routing*. Proc. ISCA, pp. 278–287, May 1992.

[19] C. J. Glass and L. M. Ni. *Fault-Tolerant Wormhole Routing in Meshes Without Virtual Channels.* IEEE TPDS, Vol. 7, No. 6, pp. 620–636, June 1996.

[20] J. Hestness et al. *Netrace: Dependency-Driven Trace-Based Network-on-Chip Simulation.* Proc. Int'l Workshop on Network on Chip Architectures, pp. 31–36, Dec. 2010.

[21] S. P. Kim and T. Han. *Fault-Tolerant Wormhole Routing in Meshes with Overlapped Solid Fault Regions*. Journal on Parallel Computing, Vol. 23, No. 13, pp. 1937–1962, 1997.

[22] R. Marculescu et al. *Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives.* IEEE TCAD, Vol. 28, No. 1, pp. 3–21, January 2009.

[23] R. Parikh and V. Bertacco. *uDIREC: Unified Diagnosis and Reconfiguration for Frugal Bypass of NoC Faults*. Proc. IEEE/ACM Micro, pp. 148–159, Dec. 2013.

[24] L.-S. Peh and W. J. Dally. *A Delay Model and Speculative Architecture for Pipelined Routers*. Proc. IEEE HPCA, pp. 255–266, Jan. 2001.

[25] M. D. Powell et al. *Architectural Core Salvaging in a Multi-Core Processor for Hard-Error Tolerance.* Proc. IEEE/ACM ISCA, pp. 93–104, June 2009.

[26] J. C. Sancho et al. *An Effective Methodology to Improve the Performance of the Up*/Down* Routing Algorithm.* IEEE TPDS, Vol. 15, No. 8, pp. 740–754 , Aug. 2004.

[27] Semiconductor Industry Association, 2014. Int'l Technology Roadmap for Semiconductors. Available [online]: http://www.itrs.net/reports.html

[28] D. Seo et al. *Near-Optimal Worst-Case Throughput Routing for Two Dimensional Mesh Networks.* Proc. IEEE/ACM ISCA, pp. 432-443, June 2005.

[29] F. Silla and J. Duato. *High-Performance Routing in Networks of Workstations with Irregular Topology.* IEEE TPDS, Vol. 11, No. 7, pp. 699–719, August 2002.

[30] S. R. Vangal et al. *An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS*. IEEE JSSC, Vol. 43, No. 1, pp. 29–41, Jan. 2008.

[31] A. Vitkovskiy et al. *A Dynamically Adjusting Gracefully Degrading Link-Level Fault-Tolerant Mechanism for NoCs.* IEEE TCAD, Vol. 31, No. 8, pp. 1235–1248, Aug. 2012.

[32] J. Wu. *A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model*. IEEE TOC, Vol. 52, No. 9, pp. 1154–1169, Sept. 2003.