Fault-Tolerant 3-D Network-on-Chip Design using Dynamic Link Sharing

Seyyed Hossein Seyyedaghaei Rezaei[1], Mehdi Modarressi[1,2], Reza Yazdani Aminabadi[1], Masoud Daneshtalab[3]

[1]Department of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran
[3]Department of Electronic and Embedded Systems, Royal Institute of Technology, Stockholm, Sweden

{s.hseyyedaghaei,modarressi,yazdani67}@ut.ac.ir, masdan@kth.se

Abstract. The most important challenge in the emerging 3D integration technology is the higher temperature, particularly in the layers that are more distant from the heat sink, compared to planar 2D chips. High temperature, in turn, increases circuit's susceptibility to permanent and intermittent faults. On the other hand, the fast and high-bandwidth vertical links in the 3D integration technology have opened new horizons for network-on-chip (NoC) design innovations. In this paper, we leverage these ultra-low-latency vertical links to design a fault-tolerant 3D NoC architecture. In this architecture, permanent and intermittent defects on links and crossbars are bypassed by borrowing the idle bandwidth from vertically adjacent links and crossbars. Evaluation results under synthetic and realistic workloads show that the proposed fault-tolerance mechanism offers higher reliability and lower performance loss, when compared with state-of-the-art fault-tolerant 3D NoC designs.

I. Introduction

3D integration has become an important means of improving performance as process scaling stops. This technology allows the core count to be increased by stacking multiple layers on top of each other interconnected by short vertical connections [1]. Among different vertical connection technologies, Through-Silicon Vias (TSVs) has attracted more attention due to their higher bandwidth, lower latency, and better compliance with standard CMOS process. The 3D integration increases power density due to the proximity of active layers. Higher temperature than 2D chips, particularly in the layer most distant from the heat sink, is the most undesirable side-effect of 3D integration.

High temperatures and temperature variations across the layers play an important role in increasing both intermittent and permanent fault rates; temperature has a direct impact on aging and other sources of permanent faults. It also leads to up to 30% variation in the performance of on-chip components, which is the main source of intermittent delay faults [2-4]. Indeed, temperature makes on-chip components too slow to complete computation and communication within a single clock cycle, causing setup and hold timing violations that eventually generate incorrect output [2].

Faults can be roughly classified as transient, intermittent, and permanent [2]. Transient faults appear randomly for one or several cycles and mainly affect data (e.g. flipping of bits in memory cells), but not the circuit. Intermittent faults appear under some conditions, such as high temperature, and affect the system as long as the condition is met. The faulty part will stop malfunctioning when the working condition (temperature, for example) becomes normal again. Permanent faults occur when transistors or wires are permanently open or shortcut or too slow to meet timing constraints. These problems can be made by manufacturing defects or aging process.

Among different on-chip components, long links and the large distributed structure of NoCs makes them very susceptible to faults generated by production or environmental influences.

In this paper, we present an architecture-level mechanism to tackle permanent and intermittent faults in 3D NoCs. The proposed mechanism relies on some slight modifications in router microarchitecture to enable it to temporarily replace a faulty link with the vertically adjacent links in the upper and lower layers. This way, packets of a faulty link borrow the idle bandwidth from the links in the same location at the upper or lower layers. They then use the low-latency TSVs to move to the remote layer, pass the remote link, and take another TSV at the downstream router to move back to their original layer, all in a single clock cycle.

TSVs provide ultra-low latency inter-layer communication with less than 3ps latency [6], thus two links at adjacent layers that are connected by TSVs behave almost as though they are next to each other in the same layer.

Our observations show that under uniform traffic, the probability that three vertically adjacent links in a 3-layer 3D NoC will be used at the same cycle is small and even high load, is below 25%. Our observation is confirmed by some prior work that report low link utilization in NoCs even in high traffic loads [7-8]. Therefore, we can speculate that most of the time that a flit has to pass a faulty link, it will find at least one vertically adjacent link idle to bypass the fault. Link borrowing does not introduce any congestion to the host layer, as flits are only allowed to borrow idle cycles that otherwise would be unused.

Currently, this method bypasses link and crossbar faults by borrowing bandwidth from other layers. Handling permanent and intermittent faults in buffers by a similar resource borrowing approach is left for future work.

Fault-tolerant routing algorithms are the most common and straightforward way of tackling NoC faults by leveraging the inherent path redundancy in NoCs. Our proposal is orthogonal and can be used along with any fault-tolerant routing algorithm to further increase reliability. Our method, however, is a better solution than routing, when the network topology (available and faulty links) varies over time frequently. This situation is very likely to happen by temperature-induced intermittent faults. The temperature of the links and other NoC components is directly related to the traffic load on them. Consequently, the intermittent faults caused by high temperature will appear on highly-loaded links and disappear when the faulty link remains idle for a period of time. While it is hard for routing methods to adapt themselves to varying topology and still guaranteeing load balancing and deadlock and livelock freedom [9-10], our method enables NoC to keep using its baseline routing method and operate normally. More precisely, packet's path is intact from the routing point of view and so, the proposed fault-tolerant link sharing introduces no deadlock, provided that the baseline routing algorithm is deadlock-free.

It also does not need the global information about the location of faulty links to adapt routing to the new NoC topology. The reason is that the link borrowing is done during switch arbitration and is transparent to the routing.

Evaluation results show that the proposed mechanism can minimize the performance loss of a faulty 3D NoC when compared to state-of-the-art fault tolerant 3D NoC designs.

II. Related Work

As mentioned before, this paper focuses on intermittent and permanent faults. Whereas transient faults and soft errors are mainly handled by error correction codes, more considerations are needed to cope with permanent and intermittent faults. These faults are tackled by different approaches at different levels including fault-tolerant routing algorithms [4][9-12], router's micro-architectures [14-15], and network topologies [13].

Designing fault tolerant routing algorithms is the most basic method to bypass faulty components. However, they all need sophisticated deadlock and livelock avoidance methods [2].

A fault-tolerant routing algorithm based on *Hamiton* path strategy is presented in [9]. This method ensures tolerating one faulty link at each part of a 3D NoC with 95% reliability. However, the limitations imposed to the routing algorithm to keep it deadlock-free affects the network performance as injection rate goes high.

In [14], a bypassing approach for faulty routers and links is presented, but it needs many virtual channels per port for deadlock freedom.

In [5] and [6], the viability of resource sharing across layers in 3D NoCs and MPSoCs is demonstrated. In [13], an architectural solution is proposed to handle faults on TSV links. In order to tolerate a failure, TSV links are arranged as an Omega network. Only half of the links are required to keep the network completely reliable, however more performance degradation is caused when number of faults increases.

As our method imposes no restriction on the routing algorithm and topology, almost all of the above methods are orthogonal to our proposal and can be applied along with it to further increase reliability. Moreover, our method only needs local fault information and does not affect the routing algorithm. Consequently, it achieves high reliability with minimum negative effect on other parts of the NoC.

III. Proposed Fault-tolerance Mechanism

Link fault in our design is tolerated by using the idle bandwidth of the links with the same topological location in the adjacent layers. To bypass faulty links, fast TSVs are utilized to move flits across layers. A baseline 3D mesh NoC features 5 bidirectional links in the 2D plane (4 links to adjacent routers, one link to local processor) and 2 links to adjacent routers along the third dimension. They forward packets in a pipelined fashion that consists of buffer write, routing, VC allocation, switch output arbitration, crossbar switch traversal, and link traversal operations.

We first show how a fault-tolerant 3D NoC can be implemented by adding two extra TSV to a baseline conventional router architecture. We then reduce the area overhead of the proposed mechanism by reusing the regular vertical links of a 3D router for vertical link sharing.

a. Fault tolerant router design

Fig. 1 outlines the main idea of the 3D link sharing. In this scenario, the faulty link between nodes 1 and 2 is dynamically replaced with the corresponding link at the upper layer (between nodes 17 and 18); a packet uses 1→17 TSV to go up and 18→2 TSV to go back to its original layer and enters the VC allocated to it during the VC allocation stage at node 1. Actually, the routing and VC allocation logic are not needed to be aware of faulty links; it is the output arbiter that knows if a downstream link is faulty and so, is responsible to request the link bandwidth from adjacent layers.

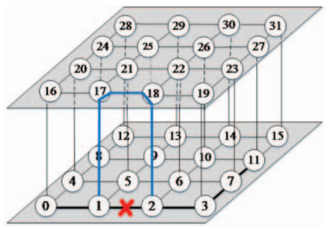

Fig. 1. The faulty link is bypassed by borrowing a link from the upper layer

Fig. 2 shows the microarchitecture of the proposed router. Components that are unique to (or modified by) the proposed fault-tolerant method are shown in gray.

This figure shows a router with two extra TSVs that connect it to the routers at the same location at the upper and lower layers. Obviously, routers at the first and last layers just need a single TSV per router. The 64-bit TSVs are used in either out-going or incoming direction. The extra components establish a bypass path around a faulty link. As Fig. 1 shows, bypassing the faulty 1→2 link requires four turns at nodes 1, 18, 17, and 2, as follows.

Router 1. When the arbiter of a faulty link receives a request, it forwards the request to the arbiters of vertically adjacent routes at the upper and lower layers (*remote* arbiters, hereinafter).

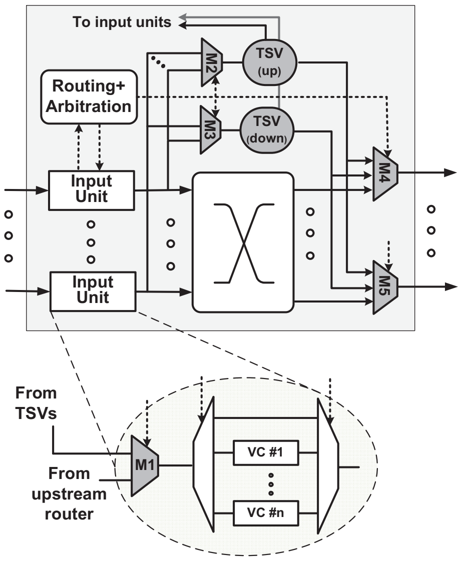

Fig. 2. The proposed router architecture

Upon successful remote link allocation, the flit is sent to the up or down TSVs through M2 or M3 multiplexers. The select line of the multiplexers is set by the arbiter.

Router 17. Once the flit reaches router 17, it is directly sent to the link between routers 17 and 18. This path is set up by multiplexers M4 and M5.

Router 18. When the flit reaches the destination column, it uses the right TSV (up or down) to return to its original layer. To this end, it does not enter the input unit at the downstream router, but skips over it by the bypass wire at the input port. It then passes through multiplexers M2 and M3 to go back to its original layer.

Router 2. As the final destination, router 2 connects the TSV on which the flit is traveling to the appropriate input unit through M1, in order to deliver the flit to its right VC allocated at the upstream node.

Please note that this method can bypass both faulty link and faulty crossbar. To reduce TSV area overhead, this work restrict routers to borrow links from their immediate neighbors at upper and lower layers, even if the NoC has more than three layers. A fault tolerant routing is inevitable in the unlikely case that three links at the same position in three adjacent layers are faulty.

b. *Link bandwidth borrowing procedure*

Arbiters with faulty link. When a link is faulty, the arbiter of the corresponding output port forwards allocation requests to the arbiters of the same output port at the adjacent routers in the upper and lower layers. Actually, the local arbiter of a faulty link aggregates the local request (by an OR gate) to directly send to remote arbiters. During the SA stage, if the arbiter receives any grant from adjacent arbiters, one of the requesting flits will be selected and forwarded to the remote allocated link. The arbiter sets up the bypass path by appropriately setting the select line of the multiplexers.

If the arbiter of a faulty link with more than one link allocation request receives two remote grants, it can send two flits in parallel. This parallel transfer will not affect packet integrity, as each flit is written to the VC secured by its header. This parallel transfer involves slightly modifying the input unit in Fig. 2 and connecting the two TSVs (and also the regular link that comes from the demultiplexer) to each individual VC via a multiplexer.

Remote arbiters. Form the remote arbiter's point of view, if an arbiter has at least one request from its local ports, it ignores all remote requests and just services the local ones. Otherwise, the arbiter considers the remote request lines and selects one remote request (if any) out of at most two requests from the upper and lower layers to grant based on a round robin policy. The grant is then sent to the initiating arbiter and from there, to the requesting VC. The two arbiters then appropriately set the select line of the multiplexers at the four routers involved in the bypassing procedure. Each router sets the configuration in its own router and the downstream router in its layer (e.g., the arbiter of router 17 configures data-paths in router 17 and 18 in Fig. 1). The path at the downstream router is set in the same way as the demultiplexers at the input ports of a conventional router are set by the upstream router to direct flits to the right VC buffer.

Allocation requests of a faulty link are forwarded to both upper and lower layers to increase successful remote allocation chance. This request may be granted by two layers; it will use one grant, but the other grant (more precisely, the allocated link bandwidth associated with this grant) is wasted.

By giving higher priority to local requests during switch allocation, it is guaranteed that the bandwidth that could be used by a local request will never go idle because it was granted to a remote request that will not use it.

c. *Handling bypass path conflicts*

If more than one incoming links of a router are faulty, a conflict may arise between the bypass paths that the source of faulty links may set up at the same cycle. For example, assume that links 1→2 and 6→2 in Fig. 1 are faulty. In this case, both node 1 and node 6 might send their flits simultaneously through bypass paths (via the upper layer) to node 2. Since the 18→2 TSV can only forward down one of the packets, this results in a conflict.

Another potential conflict scenario occurs when 1→2 and 2→6 links are faulty; two different flits from nodes 1 and 2 that start to bypass the faulty links at the same cycle will arrive (and collide) at the 18→2 TSV simultaneously, since TSVs are used in both directions.

The conflict in the latter case is eliminated by informing adjacent routers in the 2D plane about the availability of the TSVs in the next cycle. This is done by using a special signal, called *mask* signal, which is sent from each downstream router to its four upstream routers in the 2D plane. At each cycle, if an arbiter in router *x* has sent a remote allocation request to service its local flits (by reserving a bypass path), it sends a mask signal to its neighbors to prevent them from setting up a bypass path destined to *x*. The signal reaches the neighbors at the end of the arbitration stage, but masks and cancels the possible grant that a flit may receive. Routers that have no faulty links, or have at most one faulty link, do not need to drive the mask signal, as no conflict will happen on their TSVs.

The elimination of conflicts in the former case is more challenging, because it takes two cycles for each router to know about a conflicting flit. To handle this unlikely situation, if two upstream nodes have a faulty link to the same downstream node (nodes 6 and 1 in Fig. 1 in this example), one of them uses the lower layer and the other uses the upper layer to bypass faulty links. If the number of faulty links increase or the routers are at the first or last layer (hence, only have a single adjacent layer to use for bypassing), the arbitration for faulty links at those nodes will take two cycles to ensure correct mask signal propagation. In this case, the destination node (node 2 in this example) sees the attempt to establish a bypass path by the upstream routers at the first arbitration cycle and sends the mask signal to one of them, based on a round-robin policy, at the second cycle.

d. *Timing overhead*

As a typical NoC in 45 nm works at 2 GHz, we should guarantee that the architectural changes we made in a baseline wormhole-switched router do not increase the router critical path beyond 500 ps. We implemented different components of a router in SPICE to calculate their latency. The latency of TSVs in 45nm is also taken from [6] and is set to 1.3ps.

The proposed architecture just affects link traversal and arbitration latency. The arbitration's critical path delay is not changed as the arbiters with local requests simply ignore remote ones. Moreover, the arbiter of a faulty link diverts requests to the upper and lower arbiters. As the arbiters are vertically adjacent and are connected by a 1.3ps vertical link, request will be ready at the input of the remote arbiters well before the clock edge at which the request lines are sampled by arbiters.

A request that is sent to a remote arbiter is serviced even faster than the requests that are serviced locally, because it experiences the delay of a TSV traversal (less than 2ps) to send out request, a 2-to-1 round robin remote arbitration (39ps), and a second TSV traversal to receive grant. It is smaller than the regular 6-to-1 arbiter latency that we calculate as 134ps through SPICE simulation. Please note that if the remote arbiter has requests from its own layer, requests will receive no grant signal.

The regular 1mm link and crossbar traversal's latency, which we calculate as 161ps, is replaced by one link and two TSV traversal latencies in our design. Flits should also pass through some multiplexers:

- M2 or M3: to move to TSV to reach the upstream node
- M4 or M5: to reach the link at the remote layer
- M1, input unit demultiplexer and multiplexer, M2 or M3: to go back to its original layer through a TSV
- M1: to enter its right VC

These components add the 1mm link latency of 110ps by extra 98ps latency. Thus the crossbar+link traversal in a baseline router is replaced by 2 TSV, 7 multiplexer (most are small 2-to-1 logic), and one link traversals that results in a total latency of 210ps, that is still shorter than the clock period in 2 GHZ.

e. *Area-optimized fault tolerance*

The fault tolerance in our design is achieved at the price of two extra TSVs per router. Assuming 64-bit NoCs, every pair of vertically adjacent routers are connected by 64 TSVs. Consequently, a router at the middle layer requires 2 bundles of 64 TSVs to connect to the upper and lower layers. In addition, each arbiter requires four pairs of TSVs to send and receive remote requests (one pair per each output port) and another four TSV pairs for grant signals. Like data TSVs, the number of control TSVs is doubled for routers in middle layers.

To evaluate the area overhead, we use the TSV area estimation model presented in [6] for the 45nm technology point. According to this model, the total block out area of the TSVs used in our design is estimated as 14400μm^2 per router at a middle layer (7600 μm^2 for the first and last layers), which imposes 10.6% overhead to the total router area. This overhead is increased to 14% when considering the area of extra multiplexers. The router area is calculated as 0.136 mm^2 by the area model developed in [16] for a NoC with parameters described in Section 4. This area overhead can be considered negligible as compared to the area of a multi-core processor, which is usually more than 100mm^2.

One way to decrease this area overhead is to halve the number of TSVs, but double their working frequency to keep their latency of 64-bit flit transfer intact. The ultra-low latency of TSVs allows us to run them at very high frequencies, as shown by many prior works [17]. With 32 bit TSVs, every 64-bit flit is transferred in two TSV cycles (each 32 bit word in a single cycle), but the entire transfer takes a single router cycle. This technique can be also applied to halve the number of control TSVs.

Another method that eliminates the TSV overhead is to reuse the TSVs that act as vertical links in a 3D NoC (regular links along the third dimension) for link sharing. In this scheme, which is beneficial when either the traffic load or the fault rate is small, bypass paths are set up on the regular TSVs of router. To this end, all connections (extra multiplexer inputs and outputs) of the two bypassing TSVs in Fig. 2, should now be connected to the TSVs of the regular vertical links.

This method would have negligible performance loss when network is lightly loaded; in this case the probability that the vertical links are idle when they are needed to be used to bypass a faulty link is small and hence, can be used for faulty link bypassing most of the time. The same mask signal that was used to cancel a remote grant in case of a TSV conflict is used in this design to prevent remote link allocation, when the downstream router has flits to forward on its vertical links.

In Section 4, we evaluate this low-overhead method and show that it can make an appropriate trade-off between area and performance under many realistic workloads.

f. Reliability of vertical links

Although this paper focuses on the regular 2D links, the redundant TSVs can also be used to increase the reliability of vertical links. This fault-tolerance capability can be achieved by some minor changes in the router architecture depicted in Fig. 2.

Even if the TSVs of the regular vertical links are re-used for link bypassing, where there are no redundant TSVs, the constant-bandwidth TSV reduction mechanism described in Section 3.f can be applied for fault-tolerance. To this end, when several TSVs of an n-TSV links are faulty (up to n/2 faulty TSVs), the NoC selects n/2 working TSVs to send flits and deactivates the others. These TSVs work at double frequency. When the number of faulty TSVs increases between n/2 and 3n/4, a quarter of working TSVs would be utilized at 4x frequency. This strategy can continue as long as the frequency can be scaled up. In case the entire TSVs of a link are faulty, this fault should be bypassed by routing algorithm.

IV. Experimental Results

In this section, we evaluate the efficiency of the proposed fault-tolerant method under a set of synthetic and realistic traffics. The uniform and hot-spot traffic profiles are used as synthetic traffic and for real workload, the Netrace library that models the PARSEC traffic [18] is utilized. Experiments are carried out by BookSim, a cycle accurate NoC simulator.

A NoC with adaptive routing presented in [19] that allows packets to take any shortest paths between the source and destination routers in a per-plane basis and HamFA [9] are selected for comparison purpose.

All considered NoC configurations adopt a wormhole-switched router with 3-stage pipeline: routing, VC allocation and speculative allocation, crossbar and link traversal. Based on the analysis of Section 3.e the same working frequency is considered for the three NoCs. The topology of all NoCs is a 4×4×3 3D mesh with 64-bit links. Each input port has two VCs with the buffer size of 8 flits. However, the adaptive router needs three VCs to guarantee deadlock avoidance.

Packet size is set to eight flits for synthetic traffic profiles, but Netrace has a different packet size for each message class.

Performance analysis. In Fig. 3, the average packet latency of the three considered NoCs are measured for one faulty link under both uniform random (Fig. 3.a) and hotspot (Fig. 3.b) traffic profiles as a function of packet injection rate. To model intermittent faults, the fault location, (the faulty link) changes every 500K cycles.

As the results indicate, our proposed method outperforms the HamFA and adaptive routing algorithms in presence of faults. This is because the HamFa routing algorithm adopts some policies to prevent deadlock which sacrifices its performance in high injection rates. The adaptive routing should also direct packets to a longer path to bypass faulty links, whereas our method does not increase packet's path length. Please note that the adaptive router has larger area than our method and HamFa, since it needs more VCs. Fig. 4 shows the average packet latency under Netrace workloads. Again, the figure shows that the proposed method gives better performance than its counterparts.

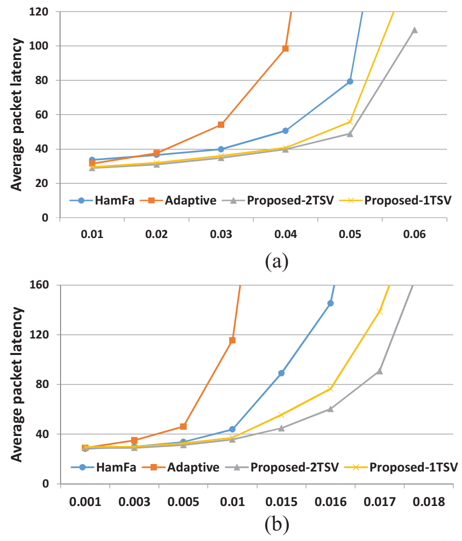

Fig. 3. Average packet latency (cycle) under (a) uniform and (b) hot-spot traffic profiles for one faulty link for the baseline proposed method (proposed-2TSV) and its alternative design that reuses regular TSVs (proposed-1TSV). The x axis shows the injection rate (packet/node/cycle)

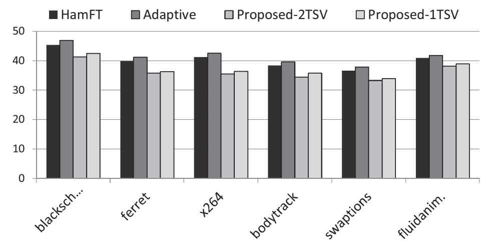

Fig. 4. Average packet latency (cycle) under PARSEC traffic for one faulty link for the baseline proposed method (proposed-2TSV) and its alternative design that reuses regular TSVs (proposed-1TSV)

An important observation about the proposed mechanism is that the NoC that reuses the regular router TSVs for link sharing (proposed-1 TSV) has roughly the same performance as the baseline method which devotes an extra TSV for link sharing.

It is, in particular, the case in low and mid traffic injection rates, where most commercial and server workloads work [8]. Consequently, the proposed method can bring fault-tolerance for 3D NoCs with insignificant area overhead.

We also added DSENT [11] network power estimation library to BookSim to evaluate power consumption. Under Netrace traffic, the proposed method increases the NoC power consumption by around 8% over the baseline. Since in most CMPs, the contribution of the NoC power consumption to the total chip power is low [8][20], we do not expect that the proposed method worsens thermal problems of the 3D integration.

Reliability Evaluation. To evaluate and compare the reliability of proposed method with HamFA, the number of faulty links increases from 1 to 8 under uniform traffic. The injection rate is set to 0.2 at which a normal NoC works well below the saturation point. All simulation and architectural parameters are the same as the previous experiments and the faulty links are selected randomly. We repeat each simulation 100 times to calculate in how many experiments the NoC is reliable. A network is reliable if (1) the network works well below the saturation point (less than 2x increase in zero load latency) and (2) all the injected packets reach their destinations. Please note that the first condition is a tight reliability constraint, so the reliability values reported in Fig. 5 are lower than the reliability values found in many related papers, which only consider packet delivery.

As shown in Fig. 5, the proposed method has better reliability than HamFa, as it uses a fine-grain link sharing to bypass faulty links with no hop-count overhead and deadlock-avoidance costs; it only borrows the unused bandwidth of adjacent links to replace faulty links.

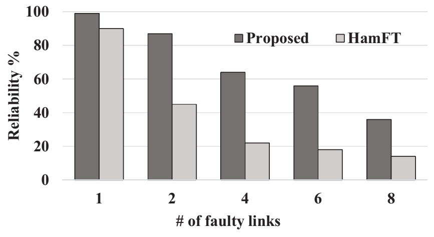

Fig. 5. Reliability comparison under uniform traffic

V. Conclusion

In this paper, we proposed a fault-tolerant dynamically reconfigurable router's micro-architecture for 3D NoCs. It tolerates multiple faulty links by enabling a very fine-grained resource sharing mechanism between different layers by using ultra-fast vertical links (TSVs). In other words, the faulty link is bypassed through borrowing the idle link bandwidth from the upper or lower vertically adjacent routers. Evaluation result shows that our method gives higher reliability than the other considered 3D fault-tolerant NoC designs. This method does not require any global information about the fault location. It also is not involved in complex mechanisms to guarantee deadlock and livelock freedom as the baseline routing method can still be used in the faulty NoC.

Not affecting routing algorithm, this method can be added to many high performance 3D NoC designs to increase their reliability. In particular, this method can be coupled with NoCs with full 3D crossbar [15][21], which fuse the crossbars of all routers in the same vertical column into a unified module, to make them fault-tolerant.

References

[1] D. H. Woo, et al., "An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth", in *Proc. of HPCA*, 2010.

[2] M. Ratezski et al., "Methods for fault tolerance in networks-on - chip", in *ACM Computing Surveys*, Vol. 14, No. 1, 2013.

[3] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits", in *Proc. of Reliability and Maintainability Symposium*, 2008.

[4] S. Pasricha, Yong Zou, "NS-FTR: A fault tolerant routing scheme for networks on chip with permanent and runtime intermittent faults", in *Proc. of ASP-DAC*, 2011.

[5] H. Homayoun, et al., "Dynamically Heterogeneous Cores through 3D Resource Pooling", in *Proc. of HPCA*, 2012.

[6] H. S. Rezaei et al., "Dynamic Resource Sharing for High-Performance 3-D Networks-on-Chip", in *IEEE Computer Architecture Letters*, 2015.

[7] Y. C. Lan et al., "BiNoC: A Bidirectional NoC Architecture with Dynamic Self-Reconfigurable Channel", in *Proc. of NOCS*, 2009.

[8] S. Volos et al., "CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers", in *Proc. of NOCS*, 2012.

[9] M. Ebrahimi, et al., "Fault-Tolerant Routing Algorithm for 3D NoC Using Hamiltonian Path Strategy", in *Proc. of DATE*, 2013.

[10] D. Fick et al., "A highly resilient routing algorithm for fault-tolerant NoCs", in *Proc. of DATE*, 2009.

[11] Chen Sun, et al., "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling", in *Proc. of NOCS*, 2012.

[12] B. Ahmed et al., "Deadlock-Recovery Support for Fault-tolerant Routing Algorithms in 3D-NoC Architectures", in *Proc. of MCSoC*, 2013.

[13] C. Hernandez, et al., "Fault-Tolerant Vertical Link Design for Effective 3D Stacking", in *Computer Architecture Letters,* Vol. 10, No. 2, 2011.

[14] M. Ebrahimi, et al., "Fault-tolerant Method with Distributed Monitoring and Management Technique for 3D Stacked Mesh, in *Proc. of International Symposium on computer Architecture & Digital Systems*, 2013.

[15] J. Kim, et al., "Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures", in *Proc. of ISCA*, 2007.

[16] M. Modarressi, et al., "Application-Aware Topology Reconfiguration for On-Chip Networks", in *IEEE Trans. VLSI Sys.*, Vol. 19, No. 11, 2011.

[17] A. M. Rahmani, et al., "Developing a Power-Efficient and Low-Cost 3D NoC using Smart GALS-Based Vertical Channels", in *Journal of Computer System Sciences*, Vol. 79, No. 14, 2013.

[18] J. Hestness, B. Grot, S. W. Keckler, "Netrace: Dependency-Driven, Trace-Based Network-on-Chip Simulation", in *Proc. of NoCArc*, 2010.

[19] N. Dahir, et al., "Deadlock-free and plane-balanced adaptive routing for 3D NoCs", in *Proc. of NoCArc*, 2012.

[20] Single-chip Cloud Computer, http://www.intel.com, 2015.

[21] S. Jeloka, et al., "Hi-Rise:A High-Radix Switch for 3D Integration with Single-cycle Arbitration", in *Proc. of MICRO*, 2014.