Scientific Research Publishing

Journal of Computer and Communications, 2016, 4, 1-10
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Twist-Routing Algorithm for Faulty Network-on-Chips

Kunwei Zhang, Thomas Moscibroda

Tsinghua University, Beijing, China
Email: aceeca.135531@gmail.com, moscitho@microsoft.com

How to cite this paper: Zhang, K.W. and Moscibroda, T. (2016) Twist-Routing Algorithm for Faulty Network-on-Chips. *Journal of Computer and Communications*, **4**, 1-10.
http://dx.doi.org/10.4236/jcc.2016.414001

Received: September 8, 2016
Accepted: November 8, 2016
Published: November 11, 2016

Abstract

This paper introduces Twist-routing, a new routing algorithm for faulty on-chip networks, which improves Maze-routing, a face-routing based algorithm which uses deflections in routing, and archives full fault coverage and fast packet delivery. To build Twist-routing algorithm, we use bounding circles, which borrows the idea from GOAFR+ routing algorithm for ad-hoc wireless networks. Unlike Maze-routing, whose path length is unbounded even when the optimal path length is fixed, in Twist-routing, the path length is bounded by the cube of the optimal path length. Our evaluations show that Twist-routing algorithm delivers packets up to 35% faster than Maze-routing with a uniform traffic and Erdös-Rényi failure model, when the failure rate and the injection rate vary.

Keywords

Network-on-Chip (NoC), Fault-Tolerant Routing, Maze-Routing Algorithm, GOAFR+ Algorithm, Bounding Circle

1. Introduction

The transistor technology scales in microprocessors, and more and more power-efficient cores are integrated on a single chip. The communication between these on-chip cores should be efficient. Therefore, Networks-on-chips (NoCs), instead of simple buses, are becoming a promising choice for on-chip interconnects for their better scalability [1]-[6]. Unfortunately, the reliability of the on-chip components is reduced as critical dimensions shrink, and a NoC might be a single point of failure [7]. As the silicon ages, the error rates become quite high [8], because of oxide breakdown, electromigration, and thermal cycling [7]. Hence, it is critical that some failures in the network do not cause an entire chip to fail.

There are some NoC reliability solutions based on architectural protection against

faults in the router logic [9] [10] [11]. But not all faults can be toleranted this way [12]. In recent works, faults are modeled by disabling such links, and a complete router loss is modeled by marking all the links connected to the affected router as faulty. The goal is to route packets around faults and finally reach the destination. Recent route-reconfiguration solutions to bypass faulty links or routers can be broadly divided into two kinds, buffered solutions and deflection solutions. Buffered solutions include Ariadne [13], uDirec [12], Hermes [14], which all utilize traditional wormhole routing [15], and routing tables. Those algorithms typically take some time to update routing tables when a new fault is detected, and incur reconfiguration overhead. The deflection solutions for non-faulty chips are introduced by BLESS algorithm [16] to overcome the significant energy consumption and design complexity caused by buffer usage. Then, CHIPPER [17] and minBD [18] develop the idea of deflection routing. For faulty chips, the Maze-routing algorithm provides a deflection routing algorithm, which is the first routing algorithm which provides guaranteed delivery in a fully-distributed manner at low cost and low reconfiguration overhead [19].

The Maze-routing is the state-of-the-art solution of deflection routing for faulty chips. However, the path length which is found by Maze-routing is unbounded even when the optimal path length is fixed. We proposed a improved algorithm named Twist-routing, taking inspiration from the idea of GOAFR+ routing algorithm, which was originally proposed for ad-hoc wireless networks [20] [21] [22]. Using our algorithm, the path length is bounded by the cube of the optimal path length. Our algorithm inherits the property of Maze-routing, and provides guaranteed delivery at low cost and the same low reconfiguration overhead. The experiments show that our algorithm is 35% faster than Maze-routing when the failure rate equals to 0.3, and the injection rate is 0.003, and keeps fast when injection rate increases.

2. Twist-Routing Algorithm

The Twist-routing algorithm is a practical routing algorithm for faulty NoCs, which is based on Maze-routing for faulty NoCs and GOAFR+ routing algorithm for ad-hoc wireless networks. The faulty model is described in Section 0. We briefly review the Maze-routing algorithm in Section 2.1. In Maze-routing, a packet is alternately in greedy and face-routing [23] mode. In Twist-routing, these two modes remains, but we use bounding circles to limit the search range in a face-routing step, proposed in Section 2.3. This enables us to prove a theoretical bound of Twist-routing in Section 2.4. The interactions of Twist-routing and deflection are described in Section 2.5.

2.1. The Model

The model of the faulty on-chip routing is a mesh, where routers are placed on each grid points, and links are available between adjacent routers. Each routers can be good or bad, and each links can be healthy or faulty bidirectionally. A bad router is modeled by disabling all of its four links. In modern chips, packets are splited into flits, and routed from source node to the destination. In the routing algorithm, each router

accepts input flits from all nearby healthy links, permute them according to some rules, and send them back to all nearby healthy links. Because links are bidirectional, there are as many output links as input links, so all flits can go somewhere after the routing.

2.2. The Maze-Routing Algorithm

The Maze-routing add a header to each flits, containing some metadata of this flit. They are src, the source; dst, the destination; md_{best}, the closest Manhattan distance to dst that the packet has reached so far assuming a fault-free mesh; $mode$, being one of $greedy$, clockwisely face-routing ($\curvearrowright$), or counter-clockwisely face-routing ($\curvearrowleft$); n_{trav} and dir_{trav}, the node and direction which indicates the destination is unreachable if it is visited again.

In Maze-routing, each flit is routed to a productive and healthy output if possible. This is called the $greedy$ mode. If there is no such output, the flit changes itself into face-routing mode (randomly chosen from $\curvearrowright$ and $\curvearrowleft$). In face-routing mode $\curvearrowright$, the flit takes the first healthy output on the left of the ray from cur to dst, and then goes clockwisely. In face-routing mode $\curvearrowleft$, the flit takes the first healthy output on the right of the ray, and then goes counter-clockwisely. Effectively, the flit traverses the face underlying the ray from cur to dst. The flit changes back to $greedy$ mode when it goes to a router that can forward it closer to its destination than the node where it entered face-routing mode, i.e., the md_{best} in header can be reduced by a neighbor link. If the md_{best} cannot decrease until the flit has traversed the whole face, which is detected by revisiting n_{trav} on the direction of dir_{trav}, then there is no path between src and dst. We can drop this flit, and report this failure to src using the same algorithm as needed.

2.3. The Use of Bounding Circles

Twist-routing is based on Maze-routing, with the extra usage of bounding circles. The bounding circle is always centered at the destination of the flit, and its radius is recorded in the header, namely *c*. Notice that in Maze-routing, once face-routing mode is chosen, the direction is fixed until the flit changes back into $greedy$ mode. In Twist-routing, we draw a bounding circle with $c = \alpha_0 \cdot md_{cur,dst}$ when a flit enters face-routing mode. If the flit is going to cross the boundary of the bounding circle, we reverse the direction of the flit ($\curvearrowright \leftrightarrow \curvearrowleft$), and enlarge the bounding circle (times it by α). Also, n_{trav} and dir_{trav} is set to the node where the reverse happens and the direction after being reversed. Then the flit finds all its way back to the beginning node of face-routing in the reversed face-routing mode, and goes on in the reversed direction. There are three cases for this flit:

Case 1: It is going to cross the other border of the bounding circle. The bounding circle is enlarged again, and the direction of the flit is reverse again and so on.

Case 2: Its mode changes back to $greedy$, and md_{best} decreases successfully.

Case 3: It appears at the n_{trav} again with the direction dir_{trav}. This indicates that there is no path from src to dst.

The differences between Twist-routing and the original Maze-routing are described in Figure 1. Through experiments, for better performance, we use $\alpha_0 = 1.5$ and $\alpha = 4$. We use these values in the our experiments.

2.4. Proofs of Being Faster

Maze-routing can be very bad in some cases (see Figure 2 for one example of such cases).

Assume the big tree contains n edges. Maze-routing randomly choose between two directions when entering face-routing mode. If Maze-routing chooses the good direction, the flit will reach the destination with 4 hops. If Maze-routing chooses the bad direction, the flit has to go to the big tree and goes all the way back, and takes $2n+10$ hops to reach the destination in total. In average, Maze-routing takes $n+7$ hops, which is $\Omega(n)$. In this example, Twist-routing chooses between two directions, too. One direction leads to 4 hops. If we take the other direction, the flit will goes back without entering the tree because of the use of the bounding circle, and takes 8 hops to reach its destination. On average, it takes 6 hops only.

In the previous example, the length of the optimal path m is a constant, but Maze-routing needs $\Omega(n)$ hops. So Maze-routing cannot be bounded by any expression of m. However, Twist-routing runs in $O(m^3)$ hops, which is asymptotically better than Maze-routing. Now we prove this bound by two theorems.

Theorem 1. *If the destination of a flit is reachable from the source, and m is the length of the optimal path of this flit, the radius of the largest bounding circle used by Twist-routing without deflection is no more than* $\max(\alpha^2 m, c_0)$, *where* c_0 *is the initial radius of the bounding circle.*

Proof. There is a case where we never enlarge the bounding circle, so the largest circle is the initial one, with radius c_0. Otherwise, we only enlarge the bounding circle to C_1 with radius αk only if we meet a boundary of the bounding circle C with radius k. Only if we first meet the other boundary of C later, we may meet the boundary of C_1, and enlarge the bounding circle again. So if we found an edge which leads to closer to destination within the bounding circle C with radius k, we will not meet the other boundary of C, and the radius of the bounding circle never exceeds αk. Assume that we use the bounding circles that $c \le m < \alpha c$. We want to prove the radius of the largest bounding circle never exceeds $\alpha^2 m$, and it is enough to show that it never exceeds $\alpha^2 c$. Then it is enough to show that in the bounding circle with radius αc, the face routing can always find an edge that goes closer to the destination. Supposing not, then we assume in the face routing step, we go through path p. The path p splits the bounding circle with radius αc into two parts, and exactly one of them is reachable from the source within the bounding circle of radius αc. In other words, the destination is unreachable from the source within the bounding circle with radius αc. But since the length of the optimal path from the source to the destination is m, the optimal path lays in the bounding circle of radius αc completely, *i.e.*, the destination

is reachable from the source within the bounding circle. That is a contradiction. □

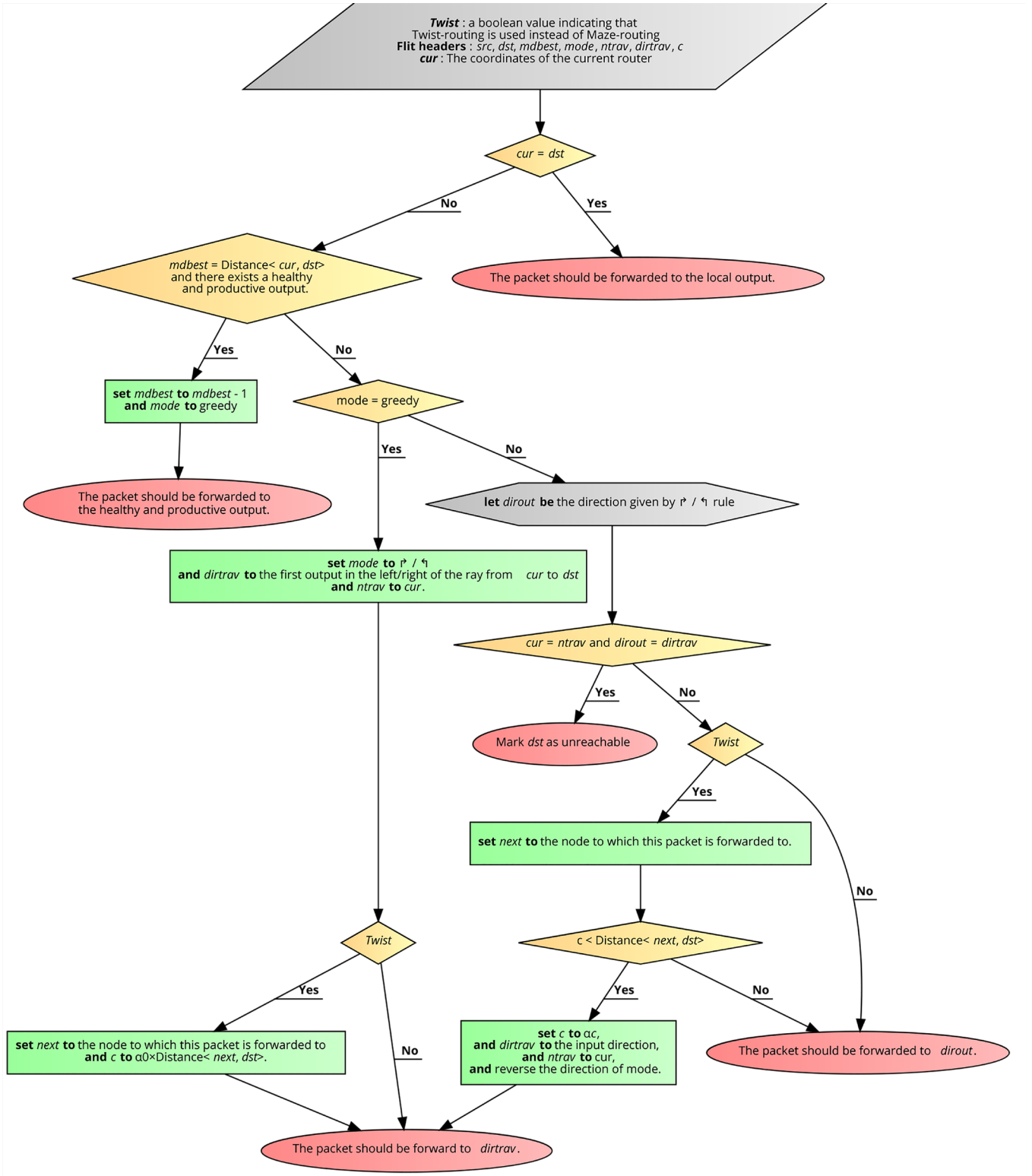

Figure 1. Maze-routing and Twist-routing algorithm.

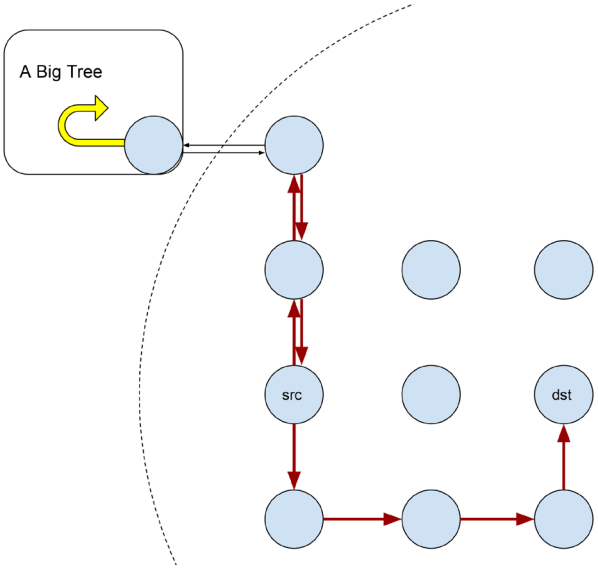

Figure 2. A setting where Twist-routing performs way better than Maze-routing.

Theorem 2. If the destination of a flit is reachable from the source, and m is the length of the optimal path of this flit, Twist-routing can find a path with length $O\left(m^3\right)$ for this flit without deflection.

Proof. Twist-routing consists of face routing steps and greedy routing steps. A greedy step reduce the md_{best} by one[1], and take one hop. A face routing step reduce the md_{best} by one, and take $O\left(m^2\right)$ hops. To prove this, notice that in a face routing step, when the bounding circle is fixed, we use each edge at most two times due to the properties of face routing. So when the bounding circle has fixed radius *k*, we need hops proportional to the total edges in the bounding circle, which is at most $O\left(k^2\right)$ hops. Since the radius of the largest bounding circle $c_{\max}$ satisfies

$$c_{\max} \le \max\left(\alpha^2 m, c_0\right) \le \max\left(\alpha^2 m, \alpha_0 m\right) = O\left(m\right) \tag{1}$$

and each time we enlarge the bunding circle exponentially, the total hops of one face routing step are

$$O\left(m^2 + \left(\frac{m}{\alpha}\right)^2 + \left(\frac{m}{\alpha^2}\right)^2 + \cdots + 1\right) = O\left(m^2\right) \tag{2}$$

Now consider that $0 \le md_{best} \le m$, and each reduction of md_{best} takes at most $O\left(m^2\right)$ hops, so all we need is $O\left(m^3\right)$ hops in total to transport this flit using Twist-routing. □

2.5. Deflection Implications

At one router, there are at most 4 input flits. Some flits have to be buffered or deflected

[1]Actually, the md_{best} decreases in the next greedy step instead of face-routing step, but since each face-routing step is always followed by a greedy step, we may regard the next greedy step as if it is part of face-routing step, and say face-routing step reduces the md_{best} by one.

if the dir_{out} of them are taken by other flits. If such case happenes, the flit may take a non-productive output, exit the face it is traversing, or be buffered and reappears in other input ports later. These behaviors result in inconsistency of the md_{best} and $mode$ values of the flit. So when a flit is deflected or buffered, its md_{best} are reset to $md_{next,dst}$, in which $next$ is the next router of this flit, and its $mode$ are reset to $greedy$. This makes the header and the state of this flit consistent.

To avoid deadlocks and licklocks, our algorithm needs to work with some deflection based mechanism proposed in literature. We mostly use minBD due to its high performance. The original method to avoid livelocks in minBD is to circularly make one flit golden for a long time *L*. However, in faulty chips, *L* needs to be at least as large as the longest path in the graph, which can be $O\left(n^2\right)$ large, where the chip is *n* by *n*. This renders the golden method to avoid locks not efficient. Instead of making one flit golden, we prioritize old flits to new flits globally to avoid livelocks. And we disable the buffer redirection in minBD because it is not compatible with our oldest-flit-based livelock-avoiding method[2].

3. Simulations

We compared Twist-routing algorithm with the original Maze-routing using an ad-hoc simulator[3]. Note that in Maze-routing, flits are independent to each other, and multiple flits are assembled to the original packet when received. For simplicity, we assume there is only on flit per packet in our simulator. We implements Maze-routing and minBD deflection method with buffer size equals to 4 in our simulator. Meanwhile, we implemented Twist-routing with minBD, too. In both algorithms, we use oldest-flit based livelock-avoiding method and without buffer redirection.

In order to compare the performance of two algorithms, we computed the average flit latency in the network under different injection rates using a uniform traffic[4]. We use 32×32 networks for evaluation. We use Erdös-Rényi model to generate faulty links, where the failure rate of any edge is 0.1 or 0.3. We generate 5 faulty chips for each case, and compute the average result across them. For each case, we run the simulations for 1000 cycles.

In a typical setting, the distances to deflect clockwisely or to deflect counter-clock wisely can be so much different. By backtracing and trying the other direction when running away from the bounding circle, our algorithm should provide better performance than the origin Maze-routing Algorithm. The simulation result shows the correctness of this conclusion. After careful measurement, in the case when the failure rate equals to 0.3, and the injection rate is 0.003, Twist-routing is 35% faster than Maze-routing. When the injection rate increases, Twist-routing keeps being fast (see **Figure 3** for details of all results).

[2]When the buffer redirection is enabled, we cannot avoid redirecting the oldest flit into the buffer, because the local information is not enough for us to determine if a flit is globally oldest or not. If the oldest flit enters the buffer, the delivery guarantee will be broken.

[3]The source code is available at https://bitbucket.org/aceeca1/twist-routing/src/master/src/.

[4]Flits that cannot reach its destination are dropped with no contribution to the average flit latency.

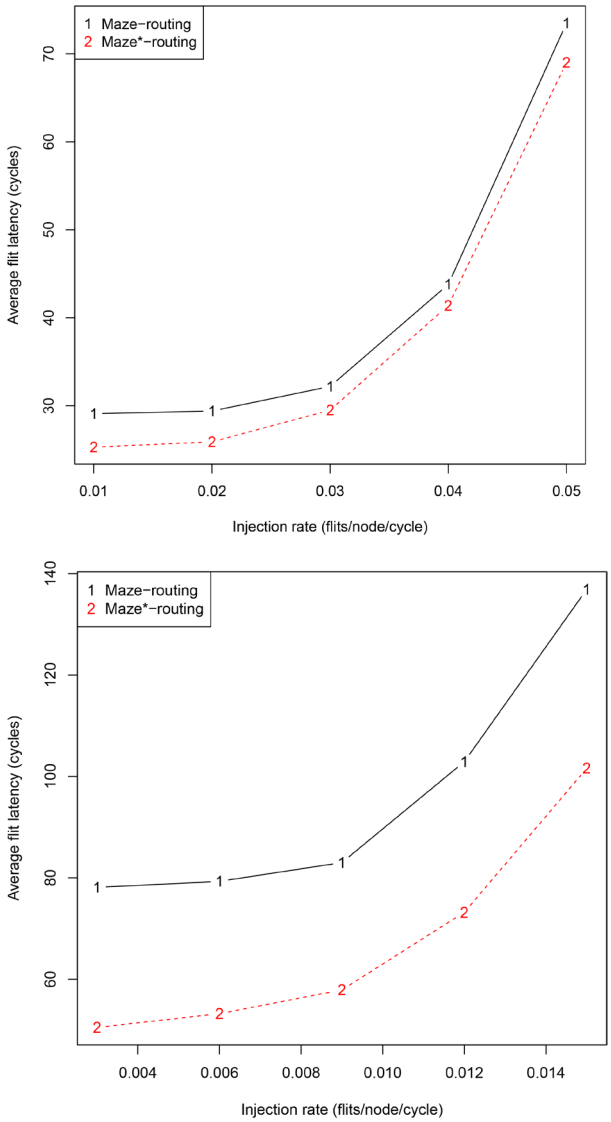

Figure 3. Simulation results with failure rate = 0.1, 0.3.

4. Conclusions

We improved Maze-routing for faulty on-chip networks, and yielded Twist-routing algorithm. We found that through the use of bounding circle and zigzag routes, the performance of our routing algorithm is better than Maze-routing, which is the state-of-the-art algorithm for faulty on-chip networks. Also, we provided a theoretical bound on our algorithm, which is not possible in previous works.

We conclude that our improvements on Maze-routing algorithm provides a better routing algorithm for faulty on-chip networks.

Acknowledgements

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of

China Grant 61033001, 61361136003.

References

[1] Borkar, S. (2007) Thousand Core Chips: A Technology Perspective. *Proceedings of the* 44*th Annual Design Automation Conference*, San Diego, 4-8 June 2007, 746-749. http://dx.doi.org/10.1145/1278480.1278667

[2] Dally, W.J. and Towles, B. (2001) Route Packets, Not Wires: On-Chip Interconnection Networks. *Proceedings of the IEEE* 2001 *Design Automation Conference*, Las Vegas, NV, USA, 18-22 June 2001, 684-689.

[3] Howard, J., Dighe, S., Hoskote, Y., *et al.* (2010) A 48-Core ia-32 Message-Passing Processor with dvfs in 45 nm cmos. *IEEE International Solid-State Circuits Conference* (*ISSCC* 2010), *Digest of Technical Papers,* San Francisco, CA, USA, 7-11 February 2010, 108-109.

[4] Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C.-C., Brown III, J.F. and Agarwal, A. (2007) On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, **27**, 15-31. http://dx.doi.org/10.1109/MM.2007.4378780

[5] Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Bao, L.W., Brown, J., *et al.* (2008) Tile64-Processor: A 64-Core soc with Mesh Interconnect. 2008 *IEEE International Solid-State Circuits Conference* (ISSCC 2008), *Digest of Technical Papers*, San Francisco, CA, USA, 3-7 February 2008, 88-598.

[6] Vangal, S.R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S., *et al.* (2008) An 80-Tile Sub-100-w Teraflops Processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, **43**, 29-41.

[7] Borkar, S. (2005) Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, **25**, 10-16. http://dx.doi.org/10.1109/MM.2005.110

[8] Nightingale, E.B., Douceur, J.R. and Orgovan, V. (2011) Cycles, Cells and Platters: An Empirical Analysis of Hardware Failures on a Million Consumer pcs. *Proceedings of the Sixth Conference on Computer Systems*, Salzburg, Austria, 10-13 April 2011, 343-356.

[9] Constantinides, K., Plaza, S., Blome, J., Zhang, B., Bertacco, V., Mahlke, S., Austin, T. and Orshansky, M. (2006) Bulletproof: A Defect-Tolerant cmp Switch Architecture. *The Twelfth International Symposium on High-Performance Computer Architecture*, Austin, Texas, 11-15 February 2006, 5-16.

[10] Fick, D., DeOrio, A., Hu, J., Bertacco, V., Blaauw, D. and Sylvester, D. (2009) Vicis: A Reliable Network for Unreliable Silicon. *Proceedings of the* 46*th Annual Design Automation Conference*, San Francisco, CA, 26-31 July 2009, 812-817. http://dx.doi.org/10.1145/1629911.1630119

[11] Kim, J., Nicopoulos, C., Park, D., Narayanan, V., Yousif, M.S. and Das, C.R. (2006) A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks. *Proceedings of the* 33*rd annual international symposium on Computer Architecture*, Boston, MA, USA, 17-21 June 2006, 4-15.

[12] Parikh, R. and Bertacco, V. (2013) udirec: Unified Diagnosis and Reconfiguration for Frugal Bypass of Nocfaults. *Proceedings of the* 46*th Annual IEEE/ACM International Symposium on Microarchitecture*, Davis, CA, USA, 7-11 December 2013, 148-159. http://dx.doi.org/10.1145/2540708.2540722

[13] Aisopos, K., DeOrio, A., Peh, L.-S. and Bertacco, V. (2011) Ariadne: Agnostic Recon-Figuration in a Disconnected Network Environment. 2011 *International Conference on Parallel Architectures and Compilation Techniques* (*PACT*), Galveston Island, Texas, USA, 10-14

October 2011, 298-309.

[14] Iordanou, C., Soteriou, V. and Aisopos, K. (2014) Hermes: Architecting a Top-Performing Fault-Tolerant Routing Algorithm for Networks-on-Chips. 2014 32*nd IEEE International Conference on Computer Design* (*ICCD*), Seoul, Korea (South), 19-22 October 2014, 424-431.

[15] Dally, W.J. and Seitz, C.L. (1986) The Torus Routing Chip. *Distributed Computing*, **1**, 187-196. http://dx.doi.org/10.1007/BF01660031

[16] Moscibroda, T. and Mutlu, O. (2009) A Case for Bufferless Routing in On-Chip Networks. *Proceedings of the* 36*th Annual International Symposium on Computer Architecture*, Austin, Texas, USA, 20-24 June 2009, 196-207. http://dx.doi.org/10.1145/1555754.1555781

[17] Fallin, C., Craik, C. and Mutlu, O. (2011) Chipper: A Low-Complexity Bufferless Deflection Router. 2011 *IEEE* 17*th International Symposium on High Performance Computer Architecture* (*HPCA*), San Antonio, TX, USA, 12-16 February 2011, 144-155. http://dx.doi.org/10.1109/hpca.2011.5749724

[18] Fallin, C., Nazario, G., Yu, X.Y., Chang, K.-P., Ausavarungnirun, R. and Mutlu, O. (2012) Minbd: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect. 2012 *Sixth IEEE/ACM International Symposium on Networks on Chip* (*NoCS*), Copenhagen, Denmark, 7-11 May 2012, 1-10.

[19] Fattah, M., Airola, A., Ausavarungnirun, R., Mirzaei, N., Liljeberg, P., Plosila, J., Mohammadi, S., Pahikkala, T., Mutlu, O. and Tenhunen, H. (2015) A Low-Overhead, Fully-Distributed, Guaranteed-Delivery Routing Algorithm for Faulty Network-on-Chips. *Proceedings of the* 9*th International Symposium on Networks-on-Chip*, Vancouver, 28-30 September 2015, 18.

[20] Kuhn, F., Wattenhofer, R. and Zollinger, A. (2002) Asymptotically Optimal Geometric Mobile Ad-Hoc Routing. *Proceedings of the* 6*th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Atlanta, GA, USA, 28 September 2002, 24-33. http://dx.doi.org/10.1145/570810.570814

[21] Kuhn, F., Wattenhofer, F., Zhang, Y. and Zollinger, A. (2003) Geometric Ad-Hoc Routing: of Theory and Practice. *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, Boston, MA, USA, 13-16 July 2003, 63-72. http://dx.doi.org/10.1145/872035.872044

[22] Kuhn, F., Wattenhofer, R. and Zollinger, A. (2003) Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. *Proceedings of the* 4*th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Annapolis, MD, USA, 1-3 June 2003, 267-278.

[23] Bose, P., Morin, P., Stojmenović, I. and Urrutia, J. (2001) Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, **7**, 609-616. http://dx.doi.org/10.1023/A:1012319418150