2014 17th Euromicro Conference on Digital System Design

Fault-Tolerant Irregular Topology Design Method for Network-on-Chips

Suleyman Tosun, Vahid Babaei Ajabshir, Ozge Mercanoglu
Computer Engineering Department, Ankara
University, Golbasi, Ankara, Turkey
Email: {stosun,vahid.babaei,omercanoglu@ankara.edu.tr}

Ozcan Ozturk
Computer Engineering Department, Bilkent
University, Bilkent, Ankara, Turkey
Email: ozturk@cs.bilkent.edu.tr

***Abstract*—As the technology sizes of integrated circuits (ICs) scale down rapidly, current transistor densities on chips dramatically increase. While nanometer feature sizes allow denser chip designs in each technology generation, fabricated ICs become more susceptible to wear-outs, causing operation failure. Even a single link failure within an on-chip fabric can halt communication between application blocks, which makes the entire chip useless. In this study, we aim to make faulty chips designed with Network-on-Chip (NoC) communication usable. Specifically, we present a fault-tolerant irregular topology generation method for application specific NoC designs. Designed NoC topology allows a different routing path if there is a link failure on the default routing. We compare fault-tolerant topologies with regular fault-tolerant ring topologies, and non-fault-tolerant application specific irregular topologies on energy consumption, performance, and area using multimedia benchmarks and custom-generated graphs.**

***Index Terms*—Fault tolerance, Network-on-Chip, topology, energy**

I. Introduction

Technology improvements have made it possible to place millions of transistors on a single chip, resulting in more complex and denser designs than ever. Now, designers can embed all system components on one chip, which is called System-on-Chip (SoC). However, this rapid increase of the number of components on chips has made current bus-based and point-to-point-based communication methods inefficient as a result of low performance and synchronization problems among components. At the beginning of the millennium, researchers introduced a better and scalable on-chip communication method, called Network-on-Chip (NoC) [1].

The NoC architectures can be constructed using regular or irregular topologies. Although regular topologies are easy to construct and reusable, applications cannot be well optimized on them. Irregular topologies are designed to be application specific, which allows optimizing power consumption, performance, and area [2]. Several studies have been published regarding energy-efficient and/or fault-tolerant regular topology-based NoC designs, especially for mesh topologies [3]. However, studies of irregular application specific topologies are restricted to exploring energy efficiency [4]; fault tolerance has not been considered.

Generated topologies using current application specific topology generation methods have only one communication path between any communicating nodes. If there is a permanent fault in any of the links or ports as a result of the fabrication process, the system cannot recover its functionality and the chip becomes useless. Motivated by this fact, in this study we propose a fault-tolerant application specific topology generation method for NoC-based designs. In our method, we first generate random non-fault-tolerant irregular topologies. We then add extra links to the generated topologies to make it fault tolerant. As an output, our method obtains several isomorphic fault-tolerant topologies. The designer can select any topology from the topology library that suits his/her design objectives.

We tested effectiveness of our method against ring and non-fault-tolerant irregular topologies based on energy consumption, area, and performance on both randomly generated graphs and multimedia benchmarks. Our results show that with very small area overhead, our method obtains fault-tolerant topologies.

II. Problem Definition

Our goals for the topology generation problem are 1) to determine a topology such that all communicating cores of the application can transmit data to each other over the network with at least two alternative paths and 2)to minimize energy consumption. To achieve these goals, the number of routers for the system has to be determined. Then, the resultant topology must ensure that each router can be reached from all other network routers via at least two paths and that all the cores are connected to at most one router port. Additionally, the routing must be deadlock and network congestion free (i.e., the router port and link bandwidth requirements must be satisfied.) To explain this problem more formally, we give the following definitions:

Definition 1 *A Core Flow Graph* (CFG) is a graph $G(V, E)$, where each vertex $v_i \in V$ represents a core (i.e., a node) in the application, and each edge $e_{i,j} \in E$ represents a dependency between two tasks v_i and v_j. The amount of data transfer between v_i and v_j is represented by weight

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 112E360 and EU COST Actions IC1204 - TRUDEVICE and IC1103 - MEDIAN.

978-1-4799-5793-4/14 $31.00 © 2014 IEEE
DOI 10.1109/DSD.2014.13

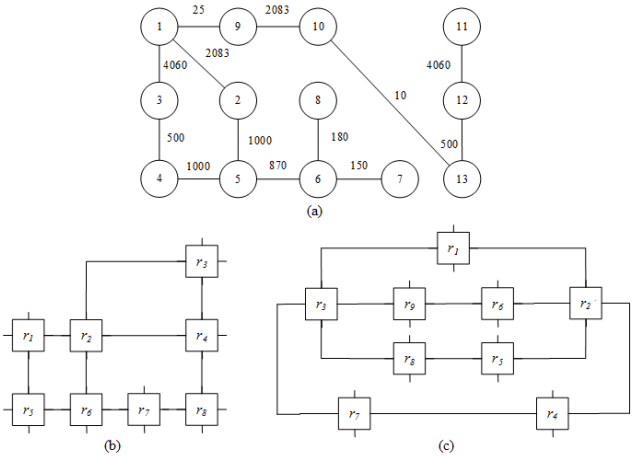


Fig. 1. (a) CFG of MP3 encoder, (b)-(c) two examples of a fault-tolerant TG.

$w_{i,j}$ for all $e_{i,j}$ and is given in bits per second. In Fig. 1.(a), we give the CFG of MP3 encoder.

Definition 2 *A Topology Graph (TG)* is a connected graph $T(R, L)$ where R represents the set of routers and L represents the set of links connecting the routers. In Fig. 1.(b) and (c), we give two examples of a TG. In both topologies, two alternative paths between any router pairs exist. In the first topology, we use eight routers and nine links, whereas in the second we use nine routers and 11 links. In our topologies, each link connected to a router port is assumed to be bidirectional (i.e., each port can be used as input or output.)

Average path length (APL) of the network affects the system's total communication cost. Thus, we try to minimize the APL in the generated topology.

Definition 3 Average path length APL_T of a topology T is the average of the shortest paths between any pairs of the vertices of the topology graph. Let r denote the number of vertices of the given topology. Then, the average path length APL_T is calculated by the following formula:

$$APL_T = \frac{2}{r(r-1)} \sum_{r_i \leq r_j} d(r_i, r_j). \quad (1)$$

For example, the APL of the graphs in Fig. 1.(b) and (c) are 1.92 and 1.86, respectively.

Problem: Fault-Tolerant Topology Generation (FTTG) Given a set of nodes (n) and the set of routers, each having p ports, determine the number of routers (r) and the number of links (l) for the topology. Then generate the topology that meets the following criteria:

- The topology must be fully connected with the set of routing paths P, where each path $p_{i,j}$ is the routing path between each pair of routers (r_i, r_j).
- For each path $p_{i,j}$, each link $l_{k,l}$ on this path should satisfy the bandwidth constraint $bw(l_{l,k})$.
- Additionally, to satisfy the fault-tolerance criteria, there must be at least two alternative routing paths between any router pairs. That is:

$$\forall (r_i, r_j) \in R, (p_{i,j}) \geq 2. \quad (2)$$

- The objective function of the FTTG is to minimize the APL of the generated topology T. In other words, our objective function is:

$$\min : APL_T. \quad (3)$$

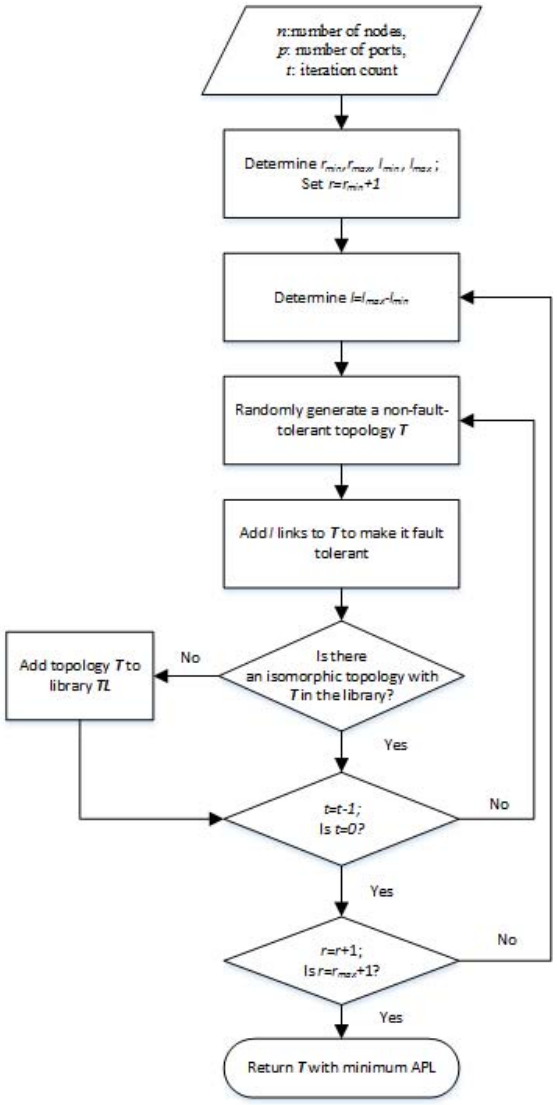


Fig. 2. Flowchart of FTTG algorithm.

III. Fault Tolerant Topology Generation (FTTG) Algorithm

We give the flowchart of the FTTG algorithm in Fig. 2. Our algorithm has two main phases: 1) generating non-fault-tolerant irregular topology using a minimum number of routers and links, and 2) adding extra routers and links to obtain a fault-tolerant version of the topology.

As shown in the flowchart, our method accepts the number of nodes (n) of the given application (CFG), the

number of ports (p) for routers, and the iteration count (t) as inputs. Based on these input values, it first calculates the minimum number of routers (r_{min}) and links (l_{min}) for non-fault-tolerant topology generation (N-FTTG) and the maximum number of routers (r_{max}) and links (l_{max}) that will be used for fault-tolerant topologies using Equations (4) - (7), respectively. Since the fault-tolerant irregular topology must utilize more than r_{min} routers, we start FTTG with $r_{min} + 1$ routers. At each outer loop of the FTTG algorithm, we add one more router to the routers at hand until we reach r_{max}.

$$r_{min} = \left\lceil \frac{n-2}{p-2} \right\rceil \tag{4}$$

$$l_{min} = r_{min} - 1 \tag{5}$$

$$r_{max} = \lceil r_{min} + \lg(r_{min}) \rceil \tag{6}$$

$$l_{max} = \left\lfloor \frac{pr_{max} - n}{2} \right\rfloor \tag{7}$$

After selecting the number of routers for the topology, we determine the number of links that can be added to the network by determining how many empty ports must be left for the application nodes, which is n here. We then generate a random, fully connected, non-fault-tolerant topology with r routers and $r-1$ links. Certainly, some of the routers must be connected to other routers with at most one port. Thus, we connect these routers to each other by adding $l = l_{max} - l_{min}$ links, aiming to minimize the APL of the topology. Each router and link must be on a cycle to have at least two alternative routing paths, thus our next step is to check that this is so. If there is a fault-tolerant topology with r ports in the topology library, we check whether the newly generated topology is isomorphic with the existing topology. If it is, we simply discard the new topology; otherwise, we add it to the library. This topology generation process iterates t times, which is a predefined iteration count.

As an output, in our library we may have several topology alternatives with different numbers of routers, varying between r_{min} and r_{max}. The designer can select any of these topologies that suits the design objectives, or the one with the minimum APL.

The run-time complexity of FTTG algorithm is dominated by the graph isomorphism algorithm [5], which is $O(r^5)$. The outer loop of FTTG algorithm runs $\lg r$ times and each inner loop iterates t times. Therefore, the time complexity of FTTG can be approximated as $O(tr^5 \lg r)$.

IV. Experimental Results

In this section, we evaluate the FTTG algorithm by comparing the topologies generated by FTTG with the ones generated by N-FTTG and ring. In the first set of experiments, we compare the FTTG algorithm based on APL and area. In the second set of experiments, we use real multimedia benchmarks to compare APL, area, energy, and performance (i.e., average hop count (AHC)).

A. Evaluating FTTG

In this set of experiments, we generate topologies using FTTG, N-FTTG, and ring or applications with different numbers of nodes, n. We select the iteration count $t = 500$ and the number of ports as four, five, and eight. Due to space concerns, we only give the results for topologies generated using four port routers. We conduct experiments with node numbers between eight and 100.

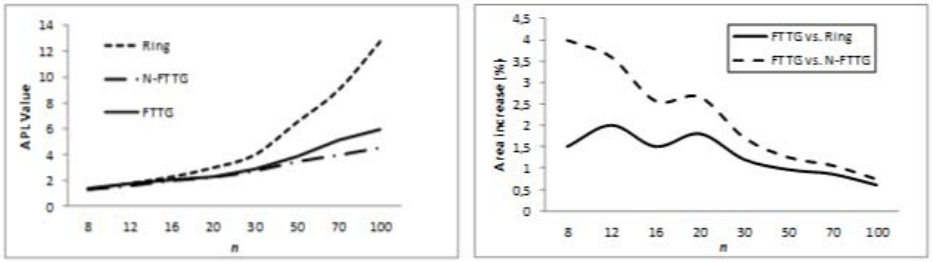
In Fig. 3, we illustrate how the APL values and percentage of area-increase scale with varying numbers of nodes for topologies that use four port routers. In Fig. 3.(a), we give the APL comparison for the ring, N-FTTG, and FTTG. We select the topologies with the best APL values for N-FTTG and FTTG. As the APL values for the three types of topologies show, for a small number of nodes, our FTTG algorithm determines topologies with similar APL values to the N-FTTG and ring alternatives. After the number of nodes exceeds 30, the N-FTTG determines better APL values than its counterparts.

We give the area overhead in percentages in Fig. 3.(b), comparing the area increase against ring and N-FTTG topologies. As the graph shows, the area overhead is within tolerable limits. In the area comparison, the fluctuation in part of the function is because the selected number of routers increases for varying n values. We select the maximum number of routers using Equation (6), and because the $\lg(r_{min})$ increases when the number of nodes increases, fluctuations occur. As the graph in Fig. 3.(b) shows, when the number of nodes increases, the area overhead decreases. For large numbers of nodes, the APL value increase of FTTG compared to N-FTTG is within tolerable limits. However, FTTG brings a fault tolerance capability with a small area and APL increase. With diminishing technology size, we can expect that future applications will be much higher than now, therefore, our FTTG method will be much more effective in the future while it still meets the current fault-tolerant topology needs.

B. Evaluating Energy Consumption

In this set of experiments, we evaluate the FTTG generated topologies based on energy consumption. We used a simulated annealing (SA) based mapping algorithm to map the given application on the generated topologies. To test the mapping, we select six video applications from the literature as benchmarks, namely the Video Object Plane Decoder (VOPD) and the MPEG-4 decoder from [6], the Multi-Window Display (MWD) from [4], and the 263 Decoder (263 Dec.), 263 Encoder (263 Enc.), and MP3 Encoder (MP3 Enc.) from [7].

For this set of experiments, we use four port routers. After determining the number of routers and links, we generate three topology alternatives. For the N-FTTG and FTTG topologies, we select the one with a minimum APL



(a) (b)

Fig. 3. Comparison of FTTG with ring and N-FTTG. (a) APL value comparison with varying numbers of nodes. (b) Area-increase percentages of FTTG against ring and N-FTTG.

TABLE I
ENERGY, AREA, AND LATENCY COMPARISONS FOR RING, N-FTTG, AND FTTG.

Graph	n	Energy (mJ)			Energy compare (%): FTTG vs.		Area Overhead (%): FTTG vs.		Average Hop Count		
		Ring	N-FTTG	FTTG	Ring	N-FTTG	Ring	N-FTTG	Ring	N-FTTG	FTTG
Mpeg-4	12	2.70	2.84	2.92	8.15	2.83	1.00	2.40	1.23	1.23	1.15
VOPD	16	2.94	2.64	3.09	5.23	17.25	0.75	1.71	1.20	1.05	1.15
MWD	12	0.83	0.97	0.87	4.90	-10.46	1.00	2.40	0.58	0.83	0.67
263 Dec	14	11.41	8.92	11.49	0.70	28.72	0.86	2.00	0.93	0.73	1.00
263 Enc	12	155.12	170.09	154.79	-0.21	-8.99	1.00	2.40	1.00	0.83	0.75
MP3 Enc	13	9.95	8.61	10.08	1.37	17.10	1.71	3.00	0.69	0.69	0.77

value. We then map the applications onto the generated topologies using our SA method. In the mapping process, we aim to minimize only the dynamic energy consumption of the network components (i.e., the total energy consumption of sending data over routers and links). We use the well-accepted energy model presented in [8] to calculate the energy consumption. In this model, the energy consumption of one bit between two communicating nodes is calculated by summing the energy consumed on router ports and links. For energy consumption parameters, we adopt the energy model for 100-nm technology given in [7]. In this model, the energy consumption of the routers is estimated at 328 nJ/Mb and 65.5 nJ/Mb for the input and output ports, respectively. In addition, the link energy consumption is estimated at 79.6 nJ/Mb/mm.

We present the results of these experiments in Table I. In the first two columns, we give the name of the graph and the number of nodes for the given graph, respectively. Columns three, four, and five give the energy consumptions of the mappings for the ring, N-FTTG, and FTTG, respectively. Columns six and seven show the energy comparison improvement of FTTG against the ring and N-FTTG, respectively. The next two columns show the area overhead of the FTTG topologies against the ring and N-FTTG topologies. Finally, the last three columns show the AHC value to compare the latency for the three mappings.

As the energy values in Table I show, our FTTG and the mappings obtain better results than the ring and N-FTTG most of the time. The area increases against ring and N-FTTG are around 1% and 2%, respectively. The AHC values for all three mappings are very close. As this set of experiments on real benchmarks demonstrates, our FTTG algorithm brings a fault tolerance capability to NoC design, with only a small area overhead and with better energy values than N-FTTG.

V. CONCLUSIONS

In this paper, we present a fault-tolerant application specific topology generation algorithm. Our algorithm generates topologies such that each router of the topology can be reached from any router with at least two alternative paths. The generated topology can be used to tolerate at least one link failure by applying the packet's alternative routings. We compare our method with ring and non-fault-tolerant topologies and show that with only a small increase in area, our method brings fault-tolerance capability to NoC designs.

REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," Proc. of DAC'01, pp. 684-689, 2001.
[2] S. Tosun, Y. Ar, and S. Ozdemir, "Application-specific topology generation algorithms for network-on-chip design," Computers & Digital Techniques, IET, vol. 6, no. 5, pp. 318-333, 2012.
[3] S. Tosun, "New heuristic algorithms for energy aware application mapping and routing on mesh-based nocs," Journal of Systems Architecture, vol. 57, no. 1, pp. 69-78, 2011.
[4] K.-C. Chang and T.-F. Chen, "Low-power algorithm for automatic topology generation for application-specific networks on chips," Computers & Digital Techniques, IET, vol. 2, no. 3, 2008.
[5] A. Dharwadker and J. Tevet, "The graph isomorphism algorithm," Proceedings of the Structure Semiotics Research Group, 2009.
[6] M. Janidarmian, A. Khademzadeh, and M. Tavanpour, "Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based Network on Chip," IEICE Electron. Express, vol. 6, no. 1, pp. 1-7, Jan., 2009.
[7] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear-programming based techniques for synthesis of network-on-chip architectures," IEEE Trans. Very Large Scale Integr. Syst. 14, 4 (Apr. 2006).
[8] J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," Proc. of DATE'03, pp. 688-693, 2003.