A Project Report

On

Fault Tolerance in Application Specific NoC

By

Kanniganti Abhishek	**2013AAPS308H**
Parth Shah	**2013A3PS309H**

Under the supervision of

Dr. SOUMYA J

SUBMITTED IN PARTIAL FULLFILLMENT OF ECE/EEE F266: STUDY ORIENTED PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI HYDERABAD CAMPUS (March 2017)

ACKNOWLEDGMENTS

I would like to thank **Dr. SOUMYA J**, Assistant Professor in the Department of Electrical & Electronics Engineering, BITS-Pilani, Hyderabad Campus for her unwavering support and valuable suggestions. I would also like to thank the moral support given by her without which there wouldn’t be any progress in this project.

ABSTRACT

In recent times the number of transistors in a chip is growing at a rapid rate. Hence the communication between the modules on a chip cannot be handled effectively with the help of the traditional bus architecture. To overcome this difficulty the network-on-chip(NoC) design was introduced. Application specific network on chip designs are asymmetric. This design is vulnerable to faults that can choke the system and prevent it from achieving its purpose. This project aims at developing a fault tolerant topology for this network.

TABLE OF CONTENTS

1. INTRODUCTION

Recent advances in technology have enabled to integrate millions of transistors on a single chip, so much so that various components of a computer can now be integrated on a single chip. This is called a System-on-chip (SoC). For communication between the components, conventionally a bus topology has been used. But this can lead to saturation very fast and is not scalable. The alternative is called network-on-chip (NoC) where the bus is replaced by several smaller interconnections.

For generic NoC, symmetric topologies like the mesh are widely used. However for application specific NoC (ASNoC), irregular topologies can reduce hardware costs and improve performance. Chip networks are vulnerable to faults that can choke the system and render the system useless. This project aims at developing fault tolerant irregular topologies for application specific network on chip.

2. LITERATURE SURVEY

Fathollah Karimi Koupaei, Ahmad Khademzadeh and Majid Janidarmian[1] talk about using a special hardware called the link interface to tackle faults in a system on chip. They propose using two router for each core. One of the router is the main router, all the communication for that core takes place through that router until it fails. When a fault occurs in a router then the core starts using the spare router assigned to it. The spare router allotment is done using an algorithm which calculates the response time between the core and the other router and then selects the router with the minimum response time.

Suleyman Tosun, Vahid B. Ajabshir, Ozge Mercanoglu, and Ozcan Ozturk[2] present an algorithm for generation of a irregular link fault tolerant topology for Application Specific NoC designs. They discuss how designed NoC topology allows different routing path if there is a link failure on the default routing path. Additionally, they present a simulated annealing-based application mapping algorithm aiming to minimize total energy consumption of the NoC design. They compare fault-tolerant topologies with nonfault-tolerant application-specific irregular topologies on energy consumption, performance, and area using multimedia benchmarks and custom-generated graphs.

3. PROBLEM DEFINITION

In brief, given the specifications of the application, we need to formulate the optimum topology.

The input here is as follows:
A weighted graph where the nodes represent the components (or cores) of the system. The weight of the edge connecting two nodes represents the communication bandwidth required between the respective cores.

A weighted graph is a graph *G(N, E)* where each node $n_i \in N$ represents a core (i.e., a node) in the application, and each edge $e_{ij} \in E$ represents a communication between two nodes n_i and n_j. The amount of data transfer between n_i and n_j is represented by the weight w_{ij} for all e_{ij} and is given in bits per second. For eg.:

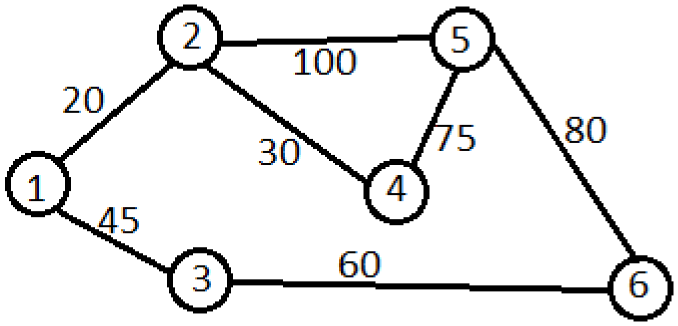

Figure 1: Sample input graph

The output is a network of routers arranged in the optimum fault tolerant topology and each router is mapped to a core. This can be represented as another graph T(R,L) where $r_i \in R$ represent the routers and $l_{ij} \in L$ represent the links between the routers. For eg.:

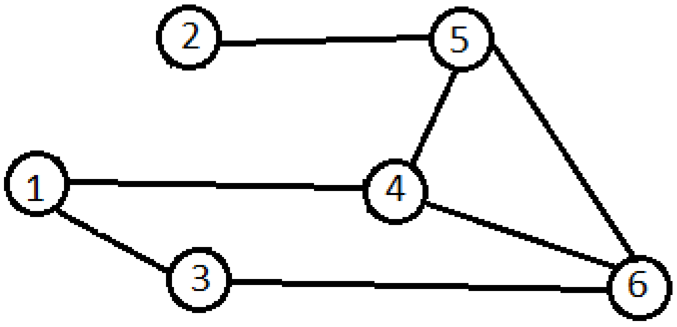

Figure 2: A sample topology

The optimal topology is deduced on the basis of the total communication cost C. For every edge e_{ij} in the weighted graph, the communication cost c_{ij} associated with it is the product of its weight w_{ij} and the number of hops from r_i to r_j (denoted by h_{ij}). The total communication can then be defined as a sum of costs c_{ij} associated with all the edges. This can be succinctly described in the two following equations:

$$c_{ij} = h_{ij} * w_{ij}$$

$$C = \sum c_{ij} \ (\forall\ e_{ij} \in E)$$

For the topology illustrated in figure 2, the total communication cost would be:

$c_{12} = h_{12} * w_{12} = 3 * 20 = 60;$ $c_{13} = h_{13} * w_{13} = 1 * 45 = 45;$
$c_{24} = h_{24} * w_{24} = 2 * 30 = 60;$ $c_{25} = h_{25} * w_{25} = 1 * 100 = 100;$
$c_{36} = h_{36} * w_{36} = 1 * 60 = 60;$ $c_{45} = h_{45} * w_{45} = 1 * 75 = 75;$
$c_{56} = h_{56} * w_{56} = 1 * 80 = 80;$

Total communication cost C $= 60 + 45 + 60 + 100 + 60 + 75 + 80$
$= 480$

In this context, fault tolerance can be classified in two:

a. Link fault tolerance: There must be at least two distinct paths between any pair of routers r_i and r_j. Thus even if a link (for example l_{ij}) connection fails, the communication between routers will not be affected.

b. Router fault tolerance: Each core must be connected to at least two routers. Thus even if a router fails the associated core will not be cut off from rest of the network.

4. NON FAULT TOLERANT TOPOLOGY

A strictly Non fault tolerant topology is generated here, for setting a benchmark for the minimum number of links and routers needed. Here, there are no alternate paths for any two given routers. Since the objective here is to minimize the total communication cost, we try to minimize the number of hops for the edges with highest weights (or bandwidth). The algorithm proposed to achieve this is as follows:

Step 1: Add a router r_i for each node n_i.
Step 2: Sort and arrange all edges e_{ij} in the descending order of the weight w_{ij} associated with the respective edges to create the list L(e_{ij}).
Step 3: For the first edge e_{ij} in L(e_{ij}), check if a path exists which connects routers r_i & r_j.
Step 4: If yes, proceed to the next step. Otherwise, connect the routers with a new link *lij*.
Step 5: Remove e_{ij} from L(e_{ij}) and if L(e_{ij}) is not yet empty, repeat from **Step 2**.
Step 6: Return the resultant topology.

This algorithm produces a graph which depicts the topology. This topology is strictly non fault-tolerant and minimizes the hardware cost.

For the sample input graph in figure 1, the topology would look something like this:

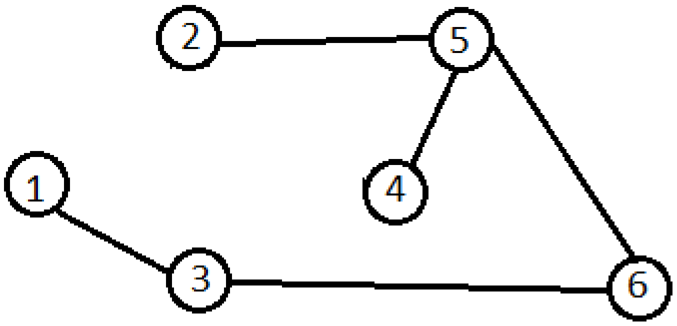

Figure 3: Non Fault tolerant topology

Communication cost C = 4 * 20 + 1 * 45 + 2 * 30 + 1 * 100
+ 1 * 60 + 1 * 75 + 1 * 80
= 500

5. NATIVE TOPOLOGY

The input graph itself can be viewed as a basic topology. The simple algorithm to create a topology that looks exactly like the input graph is as follows:

Step 1: Add a router r_i for each node n_i.
Step 2: For each edge e_{ij} create a link l_{ij} connecting routers r_i & r_j.
Step 3: Return the resultant topology.

This topology has the lowest communication cost theoretically possible as $h_{ij} = 1$ for all edges e_{ij}. Any topology that has this as a part of it, will have the same communication cost. Hence, this topology is intended to be the basis for the next algorithm. Here, we also set the benchmark for communication cost.

For the sample input graph in figure 1, the topology would look exactly like it.

The communication cost $= 20 + 45 + 30 + 100 + 60 + 75 + 80$
$= 410$

6. LINK UTILIZATION AND DEGREE

For the next topology generation algorithm, we will need concepts called link utilization and degree of a router. Here the definitions for these terms are provided.

Link utilization of a particular link l_{pq} is the number of edges e_{ij} in the input graph such that the shortest path between nodes n_i and n_j includes the link l_{pq}.

Degree of a router is the number routers it is connected to with a link.

7. POOREST NEIGHBOUR ALGORITHM

This algorithm aims to provide link fault tolerance. As mentioned earlier, it starts from the native topology described above. The algorithm is as follows:

Step 1: Generate the native topology.
Step 2: For each link l_{ij} in the above topology, check if there is an alternate path between the routers r_i and r_j.
Step 3: If yes, jump to **Step 2** for the next link.
Step 4: If no, determine the degree of routers r_i and r_j. Now we refer the router with the smaller degree as r_s and the larger degree as r_l.
Step 5: Consider the router r_l. For each router r_n that is connected to it, check the link utilization of l_{ln}. Select the router r_n which has link with the lowest link utilization.
Step 6: Connect routers r_n and r_s.
Step 7: Jump to **Step 2** for the next link.
Step 8: If no links are left, return the resultant topology.

This topology has the lowest possible communication cost as it contains the native topology as a part of it. The communication cost is the same as the native topology. Also, interestingly, the topology is also identical to the native topology as it is already fault tolerant.

8. POOREST NEIGHBOUR FAIL CASE: ISOLATED PAIR

The poorest neighbour algorithm fails to provide fault tolerance in the following scenario:

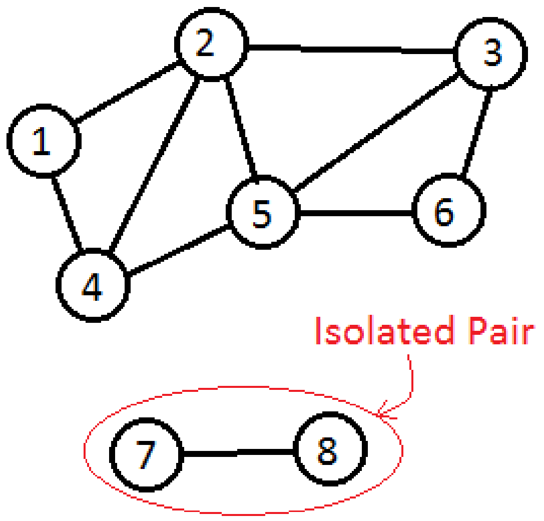

Figure 4: Graph with an isolated pair.

Consider a disconnected graph like the one shown above. Suppose this is the input to our algorithm. The 'poorest neighbour' algorithm first checks if the particular link has an alternate path for connecting the two end nodes of that link. If yes, then it does nothing. If no, then the algorithm seeks to connect one of the nodes with a neighbour of the other node. But in the above graph, consider the edge connecting nodes n_7 and n_8. None of these nodes have a neighbour. Hence, there is no one to connect either of the concerned nodes. No connection is thus made, and the resulting topology is not fault tolerance.

8.1. Proposed solutions

In this case, the proposed solution is to modify the algorithm to connect the nodes to the nodes in the rest of the graph with the lowest degree. Here, the nodes n_1 and n_6 are of degree 2. Thus, if we choose node n_6, the resulting topology would look something like this:

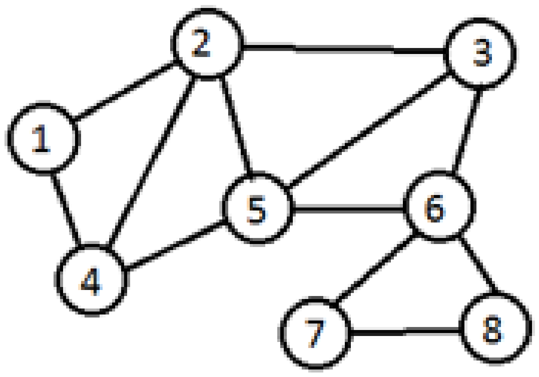

Figure 5: proposed solution

However, this increases the number of ports of the chosen node by 2. Thus, the degree of node n_6 now becomes 4. This would become a problem if the degree goes beyond the maximum number of ports of a router. For example, if the minimum degree in the above example was 3 instead of 2 and the maximum number of ports a router has is 4, it would look like this:

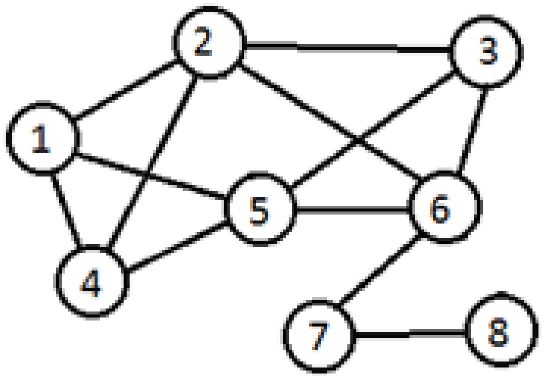

Figure 6: alternate scenario

Here, we can connect only one node to *n*$_6$. One solution would be to connect the other node to the poorest neighbour of *n*$_6$ which refers to the node *n*$_3$. Thus, the resulting fault tolerant topology would look like this:

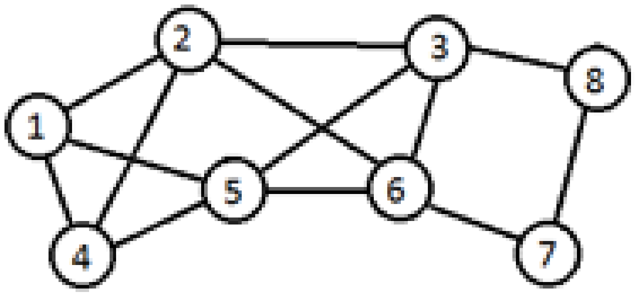

Figure 7: Proposed solution to the alternate scenario

9. CONCLUSION

The non-fault tolerant topology generation algorithm described in section 4 had the minimum hardware cost.

The native topology set the benchmark for communication cost. The poorest neighbour algorithm provided link fault tolerance while meeting the communication cost benchmark set by the native topology.

Thus in this project, we propose different algorithms for optimizing the communication cost, hardware costs and to provide fault tolerance.

REFERENCES

[1] Fathollah Karimi Koupaei, Ahmad Khademzadeh, Majid Janidarmian, “Fault-Tolerant Application-Specific Network-on-Chip”, Proceedings of the World Congress on Engineering and Computer Science 2011, Vol II, WCECS 2011, October 2011

[2] Suleyman Tosun, Vahid B. Ajabshir, Ozge Mercanoglu, and Ozcan Ozturk, “Fault-Tolerant Topology Generation Method for Application-Specific Network-on-Chips”, IEEE Transactions On Computer - Aided Design Of Integrated Circuits And Systems, vol. 34, no. 9, September 2015