ELectronics EXpress

IEICE Electronics Express, Vol.4, No.10, 332–339

Degradable mesh-based on-chip networks using programmable routing tables

Ali Shahabi[a)], **Nima Honarmand, Hassan Sohofi, and Zainalabedin Navabi**

CAD Laboratory, School of Electrical and Computer Engineering, University of Tehran, Tehran, IRAN

a) *shahabi@cad.ece.ut.ac.ir*

Abstract: The decreasing manufacturing yield of integrated circuits, as a result of rising complexity and decreased feature size, and the emergence of NoC-based design techniques, has necessitated the search for network reconfiguration techniques for reusing NoCs with faulty communication hardware. In this paper, we propose a method to cope with the problem of faulty communication links in mesh-based NoCs. The method is based on the use of programmable routing tables, which has a fixed number of entries, in network switches. Experimental results show that a network reconfigured for fault masking by programming its routing tables has acceptable but degraded performance parameters as compared to the non-faulty network.

Keywords: degradability, network on chip, programmable routing table

Classification: Integrated circuits

References

[1] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. P. Pande, C. Grecu, and A. Ivanov, "System-on-Chip: Reuse and Integration," *Proc. IEEE*, vol. 94, no. 6, pp. 1050–1069, June 2006.

[2] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[3] S. Kumar, A. Jantsch, M. Miliberg, J. Oberg, J.-P. Soininen, M. Forseli, K. Tiensyrja, and A. Hemani, "A Network on Chip Architecture and Design Methodology," *in Proc. ISVLSI'02*, pp. 117–124, 2002.

[4] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, Aug. 2002.

[5] R. Marculescu, "Networks-on-Chip: The Quest for On-Chip Fault-Tolerant Communication," *in proc. ISVLSI'03*, pp. 8–12, Feb. 2003.

[6] M. Yang, T. Li, Y. Jiang, and Y. Yang, "Fault-Tolerant Routing Schemes in RDT(2,2,1)/a-Based Interconnection Network for Networks-on-Chip Designs," *in Proc. ISPAN'05*, pp. 1–6, Dec. 2005.

[7] N. Honarmand, A. Shahabi, H. Sohofi, M. Abbaspour, and Z. Navabi,

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

"High Level Synthesis of Degradable ASICs Using Virtual Binding," *VLSI Test Symposium*, 2007, in press.
[8] C. Grecu, A. Ivanov, R. Saleh, E. S. Sogomonyan, and P. P. Pande, "On-line Fault Detection and Location for NoC interconnects," *in Proc. IOLTS'06*, pp. 145–150, 2006.
[9] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks—An Engineering Approach*, Morgan Kaufmann, 2002.

1 Introduction

System-on-chip (SoC) design methodologies [1] have provided the appropriate and integrated solutions to manage the ever-increasing complexity of state-of-the art digital and mixed-signal systems. But still communication between processing elements in an SoC is one of the most important challenges in SoC design methodology. Networks on Chips (NoCs) [2] have been proposed as a solution to this problem. Several NoC architectures and their implementation details have been presented in [3] and [4].

As the IC fabrication technologies move to nano-scale feature sizes and complexity and transistor count of the ICs grow, the produced dies become more and more susceptible to manufacturing faults and the process yield decreases. Because manufacturing faults tend to be local and affect only a limited area of the IC, it is possible (and very desirable) to find methods to make such faulty dies reusable. Such failures can occur in Processing Elements (PEs) and/or in communication fabric.

One possible method to work around the permanent faults is to use fault tolerant architectures, which has been addressed in [5] and [6]. Such techniques tend to introduce some redundancies into the circuit in order to cope with faulty elements and thus result in less-than-full hardware utilization.

Reconfiguring the faulty IC to avoid using the faulty modules (PEs, switches or links) and get the work done using the remaining non-faulty ones is another approach, that is usually referred to as degradability [7]. Generally, degradability-based methods imply a design and manufacturing flow including the following steps: First, at the design stage, special algorithms should be used which result in reconfigurable circuits. Then, after the IC has been built, diagnosis techniques [8] should be used to detect the faulty components of the circuit. Then, based on the obtained fault pattern, a proper configuration, which bypasses the faulty elements, should be chosen and programmed into the circuit. This implies that the circuit should inevitably have some programmable elements in it and post-manufacturing reconfiguration should be possible. But, this is by no means a great overhead because widely-used built-in provisions such as JTAG or scan chains can be used for this purpose. The reconfigured circuit is likely to have a degraded performance, compared to the non-faulty one, because of the decreased number of available resources.

In this paper, we focus on degradability-based methods for recovering a faulty communication fabric in networks with mesh [3] topology. Mesh-based

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

networks are appearing as a de facto standard in NoC realm because of their regular and efficient layout and simple routing mechanism.

2 Proposed Routing Mechanism

In an NoC, switches are responsible for routing the packets between nodes. Each switch has a set of bidirectional ports through which it is connected to neighboring switches or PEs. It also contains a router to define a path between input and output ports, buffers to store intermediate data and an arbiter to grant access to a given port when multiple input requests arrive in parallel.

One important aspect of the switch design is the implementation of the routing algorithm. It can be implemented as a hardwired module or as a Programmable Routing Table (PRT). In the hardwired style, no post-manufacturing programming is required but the switch cannot be changed to cope with the link failures. Thus, the chip will become unusable whenever some links go faulty. But, in the case of PRTs, the switch can be reconfigured to bypass the broken links and get the job done using the remaining links.

The paths taken by packets between source and destination switches, hence the contents of the PRTs, are defined by the routing algorithm in use. The algorithm must prevent deadlock, live-lock and starvation [9]. The live-lock refers to the situation in which some packets will not reach their destination, even if they never get blocked permanently.

Based on the particular topology and routing algorithm in use, there may be some methods to work around the above problems. We demonstrate such a technique for mesh topology, Fig. 1 (a), and its widely used XY routing algorithm.

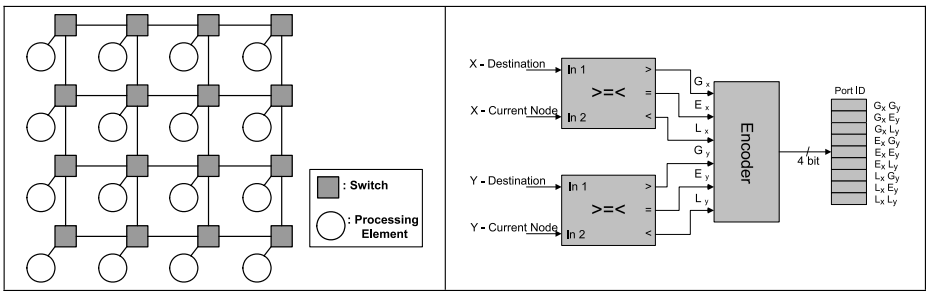


Fig. 1. (a) 2-D Mesh Topology, (b) Mesh-based routing mechanism

In mesh-based networks [3], the address of a node is a pair (x, y) which is the coordinates of the node in the mesh. Each mesh switch, in general, has five ports: one attached to the local PE (local port) and the other four to the neighboring switches (system ports). When a packet arrives, it should be delivered to the attached PE, through the local port, if it is destined for that node. Otherwise, one of the system ports should be selected according to the destination address. A simple live-lock free routing algorithm, called XY [9],

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

has been proposed for the mesh topology. In this algorithm, a packet is first routed in X direction: if the x-coordinate of the destination is less than that of the current node it will be routed to the left and if it is more than that of the current node it will be routed to right. Afterwards, the packet will be routed in the Y direction, *i.e.*, up or down, based on the y-coordinate of the destination address.

Based on this algorithm, Fig. 1 (b) shows the proposed routing mechanism. Two small comparators compare the x- and y-coordinates of the current node with those of the destination. The outputs of each comparator can assume three different one-hot coded states (G for greater, E for equal and L for less). Thus, we can have 9 different situations for the combination of comparator outputs. An encoder will generate a 4-bit signal indicating which of these 9 situations has occurred. The output of the encoder will be used to index a 9-entry, programmable lookup table. Each entry of the lookup table contains a port identifier to indicate one of the five ports that should be used. Apparently, the routing hardware has a fixed structure and does not depend on the number of nodes in the network. Throughout this paper, we refer to this method as Mesh-Based Routing (MBR) because it is tailored to the mesh topologies.

3 Programming MBR-based PRTs Under Link Failures

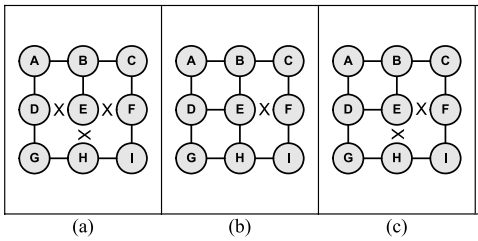
The network topology could be considered as a graph with switches being its vertices and links being its edges. If this graph is connected, then the network will be *structurally connected*. We say that the network is *routing-connected* if for every source and destination (SRC, DST) pair of nodes, a packet originated in SRC can be routed to reach DST. Whether this is possible or not depends on the routing table configuration of the nodes that the packet visits. Improper configurations might cause a live-lock and prevent the packet from reaching its destination. We call a network *routing-connectable* if there is a set of PRT configurations to render the network routing-connected.

3.1 Reconfiguration Procedure

It is possible for a network of MBR-based PRTs to be structurally connected but not routing-connectable. Figure 2 (a) shows an example of such a network. In this figure, the crosses indicate broken links. No configuration can be found for the MBR-based PRTs to create a routing-connected network. Because when B has a packet destined for E, it should send it using BE link and when it has a packet destined for H it should not use BE link. Thus, there is no feasible port identifier for (E_xG_y) entry of B's routing table.

When a packet arrives at a switch, either it is destined for the attached processing element or it should be routed using one of the four system ports. In what follows, we use L, R, D and U to indicate moving in left, right, down or up directions respectively. The decision on where to route the packet is based on the result of address comparison between current switch and the destination switch. Consider a packet which is currently at a switch addressed

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007



Node	Hard Constraints	Soft Constraints
B	D < R	D < L
C	D < L	-
D	R < D	R < U
E	-	R < U, D < L
F	-	L < U, L < D
G	R < U	-
H	-	U < L, U < R

(d)

Fig. 2. (a) a network which is not routing connectable using MBR (b) a network with one faulty link (c) a network with two faulty links (d) routing constraints in network of (c)

(x_{cur}, y_{cur}) and is destined for switch (x_{dst}, y_{dst}). We define the distance of the switches as

$$dist(cur, dst) = (|x_{dst} - x_{cur}| + |y_{dst} - y_{cur}|)$$

This distance is the minimum number of hops that the packet should traverse until it reaches its destination. If a routing algorithm routes a packet in such a way that its distance from the destination switch decreases with each move, the algorithm will be live-lock free. This is because a live-locked packet will visit a switch twice and this cannot happen with a decreasing sequence of distances. The conventional XY routing algorithm has this property and thus is live-lock free. We call every move that decreases the distance of packet from its destination a *positive* move. Otherwise, we call it a *negative* move.

Sometimes there are multiple positive moves for a packet. For example, if the destination is both to left and above the current switch, taking either direction will be a positive move. In this case, direction to choose depends on the routing table of the current switch. An MBR-based routing table has one entry for every possible combination of positive moves, as shown in Fig. 1 (b). The contents of this table indicate the relative priority of moving in different directions. For example, if in the entry corresponding to positive moves in up and left directions, $(L_x L_y)$ in Fig. 1 (b), the port identifier of the left port is given, then the priority of moving in the left direction is more than moving up. We use the "<" operator to show the priority of movements. In this case, we write U < L.

When a node becomes faulty, it will need some help from one of the neighboring nodes (helping node) to route the packet. Since the switch with a faulty link might be forced to perform a negative move, the distance may increase and if the helping node does not have its priorities properly set, it might return the packet to the original switch and cause live-lock. Hence, it would be necessary to put some constraints on the possible priority combinations that the helping node can use.

3.2 An Example with One Faulty Link

To demonstrate our technique, Fig. 2 (b) shows a mesh structure with one broken link between nodes E and F. Since the EF link is faulty, Node E

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

can select one of its three adjacent nodes (B, D, and H) as the helping node. Suppose that we choose node B. This will impose some restrictions on the routing table of B to prevent live-lock situations. Suppose node E has a packet destined for node F. Since the EF link is faulty, E will send the packet to B, instead. Now, node B has a packet that should move both right and down. If, in routing table of B, down moves have a higher priority than right moves, B will send the packet back to E and will cause a live-lock. Thus, for B, the priority of down move should be less than right move, or D < R. This is a Hard Constraint (HC) for B, *i.e.*, it must be met to have a live-lock free routing.

When we consider all the possible faulty links and extract all the required constraints, it might be the case that for a switch, the constraints are conflicting. For example, a switch might have both L < R and R < L constraints. Obviously, these are conflicting constraints and cannot be satisfied simultaneously. Suppose that in our example, we first choose B as the helping node of E and after considering other nodes, we get trapped in a conflicting situation. It might be the case that this conflict has happened because of choosing B as the helping node. Hence, we should now check the alternative choice and test the selection of H or D instead of B as the helping node. This means that our algorithm should have a backtracking nature and whenever it encounters some conflict, it should backtrack to the last point that it had an alternative choice and test that one.

Moreover, we can use another constraint type, called Soft Constraint (SC) to reduce the average length of paths in the mesh structure under faulty conditions. Suppose that in Fig. 2 (b), E is helped by node B to route in the right direction. It is better to force switch E to first route the packets using its non-faulty links (up, left and down) if it is a positive move. This means that switch E will use the helping node B to route when no other choice exists. Thus we consider the following SCs for node E: {R < U, R < D, R < L}. Since, moving to right and left are exclusive and cannot happen at the same time, the (R < L) constraint is useless and can be removed. As another example, Fig. 2 (c) shows a network with two faulty links and Fig. 2 (d) shows the assigned hard and soft constraints for each node. The nodes missing in the table have no constraints.

3.3 Reconfiguration Algorithm

Alg. 1 shows the reconfiguration algorithm. The ConfigureNetwork routine has two inputs: set of faulty links that have not been handled yet, and the constraints resulting from handling previous faults. It selects one faulty link from the list and, one by one, examines all possible helping nodes for the node affected by the faulty link. It repeats this process until there remains no faulty link in the list, *i.e.*, all the faults get handled, or at some point, the set of constraints could not be satisfied. In the former case, it checks all the routing table configurations that satisfy the constraints. If a configuration results in a live-lock free network, it will be returned as the result. Otherwise, the algorithm will backtrack to the point of the last choice. In the latter case,

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

```
CONFIGURENETWORK(failure_set, current_constraints)
begin
  if ISEMPTY(failure_set) then
    for each configuration conf which satisfies all the constraints
      test all possible (src, dst) node pairs for a live-lock free path;
      if the test succeeds then
        return conf;
      end if;
    end for;
    return NULL;
  else
    f = FIRST(failuer_set);
    n = node affected by f;
    for each possible helping node h for n
      new_constraints =
        current_constraints +
          {constraints from using h as the helper};
      if COULDBESATISFIED(new_constraints) then
        return
          CONFIGURENETWORK(failure_set - f, new_constraints);
      else
        return NULL;
      end if;
    end for;
  end if;
end
```

Alg. 1. Reconfiguration algorithm for MBR

the algorithm will consider alternative helping nodes for the node affected by current faulty link. If there is no unchecked alternative, the search fails and the algorithm will return NULL to indicate failure.

4 Experimental Result

To assess the proposed technique, we considered 10 examples 4 by 4 networks with 1 to 9 faulty links. The number of faulty links cannot exceed 9 because the network will become structurally unconnected. To measure the quality of obtained network configurations, we extracted the following parameters for each network: the average and maximum path length in the network and the average and maximum link load. Load of a particular link indicates the number of different (SRC, DST) pair of nodes whose packets should traverse that link. Table I presents the results obtained for MBR-based techniques. The row indicated with XY gives the performance parameters for a network with no faulty links in which all the routing tables are configured according to XY. The values in parentheses are normalized with regard to the corresponding values of non-faulty network.

As Table I indicates, the general trend of performance parameters is to get worse when the number of faulty links increases. However, there are some deviations. These deviations can be attributed to the sensitivity of the results to both the number and the pattern of faulty links. Ex.2 and Ex.3 demonstrate the sensitivity to the pattern of faulty links. In both cases there are 2 faulty links and only the positions of the links differ in two cases. Also, as expected, when the algorithm is able to incorporate SCs, the results will be better than when only HCs are considered. But in some cases, EX6 to

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007

Table I. Experimental Results

	# of Faulty Links	Hard + Soft Constraints				Hard Constraints			
		Longest Path	Average Path	Maximum Load	Average Load	Longest Path	Average Path	Maximum Load	Average Load
XY	0	6	2.66	28	13.3	6	2.66	28	13.3
EX1	1	6(1.00)	2.73(1.03)	32(1.14)	14.3(1.07)	7(1.17)	3.00(1.13)	48(1.71)	15.7(1.17)
EX2	2	7(1.17)	2.97(1.12)	32(1.14)	16.2(1.21)	8(1.33)	3.17(1.19)	44(1.57)	17.3(1.30)
EX3	2	8(1.33)	3.00(1.13)	32(1.14)	16.4(1.23)	8(1.33)	3.00(1.13)	32(1.14)	16.4(1.23)
EX4	3	7(1.17)	3.05(1.15)	32(1.14)	17.4(1.31)	8(1.33)	3.22(1.21)	44(1.57)	18.4(1.38)
EX5	4	8(1.33)	3.52(1.32)	40(1.43)	21.1(1.58)	8(1.33)	3.53(1.33)	46(1.64)	21.2(1.59)
EX6	5	-	-	-	-	9(1.50)	3.68(1.38)	46(1.64)	23.3(1.74)
EX7	6	-	-	-	-	8(1.33)	3.72(1.40)	52(1.86)	24.8(1.86)
EX8	7	-	-	-	-	9(1.50)	3.93(1.48)	64(2.29)	27.8(2.08)
EX9	8	-	-	-	-	8(1.33)	3.46(1.30)	56(2.00)	26.0(1.95)
EX10	9	-	-	-	-	9(1.50)	3.80(1.43)	64(2.29)	30.4(2.28)

EX10, it might not be possible to incorporate the SCs because of too much restrictions imposed.

5 Conclusion

In this paper, we have considered the problem of faulty links in NoCs and have proposed a method for reconfiguring the routing mechanism of a network in order to bypass the faulty links. This method is based on using network switches with programmable routing tables. The reconfigured network might have a lower performance due to decreased number of communication links. We considered a new optimized routing mechanism, suitable for mesh networks, and proposed an algorithm to reconfigure the networks using that routing mechanism. The experimental results show that the network reconfigured to bypass the faulty link, has acceptable but degraded performance parameters with regard to the original, non-faulty, network.

Acknowledgments

The authors would like to thank the Iran Telecommunication Research Center (ITRC) for supporting this work.

DOI: 10.1587/elex.4.332
Received January 26, 2007
Accepted March 29, 2007
Published May 25, 2007