Document downloaded from:

http://hdl.handle.net/10251/65562

This paper must be cited as:

Bishnoi, R.; Laxmi, V.; Gaur, MS.; Flich Cardo, J.; Trivino, F. (2015). A Brief Comment on "A Complete Self-Testing and Self-Configuring NoC Infrastructure for Cost-Effective MPSoCs". ACM Transactions in Embedded Computing Systems. 14(1):1-9. doi:10.1145/2668121.

The final publication is available at

http://dx.doi.org/10.1145/2668121

Copyright Association for Computing Machinery (ACM)

Additional Information

"© ACM, 2015. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ACM Transactions on Embedded Computing Systems, Vol. 14, No. 1, Article 2, january 2015. http://doi.acm.org/10.1145/2668121

A Brief Comment on "A complete self-testing and self-configuring NoC infrastructure for cost-effective MPSoCs"[ACM Transactions on Embedded Computing Systems 12 (2013) Article 106]

RIMPY BISHNOI, Malaviya National Institute of Technology Jaipur, India
VIJAY LAXMI, Malaviya National Institute of Technology Jaipur, India
MANOJ SINGH GAUR, Malaviya National Institute of Technology Jaipur, India
JOSÉ FLICH, Universitat Politècnica de València, Spain
FRANCISCO TRIVIÑO, Universidad de Castilla-La Mancha, Spain

In the paper [Ghiribaldi et al. 2013], a complete self-testing and self configuring NoC infrastructure for cost effective MPSoCs was presented in order to make NoC architecture tolerant to faults. To overcome the complexity involved during the complete reconfiguration of routing instance in face of most of the usual failure patterns, authors [Ghiribaldi et al. 2013] proposed a fast self reconfiguration algorithm. The algorithm is based on segment based routing implemented using LBDR (Logic Based Distributed Routing) and claimed to have handled most common NoC faults.

The purpose of this comment is to demonstrate the inconsistency of fast self-configuration method presented in [Ghiribaldi et al. 2013]. To handle inconsistency, we present the correct set of LBDR bits and also argue that complete reconfiguration of routing instance is mandatory to handle some fault combinations. New coverage results of the fast self reconfiguration algorithm of [Ghiribaldi et al. 2013] are also presented.

Categories and Subject Descriptors: B.7.7 [**Integrated Circuits**]: Reliability and Testing

General Terms: General Terms: Design, Algorithms, Reliability

Additional Key Words and Phrases: Network-on-Chip, Fault Tolerance

ACM Reference Format:
Rimpy Bishnoi, Vijay Laxmi, Manoj Singh Gaur, José Flich, Francisco Trivino 2013. A Brief Comment on "A complete self-testing and self-configuring NoC infrastructure for cost-effective MPSoCs". ACM Trans. Embedd. Comput. Syst. V, N, Article A (January YYYY), 8 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Paper [Ghiribaldi et al. 2013] addressed two conflicting challenges of designing an effective testing as well as reconfiguration strategy to deal with irregular topologies generated due to manufacturing faults. To support reconfiguration, authors [Ghiribaldi et al. 2013] employed a flexible routing framework, including Segment based routing algorithm (SR) [Mejia et al. 2008], a fault-tolerant topology agnostic routing algorithm along with a reprogrammable and scalable logic based distributed routing implementation mechanism (LBDR) [Rodrigo et al. 2009]. Authors in [Ghiribaldi et al. 2013] pointed out that for a given irregular topology, underlying routing algorithm instance

This work is supported by Indo-Spain project grant (DST/INT/Spain/P-35/11/1).
Authors' addresses: R. Bishnoi, V. Laxmi and M. S. Gaur, Computer Science Department, Malaviya National Institute of Technology Jaipur, India; J. Flich, Universitat Politècnica de València, DISCA, Spain; F. Trivino Universidad de Castilla-La Mancha, Spain;
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 1539-9087/YYYY/01-ARTA $15.00
DOI: http://dx.doi.org/10.1145/0000000.0000000

may not be suitable for LBDR i.e unable to offer full connectivity and deadlock freedom. In that case, a new routing instance is required to support this i.e complete reconfiguration of underlying routing instance. However, as the process of complete reconfiguration of routing instance is complex and time consuming, paper [Ghiribaldi et al. 2013] presented a fast self reconfiguration algorithm to handle most common NoC failures. This algorithm, at design time, computes the set of LBDR bits [Rodrigo et al. 2009] that need to be changed in order to support failures and stores them in a table called transition table (TT). LBDR bits are calculated to support all the 1-link failure combinations and most of the 2-link failure combinations. On the event of a topology change, the fast self configuration algorithm generates the new modified set of LBDR bits only by accessing the transition table. This reduces the reconfiguration time required to handle a failure, i.e transition table access time. Thus becomes faster than a complete reconfiguration of routing algorithm instance.

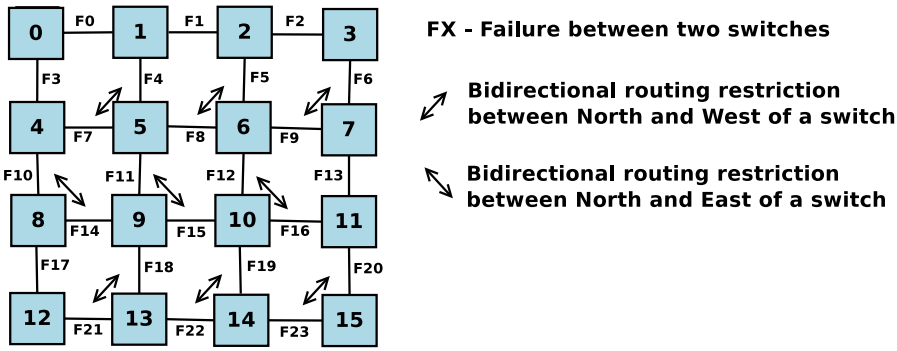


Fig. 1. SR routing instance [Mejia 2008] applied on a 4×4 mesh topology used in our counter examples

Table I shows the original transition table of [Ghiribaldi et al. 2013] (shown in Figure 12 of [Ghiribaldi et al. 2013]), to support all single link fault patterns of the topology shown in Figure 1. This table is precomputed at design time in advance to any possible failure detection and lists all the configuration bits that need to be modified in order to support all possible cases (24 for 4×4 mesh) of one-link fault. The purpose is to use the table later at run-time to retain connectivity and deadlock freedom in the event of a link fault by changing the respective LBDR bits at switches. Authors of [Ghiribaldi et al. 2013] represent all possible one link failures as F0 to F23 as shown in Table I. Entry labeled as Fx represents the LBDR bits to be changed on a failure of a corresponding link marked as Fx in the Figure 1. Authors of [Ghiribaldi et al. 2013] optimizes the configuration algorithm for two link failure cases by applying the algorithm of one link failure two times, each one focused on each single link failure. That is accessing the transition table two times corresponding to each link failure. Paper [Ghiribaldi et al. 2013] claims to support all the single link failure cases and most of the two link failures. Authors [Ghiribaldi et al. 2013] have classified all the combinations in four sets: 1F ,2FC , 2FI , and 2FS set. 1F set represents all the combinations with having only one failed link. 2FC and 2FI sets refers to two link failures which are compatible (supported) and not compatible (not supported) respectively, with the fast self reconfiguration algorithm. Finally, 2FS refers to the case of special two link failures that can be supported by the algorithm by adding new changes. These special changes are stored in another table referred as special table (ST [1]). In the paper [Ghiribaldi et al.

[1]The ST table is too long for being detailed in this paper

2013] authors claim to support all link failures of sets 1F, 2FC, and 2FS by the self reconfiguration algorithm.

Contributions: In this comments paper, we demonstrate that the self reconfiguration strategy is not able to handle all types of failures (1F, 2FC, 2FS) as reported in [Ghiribaldi et al. 2013]. We have identified that the transition table (Table I) used by the self reconfiguration algorithm contains incorrect values of LBDR bits and consequently proposed a modified transition table with correct set of LBDR bits. Additionally we also show that the complete reconfiguration of a routing instance is mandatory to handle some of the fault cases. This observation reduces the supported number of link failures reported by [Ghiribaldi et al. 2013]. New coverage results of the fast configuration algorithm are presented along with counter examples to demonstrate the scenarios.

Table I. Original table of LBDR bits to be changed on fault detection for a 4×4 mesh topology

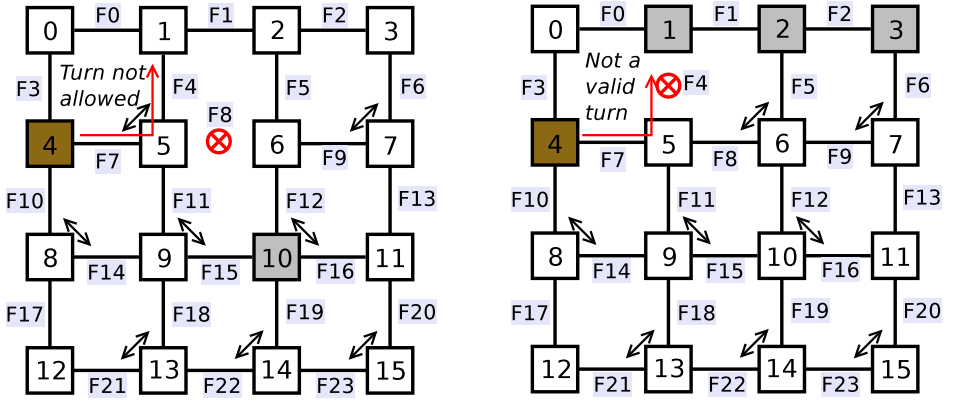
FAIL	SW	OP	SW	OP	LBDR BITS TO BE CHANGED						
F0	0	E	1	W	Ce[0]=0	Cw[1]=0	DR[0,1]=S	Ren[4]=1	Rsw[1]=1		
F1	1	E	2	W	Ce[1]=0	Cw[2]=0	DR[1,2]=S	Ren[5]=1	Rsw[2]=1		
F2	2	E	3	W	Ce[2]=0	Cw[3]=0	DR[2,3]=S	Ren[6]=1	Rsw[3]=1		
F3	0	S	4	N	Cs[0]=0	Cn[4]=0	DR[0,4]=E	Ren[4]=1	Rsw[1]=1		
F4	1	S	5	N	Cs[1]=0	Cn[5]=0	DR[1,5]=W	Ren[4]=1	Rsw[1]=1		
F5	2	S	6	N	Cs[2]=0	Cn[6]=0	DR[2,6]=W	Ren[5]=1	Rsw[2]=1		
F6	3	S	7	N	Cs[3]=0	Cn[7]=0	DR[3,7]=W	Ren[6]=1	Rsw[3]=1		
F7	4	E	5	W	Ce[4]=0	Cw[5]=0	DR[4,5]=N	Ren[4]=1	Rsw[1]=1		
F8	5	E	6	W	Ce[5]=0	Cw[6]=0	DR[4,5,6]=N	Ren[5]=1	Rsw[2]=1	Ree[4]=0	
F9	6	E	7	W	Ce[6]=0	Cw[7]=0	DR[5,6,7]=N	Ren[6]=1	Rsw[3]=1	Ree[5,6]=0	
F10	4	S	8	N	Cs[4]=0	Cn[8]=0	DR[4,8]=E	Rse[4]=1	Rwn[9]=1		
F11	5	S	9	N	Cs[5]=0	Cn[9]=0	DR[5,9]=E	Rse[5]=1	Rwn[10]=1		
F12	6	S	10	N	Cs[6]=0	Cn[10]=0	DR[6,10]=E	Rse[6]=1	Rwn[11]=1		
F13	7	S	11	N	Cs[7]=0	Cn[11]=0	DR[7,11,3]=W	Rse[6]=1	Rwn[11]=1	Rww[11]=1	Rss[3]=0
F14	8	E	9	W	Ce[8]=0	Cw[9]=0	DR[8,9,10,11]=N	Rse[4]=1	Rwn[9]=1	Rww[10,11]=0	
F15	9	E	10	W	Ce[9]=0	Cw[10]=0	DR[9,10,11]=N	Rse[5]=1	Rwn[10]=1	Rww[11]=0	
F16	10	E	11	W	Ce[10]=0	Cw[11]=0	DR[10,11]=N	Rse[6]=1	Rwn[11]=1		
F17	8	S	12	N	Cs[8]=0	Cn[12]=0	DR[8,4]=E DR[12]=N	Ren[12]=1	Rsw[9]=1	Rss[4]=0	
F18	9	S	13	N	Cs[9]=0	Cn[13]=0	DR[9,13]=W	Ren[12]=1	Rsw[9]=1		
F19	10	S	14	N	Cs[10]=0	Cn[14]=0	DR[10,14]=W	Ren[13]=1	Rsw[10]=1		
F20	11	S	15	N	Cs[11]=0	Cn[15]=0	DR[11,15]=W	Ren[14]=1	Rsw[11]=1		
F21	12	E	13	W	Ce[12]=0	Cw[13]=0	DR[12,13]=N	Ren[12]=1	Rsw[9]=1	Ree[12]=0	
F22	13	E	14	W	Ce[13]=0	Cw[14]=0	DR[12,13,14]=N	Ren[13]=1	Rsw[10]=1	Ree[13,14]=0	
F23	14	E	15	W	Ce[14]=0	Cw[15]=0	DR[13,14,15]=N	Ren[14]=1	Rsw[11]=1		

2. COUNTER EXAMPLE SCENARIOS

This section presents the example scenarios to demonstrate that the original table (Table I) used by the self configuration algorithm provided in [Ghiribaldi et al. 2013] contains incorrect values of LBDR bits. We also show the cases in which LBDR mechanism is not compatible with the current routing instance and a complete reconfiguration of routing instance is mandatory to handle those cases.

2.1. Counter Example-I

Entry labeled as F8 in Table I belongs to a link fault located east of switch 5 and west of switch 6 as shown in Figure 2(a). The routing restriction situated at switch 6 before failure is removed by setting R_{en} bit of switch 5 and R_{sw} bit of switch 2 to one. Connectivity bits of particular switch output ports are set to zero and deroute of switches 4, 5 and 6 is set to North. R_{ee} bit of node 4 is also set to zero.



(a) Link Failure (F8) between switch 5 and 6 (b) Link Failure (F4) between switch 1 and 5

Fig. 2. Illustrative examples

However, according to LBDR fundamentals, to provide deadlock freedom, deroute bit (required for non-minimal paths) for a switch should be set in such a way that it should not violate (cross) any routing restriction set by underlying routing algorithm. Our purpose here is to demonstrate that the entry labeled as F8 contains incorrect bits and leads to a deadlocked situation. As an example we take 4 as a source and 10 as a destination switch. First part of LBDR logic tells that destination is in SE quadrant from the source switch. After knowing the destination's quadrant, the second part of LBDR logic computes a valid output port among S and E using routing and connectivity bits. In this case, it provides E as valid output port as R_{es} and C_e of node 4 is set to one and S as invalid output port because R_{se} of switch 4 is set to zero due to the restriction located at switch 8. After arriving at switch 5, LBDR logic performs the same computation and finds destination is in SE quadrant. But at this switch, LBDR cannot provide any valid output port because R_{se} and C_e both are set to zero of switch 5. Since LBDR logic is not able to provide any output port, deroute logic gets activated. But in entry labeled as F8, deroute of switch 5 is set to North, which is incorrect due to the presence of W-N routing restriction located at switch 5. Indeed according to the configuration bits of Table I, the packet will go north as DR[5] = N and crosses a routing restriction, thus, potentially creating a cycle. Hence, an improper situation has occurred with the original transition table for this case (F8) in [Ghiribaldi et al. 2013] and with this set of bits, the fast configuration algorithm shall not be able to handle failure F8. Similar scenarios exist for failures labeled as F9, F14, and F15 and their corresponding entries in Table I.

2.2. Counter Example-II

Another example scenario is shown in Figure 2(b). In case of fault F4, for source 4 and any destination present in NE (North-East) quadrant (switch 1, 2, and 3), LBDR provides both North (N) and East (E) as valid output port. Whereas, in reality only N is a valid output port as shown in Figure 2(b). This happens because the entry corresponding to failure F4 in Table I is incorrect. On failure of a link labeled as F4, configuration algorithm generates modified LBDR bits by accessing entry F4 of Table I that sets the R_{en}[4] of switch 4 to one. Due to this, at switch 4, along with N port, LBDR provides E port also as a valid port for any destination present in its NE quadrant. But considering E port at switch 4 for destinations present in NE quadrant could lead to packet drop situation at next switch 5. Because for packets coming at west port of

switch 5 and intended to destinations located in NE quadrant, LBDR is unable to generate any path. All possible paths from switch 5 either cross a routing restriction or require U-turn, both are not allowed in LBDR to provide deadlock freedom. Similar type of inconsistencies exist in all the entries of Table I except those mentioned in example 1.

3. UNDERLYING PROBLEM AND FIX IN THE TRANSITION TABLE

As demonstrated in section 1, the original table (Table I) of [Ghiribaldi et al. 2013] contains incorrect bits arising out of proposed implementation. Some additional bits need to be added and some need to be updated in Table I. In order to fix the transition table, firstly, we fixed the checker tool internally used by the authors of the original paper. The tool initially starts with a fault-free LBDR configuration for a particular mesh dimension. For each fault combination, the tool applies the LBDR bits to be changed (TT and ST tables) in order to support the fault combination that is being analyzed. Once the corresponding LBDR bits have been modified, the next step is to check whether the final topology plus the LBDR configuration preserves connectivity and freedom from deadlock.

Connectivity property is guaranteed if any pair of end-nodes of the mesh are connected through all the possible paths provided by LBDR (minimal or non-minimal). The tool checks if every pair of end-nodes are reachable by sending a message, which is routed by using the LBDR mechanism. At the same time, the tool checks if cycles are formed. To do this, the corresponding channel dependency graph is built and cycles are searched. In case of no cycles found, the network is deadlock free. Therefore, if every pair of end-nodes is reachable without containing cycles in the path, then the fault combination is supported by the configuration algorithm.

This helped us also to understand where the problem was. Basically, the compatibility of the LBDR mechanism with the underlying routing instance was not correctly tested by the tool. As LBDR mechanism may not be compatible with the current routing instance generated due to failure. In this case a new routing instance needs to be configured, i.e complete reconfiguration of routing instance. For example, as shown in counter example 2.1, due to failure F8, available minimal path between some source destination pair gets faulty and becomes a blocked path[2]. This failure is not visible from source switches like 0 or 4 because LBDR is unable to capture this. Due to this LBDR keeps forwarding packets to a blocked path having failure F8 and results into a non supported topology. Hence this particular routing instance generated from link failure F8 is not compatible with LBDR mechanism. To solve this problem, the tool has been improved in order to consider the compatibility of the LBDR mechanism with the current routing instance. Now the tool is able to test all the possible branches that are generated by LBDR at a given switch. In case one of these branches fails and blocks at some intermediate switch, then the fault combination under test is flagged as not covered. These are the cases which require a complete reconfiguration of routing instance and shown using a asterisk in Table II (F8, F9, F14, F15).

Besides this, other reason is the generation of incorrect values of routing bits on event of any failure. For example, as shown in counter example 2.2, on link failure F4, self configuration algorithm sets routing bit R_{en} of switch 4 to one as shown in Table I. This resulted into a non-supported failure. To solve this, Table I is corrected and modified set of bits are provided in Table II. Each entry of Table II is now tested by the tool against the connectivity and deadlock freedom. Finally, as can be deduced, the test is now more restrictive and the coverage is slightly reduced as shown in the next section.

[2]Path which gets block at some intermediate node and no other (minimal or non-minimal) path is available from that particular node (In case of F8, path between node 4 and 10 blocks at intermediate node 5)

Table II. New table for LBDR bits to be changed on fault detection for a 4×4 mesh topology

FAIL	**SW**	**OP**	**SW**	**OP**	**BITS TO BE CHANGED**											
F0	0	E	1	W	Ce[0]=0	Cw[1]=0	DR[0,1]=S	Ren[4]=1	Rsw[1]=1	**Rne[4]=0**	**Rnw[5]=0**					
F1	1	E	2	W	Ce[1]=0	Cw[2]=0	DR[1,2]=S	Ren[5]=1	Rsw[2]=1	**Rnw[6]=0**	**Rne[5]=0**					
F2	2	E	3	W	Ce[2]=0	Cw[3]=0	DR[2,3]=S	Ren[6]=1	Rsw[3]=1	**Rne[6]=0**	**Rnw[7]=0**					
F3	0	S	4	N	Cs[0]=0	Cn[4]=0	DR[0,4]=E	Ren[4]=1	Rsw[1]=1	**Rws[1]=0**	**Rwn[5]=0**					
F4	1	S	5	N	Cs[1]=0	Cn[5]=0	DR[1,5]=W	**Res[0]=0**								
F5	2	S	6	N	Cs[2]=0	Cn[6]=0	DR[2,6]=W	**Res[1]=0**								
F6	3	S	7	N	Cs[3]=0	Cn[7]=0	DR[3,7]=W	**Res[2]=0**								
F7	4	E	5	W	Ce[4]=0	Cw[5]=0	DR[4,5]=N	**Rse[0]=0**								
F8*	5	E	6	W	Ce[5]=0	Cw[6]=0	**DR[5,6]=N**	**Rse[1]=0**	**Rse[0]=0**	**Rwn[5]=0**	**Ren[4]=1**	**Rsw[1]=1**				
F9*	6	E	7	W	Ce[6]=0	Cw[7]=0	**DR[6,7]=N**	**Ren[4]=1**	**Rsw[1]=1**	**Ren[5]=1**	**Rsw[2]=1**	**Rse[0]=0**	**Rwn[5]=0**	**Rse[1]=0**	**Rwn[6]=0**	**Rse[2]=0**
F10	4	S	8	N	Cs[4]=0	Cn[8]=0	DR[4,8]=E	**Rws[5]=0**								
F11	5	S	9	N	Cs[5]=0	Cn[9]=0	DR[5,9]=E	**Rws[6]=0**								
F12	6	S	10	N	Cs[6]=0	Cn[10]=0	DR[6,10]=E	**Rws[7]=0**								
F13	7	S	11	N	Cs[7]=0	Cn[11]=0	DR[3,7,11]=W	Rse[6]=1	Rwn[11]=1	Rss[3]=0	**Res[2]=0**	**Res[6]=0**	**Ren[10]=0**			
F14*	8	E	9	W	Ce[8]=0	Cw[9]=0	**DR[8,9]=N**	**Rsw[5]=0**	**Rse[5]=1**	**Ren[9]=0**	**Rse[6]=1**	**Ren[10]=0**	**Rwn[10]=1**	**Rsw[6]=0**	**Rwn[11]=1**	**Rsw[7]=0**
F15*	9	E	10	W	Ce[9]=0	Cw[10]=0	**DR[9,10]=N**	**Rsw[6]=0**	**Rse[6]=1**	**Rsw[7]=0**	**Rwn[11]=1**	**Ren[10]=0**				
F16	10	E	11	W	Ce[10]=0	Cw[11]=0	DR[10,11]=N	**Rsw[7]=0**								
F17	8	S	12	N	Cs[8]=0	Cn[12]=0	**DR[8,4,12]=E**	**Rws[9]=0**	**Rwn[13]=0**	**Rws[5]=0**	Rss[4]=0	Ren[12]=1	Rsw[9]=1			
F18	9	S	13	N	Cs[9]=0	Cn[13]=0	DR[9,13]=W	**Res[8]=0**								
F19	10	S	14	N	Cs[10]=0	Cn[14]=0	DR[10,14]=W	**Res[9]=0**								
F20	11	S	15	N	Cs[11]=0	Cn[15]=0	DR[11,15]=W	**Res[10]=0**	**Ren[14]=0**							
F21	12	E	13	W	Ce[12]=0	Cw[13]=0	DR[12,13]=N	**Rse[8]=0**								
F22	13	E	14	W	Ce[13]=0	Cw[14]=0	DR[12,13,14]=N	**Rse[9]=0**	**Ree[12]=0**	**Rse[8]=0**						
F23	14	E	15	W	Ce[14]=0	Cw[15]=0	**DR[12,13,14,15]=N**	**Rse[8]=0**	**Rse[9]=0**	**Rse[10]=0**	**Ree[12]=0**	**Ree[13]=0**				

We have incorporated these modifications and computed a new set of LBDR bits as shown in Table II. Using these modified sets of bits, each fault can be handled properly and LBDR provides valid output ports (Counter example-I and II). The table shows in bold face the new bits that need to be changed. Also, those entries labeled with an asterisk require a slow reconfiguration strategy, as during the transition potential deadlocks may arise. This is due as these configurations require changing the location of some routing restrictions, which is deadlock-prone. The mechanism proposed in [Strano et al. 2012] can be used to avoid deadlocks. Figure 3 shows the most complex case for F14, which involves a reconfiguration process to avoid deadlock. The figure shows the situation before and after the failure. Notice that routing restrictions located at switch 8 and 9 between their North and East port have been moved to switch 9 and 10 between their North and West port. Finally, we have accordingly modified the special table (ST) in order to increase coverage for special two-link failures. As the number of combinations generated from two link failures are very high, the corresponding ST table is also too long for being detailed in this paper.

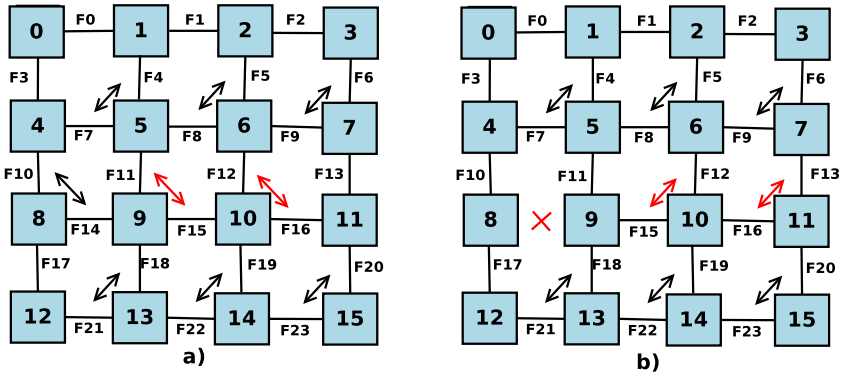


Fig. 3. Situation **a)** before and **b)** after the link failure between node 8 and 9

3.1. New Coverage results of the Configuration Algorithm

In this section, we show the obtained coverage results of configuration algorithm of [Ghiribaldi et al. 2013] with the updated table for all the possible combinations of one and two-link faults. Table III shows the new coverage result of fast self configuration algorithm for all one link fault cases in all the possible mesh dimensions. Notice that now the self configuration algorithm is not able to offer full coverage support. Reason being is that, few one link failure cases belonging to set 1F are not compatible with the LBDR mechanism. The corresponding failures for a 4×4 mesh are identified in this comments paper and marked using asterisk in Table II. These failures require the complete reconfiguration of underlying routing instance and can not be handled by fast self configuration method.

Table III. New coverage percentage for one-link faults

Topology mesh	Total Links	Supported Links	% Old coverage	% New coverage	% Reduction
4x4	24	20	100	83.3	-16.7
5x5	40	31	100	77.5	-22.5
6x6	60	44	100	73.3	-26.7
7x7	84	59	100	70.2	-29.8
8x8	112	76	100	67.8	-32.2

In the case of two-link faults, there are several topologies that are not supported, as was the case in the previous version of the algorithm. Table IV shows the new coverage compared with the previous one. According to our new tool, the new coverage is decreased and the number of supported combinations is reduced by 4.5% and 10.4% for 4×4 and 8×8 meshes, respectively. The main reason lies in the adaptivity of the LBDR mechanism. As we have mentioned previously, the new tool has been enhanced in order to test all the possible output ports provided by the LBDR mechanism at any hop along the path. Therefore, the new tool is accurate in finding an incorrect path and therefore to label the topology as invalid.

Table IV. New coverage percentage for two-link faults

Topology mesh	Supported Links	Unsupported Links	% Old coverage	% New coverage	% Reduction
4x4	267	33	93.5	89.0	-4.5
5x5	684	136	96.3	83.4	-12.8
6x6	1537	293	97.6	84.0	-12.3
7x7	3186	384	98.4	89.2	-9.15
8x8	5589	739	98.8	88.3	-10.4

4. CONCLUSIONS

The issues raised in this comments paper are about the fast self-configuration part proposed in [Ghiribaldi et al. 2013]. In this comment, we gave counter examples covering different possible scenarios to demonstrate that the self-configuration algorithm of [Ghiribaldi et al. 2013] is not sufficient to handle all types of single (1F set) and double link faults (2FC & 2FS set). Through appropriate counter examples, we illustrated the limiting cases of [Ghiribaldi et al. 2013] and argued that the complete reconfiguration process of a routing instance is mandatory to handle some link failures. We also provide suitable modification of table entries to tackle reported faults. New coverage results of the self configuration algorithm are also provided for link failures of 1F, 2FC and 2FS sets. It has been analysed that the coverage is reduced for set of both one and two link failures.

REFERENCES

Alberto Ghiribaldi, Daniele Ludovici, Francisco Triviño, Alessandro Strano, José Flich, José LUIS Sánchez, Francisco Alfaro, Michele Favalli, and Davide Bertozzi. 2013. A Complete Self-testing and Self-configuring NoC Infrastructure for Cost-effective MPSoCs. ACM Trans. Embed. Comput. Syst. 12, 4 (July 2013), 106:1–106:29. DOI:http://dx.doi.org/10.1145/2485984.2485994

A. Mejia. 2008. Design and Implementation of Efficient Topology Agnostic Routing Algorithms for Interconnection Networks. Ph.D. Dissertation. University of Valencia.

A. Mejia, J. Flich, and Duato J. 2008. On the Potentials of Segment-Based Routing for NoCs. In Proceedings of the 37th International Conference on Parallel Processing(ICPP'08). IEEE, 594–603. DOI:http://dx.doi.org/10.1109/ICPP.2008.56

S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato. 2009. Efficient implementation of distributed routing algorithms for NoCs. Computers Digital Techniques, IET 3, 5 (2009), 460–475. DOI:http://dx.doi.org/10.1049/iet-cdt.2008.0092

A. Strano, D. Bertozzi, F. Trivino, J.L. Sanchez, F.J. Alfaro, and J. Flich. 2012. OSR-Lite: Fast and deadlock-free NoC reconfiguration framework. In Proceedings of the International Conference on Embedded Computer Systems (SAMOS). 86–95. DOI:http://dx.doi.org/10.1109/SAMOS.2012.6404161