Improving Fault Tolerance of Network-on-Chip Links via Minimal Redundancy and Reconfiguration

Hamed S. Kia, and Cristinel Ababei
Department of Electrical and Computer Engineering
North Dakota State University
Fargo ND, 58108-6050
Email: {hamed.sajjadikia, cristinel.ababei}@ndsu.edu

***Abstract*—We propose to partition links in a network-on-chip into multiple segments and use spare wires at the level of each segment to address permanent errors due to manufacturing or wearout defects. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. The proposed self-repairing segmented link structure is implemented and simulated in Verilog and verified on a Virtex 5 FPGA. Experimental results on area, power consumption, delay, and reliability show that the optimal link is achieved when the link is partitioned into two segments.**

I. Introduction

Advances in integrated circuit fabrication technology enables the integration of tens and hundreds of cores on the same system-on-chip (SoC). To address the demand for increased communication and concurrency between cores, network-on-chip (NoC) has emerged as a new communication design paradigm [1]. An NoC is constructed by connecting a set of routers via bidirectional links. Cores are connected to routers through network interfaces and communicate via messages organized as packets. For example, a typical 2D regular mesh NoC topology is shown in Fig.1.

However, advances in fabrication technology that can make this integration possible may also make the underlying hardware less reliable due to an increasing number of defects and wearout or aging mechanisms. Errors due to these faults may occur in both the computation (cores) and communication (network) components of the SoC. Because it is critical to deal with these faults and address them with cost-effective solutions, one of the major problems facing the design of network-on-chip based systems-on-chip is reliability.

In this paper, we focus on the communication component and address permanent faults which can occur in NoC links during fabrication or due to wearout mechanisms. More specifically, we improve the fault tolerance of NoCs by using redundant or spare wires within the context of segmented links. We do not focus on the computation component, which has been studied more extensively [2], [3]. The remainder of this paper is organized as follows. In the next section, we discuss previous work and then outline our main contribution. Then, we describe the proposed self repairing link and introduce a simple technique to detect faults. Later, we report experimental results. Finally, we conclude by summarizing our main contribution.

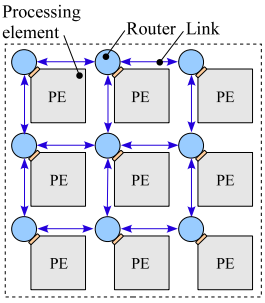

Fig. 1. Example of typical 2D regular mesh NoC topology.

II. Previous Work

There are basically two methods to address permanent faults in NoC links. The most popular method is based on the use of adaptive or dynamic routing strategies. The idea is that after faulty links are detected and located, specialized routing algorithms compute new routing paths through the network such that these broken links are avoided. Examples of recently proposed adaptive routing algorithms for NoCs include [4]–[9]. While this method is straightforward, it has the disadvantage of affecting the network performance (average flit latency) due to typically longer routing paths. Also, it must deal with the issue of deadlock and livelock.

The second method to address permanent faults is based on the concept of adding redundant or spare elements and on the use of reconfiguration [10]–[14]. In the case of NoC links, several redundant wires are added to each link. When faults occur, reconfiguration is used to replace the faulty wires with spare healthy wires. For example, the authors of [15] have used redundant spare wires to repair broken links. The use of redundant wires to replace faulty wires without interruption of the data flow is presented in [16]. The authors use a periodic inline test method to detect faulty wires. The authors in [17] demonstrate link structures, which can tolerate transient, intermittent, and permanent faults. They use Hamming coding to address transient errors and retransmission as the recovery technique. They have also introduced two techniques to address intermittent and permanent faults. The first technique is based on spare wires and reconfiguration

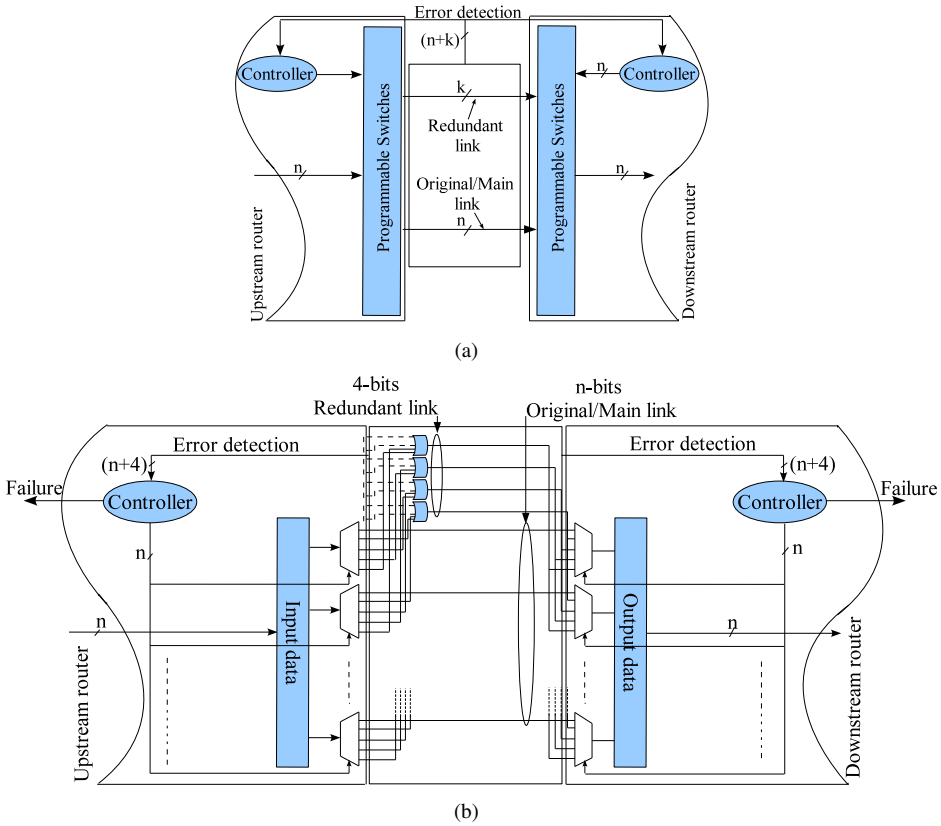

Fig. 2. (a) Simplified block diagram of an n-bit reconfigurable link structure with k redundant bits. (b) Details of the reconfiguration logic for an example with $k = 4$.

while the second technique uses time redundancy. This method has the disadvantage of addressing a rather small number of faults in a given link, which is limited by the additional area occupied by the multiplexers and demultiplexers utilized to reconfigure the link.

III. Contribution

In this paper, we focus on permanent faults which can occur in NoC links during fabrication or due to wearout mechanisms. Our goal is to design a self repairing NoC link, using spare wires and reconfiguration, which offers the best trade off between the overhead in area, power consumption, delay and reliability. Our main contribution lies in: 1) We propose to partition NoC links into multiple segments and use spare wires at the level of each segment. Because different segments of the spare wires address different faults from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. 2) We implement and simulate the proposed self repairing link structure in Verilog using Cadence tools and verify it on a Virtex 5 FPGA. Experimental results on area, power consumption, delay, and reliability show that the optimal link is achieved when the link is partitioned into two segments.

IV. Proposed Self Repairing Link Structure

In this section, we introduce the proposed segmented self repairing link structure. However, first we present the traditional non-segmented link structure, which we will use to build our discussion and for comparison purposes.

A. Non-segmented link structure

The block diagram of the traditional reconfigurable link is shown in Fig. 2.a. The main link connecting two routers of the NoC is essentially an n-bit bus formed of n primary wires. In addition, the link contains k spare wires. These spare wires are redundant and are normally not used. The error detection logic is responsible with the detection of permanent faults, which initially are assumed to occur only in the n primary wires. This is a reasonable assumption because the spare wires are not normally used and therefore they are not affected by wearout mechanisms. The error detection logic is designed to test all $n + k$ wires because once a spare wire is used to reconfigure the link, it will also be affected by wearout mechanisms. Once a fault is detected by the error detection logic, the link is reconfigured to replace the faulty wire with one of the healthy k redundant wires. The reconfiguration is realized with two sets of configurable switches (multiplexers and demultiplexers) controlled by signals generated by controllers, which reside in the downstream and upstream routers.

While the link structure in Fig.2.a is simple, it suffers from poor scalability. Obviously, as the number of redundant bits k increases the reliability of the whole link increases too. However, the link structure must use $n \times 1 - to - (k + 1)$

demultiplexers in the upstream router and $n \times (k+1) - to - 1$ multiplexers in the downstream router. In addition, the complexity and hence area of the two controllers also increases with the increase of k. Therefore, in order to keep the area overhead within reasonable limits, in practice k must be limited to a small number. The block diagram in Fig.2.b shows implementation details. In this case the number of redundant bits is $k = 4$ and therefore the link can only handle up to 4 faults. Once the number of faults increases to more than 4, the controllers enable a failure signal to inform the routers that the link is broken and cannot be used anymore at the initial bandwidth. This information can then be utilized by adaptive routing algorithms, which will find new routing paths such that the broken link is avoided.

The reliability of the link structure from Fig.2.a can be computed using an approach similar to the study in [18]. For simplicity, we assume that faults can only occur in the main link and that the redundant bits are fault free. The link structure in Fig.2.a can be seen as a parallel system. Assuming that all n wires of the main link are identical and that the probability of success of any of the n bits is p (that is the probability of the wire to be functional), then the probability of exactly $(n-k)$ bits working correctly out of n bits is given by the binomial distribution:

$$P(n-k, n, p) = \binom{n}{n-k} p^{n-k} (q)^k \tag{1}$$

where $q = 1 - p$ represents the probability of the wire to be non-functional.

The link structure is said to work successfully as long as at least $(n-k)$ bits of the main link remain functional. Therefore, the reliability of the link structure can be defined as the probability obtained by summing-up the probabilities of all possible successful configurations:

$$R(n-k, n, p) = \sum_{i=n-k}^{n} \binom{n}{i} p^i (q)^{n-i} \tag{2}$$

B. Proposed segmented link structure

To improve its fault tolerance, we propose to partition the link into multiple segments and use spare wires at the level of each segment. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. Fig.3 illustrates the main idea behind the proposed segmented link structure. In this figure, the link with only one redundant wire is divided into 3 segments. Faults, represented as 'x', can occur anywhere along the length of the link. The 'x' on the first wire in the first segment of the link illustrates the occurrence of a fault. The link structure from Fig.2 would replace this faulty wire with the only available redundant wire. However, in this example there are two remaining faults, which can not be addressed by the link structure from Fig.2. In the proposed segmented link structure, we use only the first segment of the redundant wire to replace the faulty wire from the first link segment.

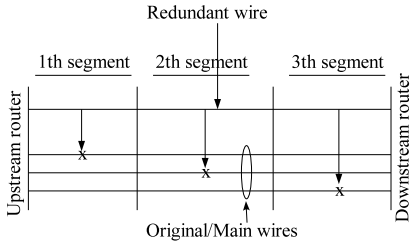

Fig. 3. Illustration of proposed segmented link structure.

The second and third faults denoted with 'x' in Fig.3 can be handled using the second and third segments of the redundant wire. Next, we discuss the main limitation of the proposed segmented link structure and present an elegant solution to address it.

The proposed segmented link has better fault tolerance (hence better reliability) compared to traditional non-segmented link structure. However, it also suffers from the scalability problem as the link structure from Fig.2. In fact, the proposed link structure would utilize larger area overheads due to the segmentation, which requires multiple sets of multiplexers and demultiplexers, error detection circuits, and more sophisticated controllers.

To solve the scalability problem and to reduce the hardware overhead, the main and redundant wires are divided into L groups or subsets. Each of the groups of main wires has n_i bits and each of the groups of redundant wires has k_i bits, where $i = 1, 2, ..., L$. Each of the groups of main wires has designated a group of redundant wires. Faulty wires from a given main group can be replaced only by wires from the designated group of redundant wires. This solution effectively reduces the size of the programmable switches (that is multiplexers and demultiplexers) yet enabling the improvement in fault tolerance. In this case, we only need to use $L \times n_i \times 1 - to - (k_i + 1)$ demultiplexers and $L \times n_i \times (k_i + 1) - to - 1$ multiplexers. The proposed solution is illustrated in the block diagram from Fig.4.

To derive an expression for reliability, we again assume that faults can only occur in the main wires and that the redundant wires are fault free. Our derivation is based on the block diagram shown in Fig.5. In this figure R_b represents the reliability of a block composed of a group of main wires together with its designated group of redundant wires. Notice that since we divided the link into m segments, the probability of failure of a bit of the main link in any segment is now $\frac{1}{m} \times q$, and hence the probability of success is $(p + \frac{m-1}{m} \times q)$. Similar to the discussion from the previous section, we assume that all wires of the main link are identical. Then, the probability of exactly $(n_i - k_i)$ bits working correctly out of n_i bits is given by the binomial distribution:

$$P(n_i - k_i, n_i, p) = \binom{n_i}{n_i - k_i} (p + \frac{m-1}{m} \times q)^{n_i - k_i} (\frac{1}{m} \times q)^{k_i} \tag{3}$$

Therefore the reliability of one block of a segment (formed

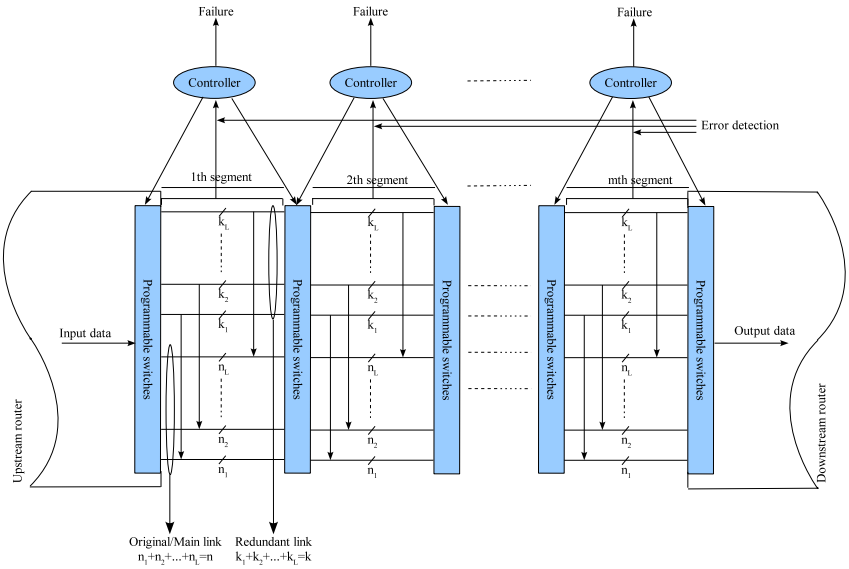

Fig. 4. Block diagram of the proposed segmented self repairing link. In each segment, the link is divided into groups of main wires and redundant wires.

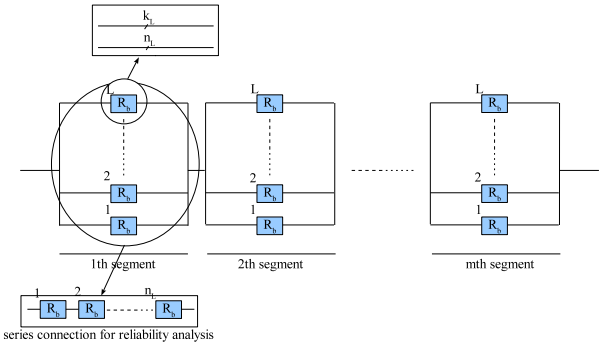

Fig. 5. System level diagram utilized for reliability computation.

by a group of main wires and its designated group of redundant wires) can be computed as:

$$R_b(n_i - k_i, n_i, p) = \sum_{j=n_i-k_i}^{n_i} \binom{n_i}{j} (p + \frac{m-1}{m} \times q)^j (\frac{1}{m} \times q)^{n_i - j} \quad (4)$$

Because failure of any block of a segment results in system failure, the segment must be modeled as a system of a series of blocks as illustrated in Fig.5. Therefore, the overall reliability of a segment can be computed as:

$$R_s = R^L \quad (5)$$

Finally, the whole segmented link is modeled as a series system, for which the total reliability of the system can be computed as:

$$R_{link} = R_s^m \quad (6)$$

V. Fault Detection

The operation of both non-segmented and segmented reconfigurable links depends on the ability to detect faulty wires. To detect faulty wires, we propose an effective FSM-based fault detection circuit. We assume that the NoC based system enters a test mode periodically. Fig.6 shows the design of the fault detection circuit for one bit of the link. In this circuit, during normal operation, the *test mode enable* signal is 0, hence the link carries regular packets from the upstream router. When the system enters the test mode (*test mode enable* signal becomes 1) the link carries the *test signal*. Testing is done by sending over the link a pre-defined *pulse test signal*, which exercises each wire of the link with both low and high logic levels (to address stuck at low or high faults). At the receiving downstream router side, initially the *error signal* is set to 0 by the *reset* signal. As long as the *test mode enable* signal is 0 the *error signal* holds its value in state S_0. When the system enters the test mode, the receiving downstream router expects to receive a 0 first and then a 1. This causes a transition from S_0 to S_1 and then a transition back to S_0 while the error signal remains 0 indicating that the wire is healthy. However, any different signal received during the test mode, generates an error signal, which is set to 1. Once the system leaves the test mode (*test mode enable = 0*) the fault detection circuit holds the value of the final *error signal*. This error signal will be used by controllers during normal operation to decide if a wire needs to be replaced by a healthy redundant wire or not.

VI. Simulation and Experimental Results

In this section we compare the proposed segmented link structure against the non-segmented link structure. Both link structures, shown in Fig.2.b and Fig.4, are coded in Verilog-HDL. The Verilog-HDL implementations are synthesized and simulated using Cadence tools [19] while the hardware validation is done using Xilinx ISE tools [20]. We use an

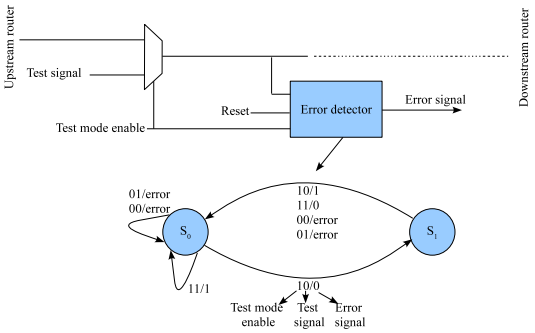

Fig. 6. Block diagram of the error detection circuit for one bit of the link. Each wire of the link is equipped with an error detection circuit.

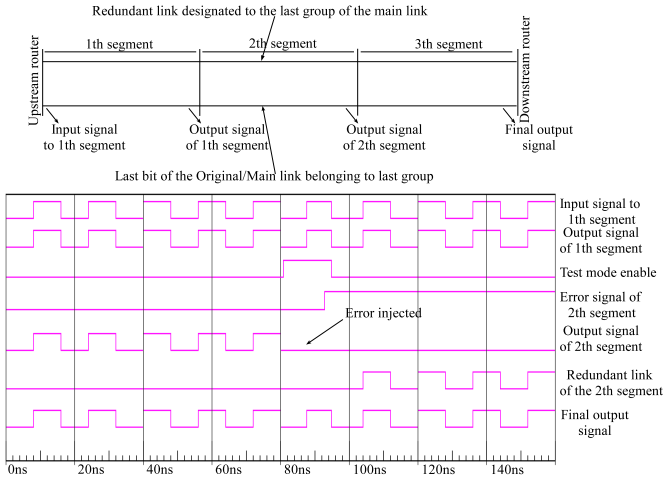

Fig. 7. Simulation result that illustrates how a segmented link recovers a faulty bit of the link. A fault is injected on 63th bit of the main link.

FPGA development board with an Virtex 5 FPGA. Numerical simulations to estimate reliability is done using Matlab [21].

Our experiments reveal that in order to keep the area overheads within reasonable limits, the number of redundant wires k should be limited to only a few. For the same reason, the number of segments m for the segmented link should be a small number. In our implementation $n = 64$ main bits and $k = 4$ redundant bits. For the segmented link, the main link is divided into $L = 4$ groups (each group has $n_i = 16$ bits) with a single redundant bit, $k_i = 1$, designated to each group. We have implemented three different variants of the segmented link structure with $m = 2, 3, 4$.

Fig.7 shows a snap-shot of our simulation of the segmented link for $m = 3$. In this simulation, we inject a fault on the second segment of the last bit (bit index 63) of the main link. As can be seen, once the fault is detected, the error signal is set to logic high permanently. The controllers reconfigure the link to replace the faulty wire with the healthy redundant wire, thereby facilitating self repairing.

A. Reliability

Numerical results based on the expressions of reliability derived in Section IV show that the proposed segmented link structure offers considerably higher reliability compared to the non-segmented link. Fig.8 shows the percentage of increase in reliability for different values of the probability of wire failure. This result is intuitive and confirms that the more segments the link has, the higher its reliability (or fault tolerance) will be. However, as we will see shortly, the increase in the number of segments results also into an increase in area overhead, power consumption, and link delay.

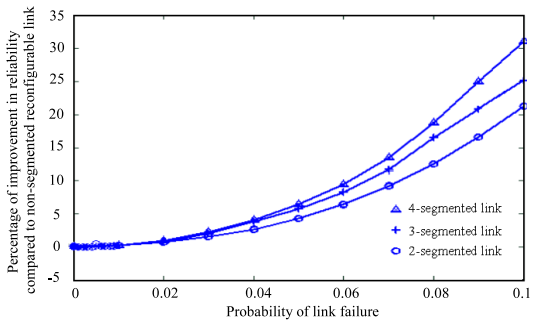

Fig. 8. Percentage of increase in reliability achieved with the proposed segmented link compared to the non-segmented link.

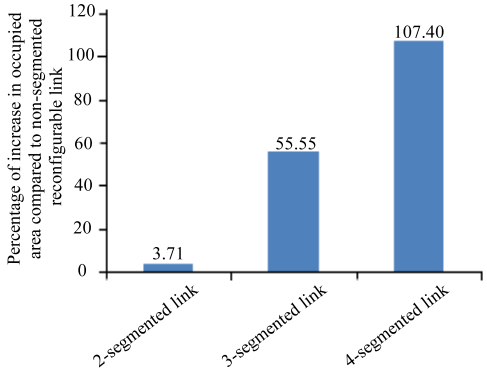

Fig. 9. Percentage of increase in area occupied by the proposed segmented link compared to non-segmented link.

B. Area

Fig.9 shows the percentage of increase in area occupied by the proposed segmented link structure compared to the non-segmented link. Notably, the area occupied by the 2-segmented link structure is only 3.71% larger than area of the non-segmented link. This is due to the considerable reduction in size of the programmable switches achieved via the proposed grouping of the main and redundant wires. Nevertheless, it can be seen that as the number of segments increases, the area overhead increases significantly compared to non-segmented link.

C. Power consumption

Fig.10 shows the comparison in terms of power consumption, estimated using Cadence tools. The power consumption of the 2-segmented link is 23.19% higher than that of the non-segmented link. It can be seen that as the number of segments increases, the amount of power consumption increases considerably too.

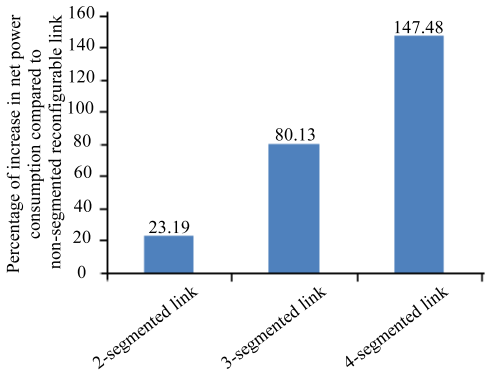

Fig. 10. Percentage of increase in power consumption of the segmented link compared to non-segmented link.

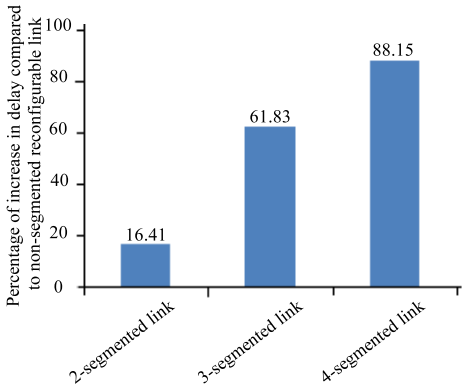

Fig. 11. Percentage of increase in link delay of the segmented link compared to non-segmented link.

D. Delay

Fig.11 shows the comparison in terms of link delay, as reported by Xilinx ISE tools. The increase in delay of the 2-segmented link is 16.41% compared to the non-segmented link. Similar to the cases of area and power comparisons, as the number of segments increases, the performance penalty increase considerably.

E. Discussion

It is clear that the best trade off between reliability improvement and penalty in area overhead, power consumption, and link delay is offered by the 2-segmented link structure. It is interesting to note that the concept of link segmentation for further improvement in fault tolerance can also be applied on top of the reconfigurable link designs presented in [15]–[17].

VII. Conclusion

We proposed link segmentation and wire grouping as a novel technique to improve the fault tolerance against permanent faults of NoC links. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. Cost effective fault detection circuits and segment level multiplexers and demultiplexers enable link reconfigurability, thereby self repairing capability. Experimental results reveal that the 2-segmented link structure offers the best trade off between reliability improvement and penalty in area overhead, power consumption, and link delay.

References

[1] L. Benini, and G. De Micheli, "Networks on chips: technology and tools," *Morgan Kaufmann*, 2006.
[2] F.A. Bower, S. Ozev, and D.J. Sorin,"Automatic microprocessor execution via self-repairing arrays," *IEEE Trans. on Dependable and Secure Computing*, vol. 2, no. 4, pp. 297-310, 2005.
[3] Z. Vasicek, L. Capka, and L. Sekanina, "Analysis of reconfiguration option for a reconfigurable polymorphic circuit," *NASA/ESA Conference on Adaptive Hardware and Systems*, 2008.
[4] S. D. Mediratta and J. Draper, "Characterization of a fault-tolerant NoC router," *IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2007.
[5] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: a reliable network for unreliable silicon," *ACM/IEEE Design Automation Conference (DAC)*, 2009.
[6] D. Fick, A. Deorio, G. Chen, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault tolerant NoCs," *ACM/IEEE Design Automation and Test in Europe Conference (DATE)*, 2009.
[7] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, "A reconfigurable fault tolerant deflection routing algorithm based on reinforcement learning for network-on-chip," *Int. Workshop on Network-on-Chip Architectures (NocArc)*, 2010.
[8] H.S. Kia and C. Ababei, "A new fault-tolerant and congestion-aware adaptive routing algorithm for regular Networks-on-Chip," *IEEE Congress on Evolutionary Computation (CEC)*, 2011.
[9] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3D Networks-on-Chip," *IEEE Int. Symposium on Quality Electronic Design (ISQED 2011)*, 2011.
[10] D. Kim, K. Lee, S.J. Lee, and H.J. Yoo, "A reconfigurable crossbar switch with adaptive bandwidth control for Networks-on-Chip," *IEEE Int. Symposium on Circuits and Systems*, 2005.
[11] B. Ahmad, A. T. Erdogan, and S. Khawam, "Architecture of a dynamically reconfigurable NoC for adaptive reconfigurable MPSoC," *NASA/ESA Conference on Adaptive Hardware and Systems*, 2006.
[12] M. Palesi, S.i Kumar, R. Holsmark, and V. Catania, "Exploiting communication concurrency for efficient deadlock free routing in reconfigurable NoC platforms," *IEEE Int. Symposium on Parallel and Distributed Processing*, 2007.
[13] R. Dafali, J. Ph. Diguet, and M. Sevaux, "Key research issues for reconfigurable Network-on-Chip," *Int. Conference on Reconfigurable Computing and FPGAs*, 2008.
[14] M. B. Stensgaard and J. Sparso, "ReNoC: a Network-on-Chip architecture with reconfigurable topology," *ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS)*, 2008.
[15] Q. Yu and P. Ampadu, "Transient and permanent error co-management method for reliable Networks-on-Chip," *ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS)*, 2010.
[16] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, 2010.
[17] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant NoC," *VLSI Design*, 2007.
[18] C. Ortega and A. Tyrrell, "Reliability analysis in self-repairing embryonic systems," *NASA/DoD Workshop on Evolvable Hardware*, 1999.
[19] http://www.cadence.com.
[20] Xilinx ISE Tools, http://www.xilinx.com.
[21] http://www.mathworks.com/products/matlab.