Chapter 15
Overhead- and Performance-Aware Fault-Tolerant Architecture for Application-Specific Network-on-Chip

Fathollah Karimi Koupaei, Ahmad Khademzadeh and Majid Janidarmian

Abstract Defect in manufacturing of integrated circuits is almost inevitable, and fast scaling in technology has caused the components of a Network-on-Chip (NoC) to be more susceptible to faults. Therefore, it is crucial to sustain chip production yield and reliable operation in the presence of defects. The permanent faults are a consequence of manufacturing defects that occur during fabrication or aging defects that occur during system lifetime. A fault-tolerant application-specific NoC should be able to detect a fault and recover the system to correctly operate the mapped application. In this paper, a fault-tolerant NoC architecture designed in VHDL and synthesized using Xilinx ISE is presented which not only is able to recover from single permanent router failure, but also improves the average response time of the system in the different traffic loads. As hardware overhead is a major issue while considering fault tolerance, a new component, called Link Interface (LI) is also developed to reduce cost overhead. The Video Object Plan Decoder (VOPD) and MPEG-4 core graphs are used as two real applications in this study.

F. Karimi Koupaei (✉)
CE Department, Arak Branch of Azad University, Arak, Iran
e-mail: karimi.fathollah@gmail.com

A. Khademzadeh
Iran Telecommunication Research Center, Tehran, Iran
e-mail: zadeh@itrc.ac.ir

M. Janidarmian
CE Department, Science and Research Branch of Azad University, Tehran, Iran
e-mail: jani@ieee.org

H. K. Kim et al. (eds.), *IAENG Transactions on Engineering Technologies*,
Lecture Notes in Electrical Engineering 170, DOI: 10.1007/978-94-007-4786-9_15,

Keywords Application-specific network-on-chip · Deadlock-free routing algorithm · Fault-tolerant design · Link interface · Mapping algorithm · Permanent failure

15.1 Introduction

The number of processor, memory and accelerator cores on systems-on-chip is rapidly increasing to support evolving standards and new applications. Computation and communication complexity is skyrocketing, and scalability-centric design paradigms are critically needed [1]. Networks-on-Chip (NoCs) have emerged as the best alternative to provide high performance in communication for futures Systems-on-Chip (SoCs) with dozens of cores integrated on a single silicon die. Mapping an application to on-chip network is the first and the most important step in the design flow as it will dominate the overall performance and cost [2]. Several approaches have been proposed in literature in the context of topological mapping in NoCs [3]. Mapping algorithms are mostly focused on 2D mesh topology which is the most popular topology in NoC design due to its layout efficiency, good electrical properties and simplicity in addressing on-chip resources. Another concern in NoC implementation is selecting an efficient routing strategy while providing freedom from deadlock. The routing algorithm determines the path that each packet follows between a source–destination pair. In the future chip generations, faults will appear with increasing probability due to the susceptibility of shrinking feature sizes to process variability, age-related degradation, crosstalk, and single-event upsets. To sustain chip production yield and reliable operation, very large numbers of faults will have to be tolerated [4, 5]. This argument strengthens the notion that chips need to be designed with some level of built-in fault tolerance. Furthermore, relaxing the requirement of 100 % correctness in the operation of various components and on-chip channels profoundly reduces the manufacturing cost as well as cost incurred by test and verification [6].

This paper is an extension of [7], and the remainder of this paper is organized as follows: in Sect. 15.2, an overview of some fault-tolerant research efforts in NoC is given. Section 15.3 illustrates the basic concepts of application-specific NoC design, and a new fault-tolerant architecture is introduced in Sect. 15.4. Simulation results i.e. average response time and hardware overhead will be presented in Sect. 15.5 followed by the concluding remarks in Sect. 15.6.

15.2 Related Work

Scaling of interconnects exacerbates the already challenging reliability of on-chip networks. As process technology scales, the integration of billions of transistors comes with an increased likelihood of failures. Smaller dimension circuits are

getting more sensitive to a particle strike, increasing the probability that the charge due to a high-energy particle flips the value in a cell [8]. With technology trends in device scaling, high clock frequencies, and supply voltage decreases, fault rates are increasing, which makes a reliable design a real challenge. Transient and Permanent errors are two main kinds of errors which generally occur in an NoC. The most common transient error recovery technique is the retransmission mechanism following error detection techniques like coding [9]. Many recently developed solutions focus on methods keeping the system working in spite the fact that some parts of the system are shut down [10]. Permanent faults in NoCs due to fabrication challenges in sub-65 nm CMOS technologies and due to wear-out underscore the need for fault-tolerant design [11]. Fault-tolerant routing algorithms are recently investigated to bypass the failed hardware. The inherent redundancy in NoCs due to multiple paths between packet sources and sinks can greatly improve communication fault resiliency [11]. Fault-tolerant routing algorithms should be able to find a path from source to destination in presence of the faults in NoC with a certain degree of tolerance [12]. Many algorithms in this area have been proposed which follow their own optimization aims. To name just a few, in [11], the authors propose a novel low-overhead neighbor aware, turn model based fault-tolerant routing scheme (NARCO) for NoCs which combines threshold-based replication in network interfaces, a parameterizable region-based neighbor awareness in routers, and the odd–even and inverted odd–even turn models. Shi et al. [13] presents the scalable and fault-tolerant distributed routing (SFDR) mechanism. It supports three routing modes including corner-chains routing, boundary-chains routing and fault-ring routing. Inspired by divide-and-conquer concept, system is partitioned into nine regions. Each region promises fault-tolerance of one's own when packets bounded into its area to guarantee total fault tolerance. The main problem with the fault-tolerant routings is that if a router fails, considering mesh architecture, recovery cannot be accomplished only by rerouting. In addition, hardware redundancy is inevitable in order to repair the lost connection to the network of the core directly connected to the failed router. A fault-tolerant mesh-based NoC architecture with the ability of recovering from single permanent failure is presented in [14]. This method adds a redundant link between each core and one of its neighboring routers, resulting in significant improvement in reliability while has little impact on performance. In this architecture, only one spare router should be selected among all possible alternative ones. This has an influential effect on overall performance in terms of the average response time and reliability of the system. Regarding to this work, in [15] a new fast and optimum algorithm based on performance measurement and extra communication cost is proposed to find the best configuration that also results in a more reliable system. It also shows that mapping algorithm has a great impact on mentioned parameters. Following this concept, in this paper, a hardware and performance-aware design for the fault-tolerant NoC architecture is presented which takes into account the specific application mapped onto mesh topology.

15.3 Prerequisites of Application-Specific NOC Design

15.3.1 Mapping Problem

To formulate mapping problem in a more formal way, we need to first introduce the following two concepts borrowed from [16]:

Definition 1 The core graph is a directional graph $G(V, E)$ whose each vertex $v_i \in V$ shows a core, and a directional edge $e_{i,j} \in E$ illustrates connection between v_i and v_j. The weight of $e_{i,j}$ that is shown as $comm_{i,j}$, represents the communication volume from v_i to v_j . The IP core along with a router connected to it by Network Interface (NI) is displayed as a tile.

Definition 2 The NoC architecture graph is a directional graph $A(T, L)$, whose each vertex $t_i \in T$ represents a tile in the NoC architecture, and its directional edge that is shown by $l_{i,j} \in L$ shows a physical link from t_i to t_j.

The core graph mapping $G(V, E)$ on NoC architecture graph $A(T, L)$ is defined by a one to one mapping function.

$$map : V \rightarrow T, s.t.map(v_i) = t_j, \forall vi \in V, \exists t_j \in T, |V| \leq |T|$$

Mapping algorithms are mostly focused on 2D mesh topology (Architecture Graph) which is the most popular topology in NoC design due to its suitability for on-chip implementation and low cost. The definitions are presented in Fig. 15.1.

15.3.2 Routing Algorithm

The routing algorithm determines the path that each packet follows between a source-destination pair. Routing algorithms noticeably affect the cost and performance of NoC parameters i.e. area, power consumption and average message latency [17]. Due to determined sources and destinations in application-specific NoC, minimum-distance routing algorithms are mostly considered in this area which the routing is computed off-line and admissible paths stored into the routing tables.

In general, every routing algorithm should include deadlock freedom feature [18]. So channel dependency graphs (CDG) is used to avoid any possible deadlocks. The CDG is a directed graph with the network channels as the vertices and the direct dependencies between the two channels as the edges. A dependency exists between the links $l_{i,j}$ and $l_{j,k}$ whenever there is a path to route packets from v_i to v_k through v_j which uses those links. An extension to CDG as a sub graph is the concept of application specific channel dependency graph (ASCDG) introduced in [19]. The ASCDG is a sub graph of the CDG and an edge in CDG between channels, $l_{i,j}$ and $l_{j,k}$ is removed if there was no application-specific dependency

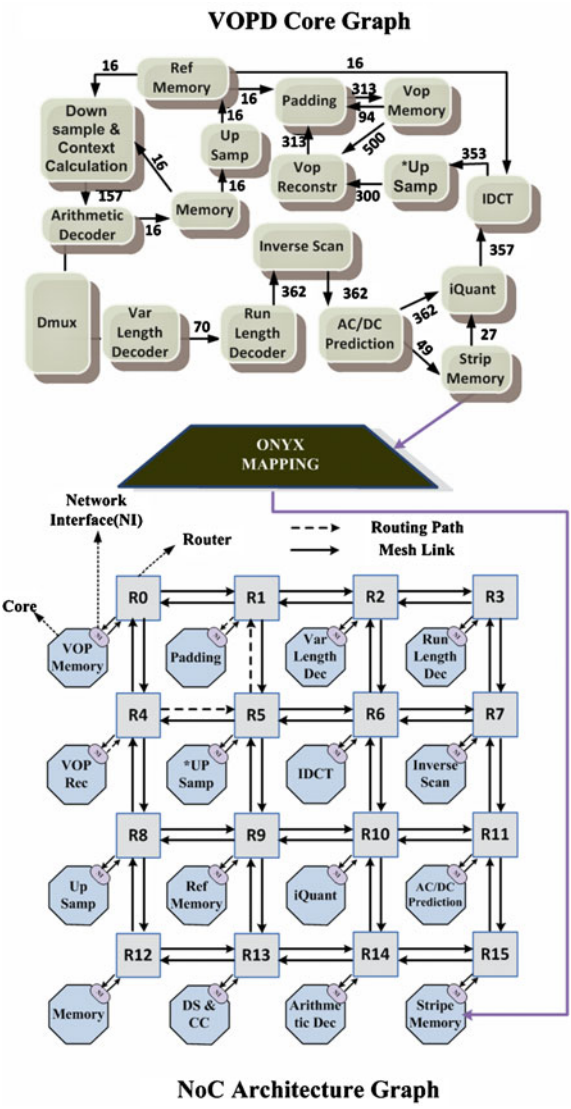

Fig. 15.1 Mapping problem concepts

between $l_{i,j}$ and $l_{j,k}$. Deadlock is inevitable when there are any cycles through ASCDG graph. A cycle in the ASCDG is a succession of application specific direct dependencies, $D = \{d_1, d_2, \cdots, d_n\}$, where $d \in D$ is a pair $(l_{i,j}, l_{j,k})$ with $l_{i,j}, l_{j,k} \in L$. If there is any cycle, we need to break it. By removing a dependency, all of the corresponding paths to that dependency will be removed. Using this method guarantees that routing algorithm is still deadlock-free. It is worth noting that using deterministic routing algorithm and efficient mapping algorithm, a few existing cycles can be easily broken.

Although the proposed methodology is topology and application agnostic, the state-of-art mapping algorithm proposed in [18] is used to map Video Object Plan Decoder (VOPD) cores onto mesh topology. We have also used two minimum-distance routing algorithms in mesh i.e. XY and YX. In XY (YX) routing algorithm, a packet is routed first in the X(Y) direction and then along the perpendicular Y(X) dimension. Because of using both algorithms together to route and reroute the packets, deadlock problem which is easily solved by described ASCDG graph should be taken into consideration.

15.4 The Proposed Fault-Tolerant Application-Specific Architecture

In the mesh-based architecture which is the simplest and most dominant topology for today's regular network-on-chips, each core is connected to a single router as depicted in Fig. 15.1. If a failure occurs in a router in this topology, the failed router cannot be used any more for routing packets and the directly connected core obviously loses its communication with the network, so the expected requirements of the mapped application are not satisfied and the whole system breaks down. As shown in Fig. 15.1, each router consists of five identical input/output ports and each port is a bidirectional link with a circular FIFO on its input side. In order to recover the inaccessibility of the core, a fault-tolerant architecture which each core is connected to two routers i.e. main and spare have been proposed [14] shown in Fig. 15.2. The spared router is used while a permanent fault is detected in the main one and in this situation; average response time of the system degrades due to rerouting phase. All essential modifications in Network Interface (NI), routers and packets were explained in [14]. The authors also supposed that fault detection is done by self-testing facilities embedded in each router which is responsible for continuously testing its operation. A router can inform the neighbors about its faulty status by setting a fault_status flag. This flag is checked by the neighboring routers and cores before starting any communication with the router.

Adding one port to all non-edge routers in this architecture leads to a great waste of hardware. Although it is able to tolerate one permanent router failure, it does not exploit the full potential of the architecture. However, in this paper the provided path diversities between adjacent routers are used to improve performance of the system and a new component called Link Interface (LI) instead of router port is developed to reduce hardware overhead.

According to the mentioned architecture shown in Fig. 15.2, after mapping an application onto mesh topology, an efficient spare router selection should be considered to find the best spare router for any core. To minimize the hardware cost, only one of the possible spare routers for a core is chosen [14]. On the other hand, with respect to the possible places for each core as shown in Fig. 15.2, there are two constraints to select a spare router.

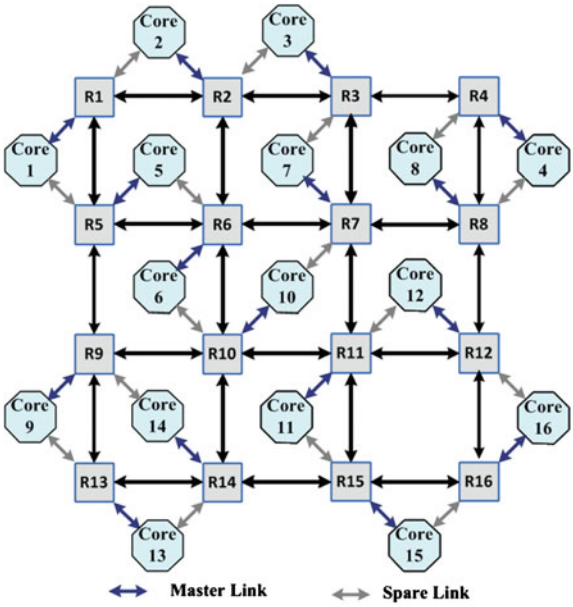

Fig. 15.2 The proposed architecture in [14]

1. Each router is limited to be linked as a spare by only one core.
2. Each core is located in the neighborhood of its local and adjacent routers and all cores should be placed in different locations.

The spare router selection algorithm in [15] called FERNA results in better average response time, extra communication cost and system reliability than greedy algorithm and has a polynomial time complexity. Since both main and spare routers can be used at the same time in the developed solution, we need to modify FERNA algorithm with regards to more routing path opportunities (explained in details in Fig. 15.3).

As you can see in Fig. 15.4, in the proposed architecture, each core is connected to its router via main local port and to the links using Link Interface via backup local port. The architecture of LI will be discussed in the following subsection.

In this architecture, if both main and spare routers are working properly, the best (minimum-distance) paths to send and receive packets are derived from path diversities and if main (spare) is faulty, spare (main) will be responsible to transfer packets through rerouting paths. Because this architecture is supposed to recover from only single permanent failure, all the best and rerouting paths are easily found while are deadlock-free by applying the ASCGD concept. For example, the ASCDG graph has been drawn in Fig. 15.5 while all routers are working properly. All admissible paths are offline stored into the routing tables and used with regards to routers conditions.

```
Initialize (G (E, V));
For (All Routers)
   Router_Unused[i] =1;
Do
{
   Selected_Core=Find_Max_Comm (G (E, V));
   Available_Palces=Find_All_Available_Places (Selected_Core);
   K=1;
   For (All Neighbor Routers)
      If (Router_Unused[i] =1)
      {
         Attach (Router[i], Selected_Core (Backup_Port));
         Response_Time[k] =Calculate_Response_Time (Architecture);
         Detach (Router[i], Selected_Core (Backup_Port));
         K=K=1;
      }
   Selected_Router=Find_Best_Neighbor_Router (Response_Time []);
   If (Selected_Router_Unused_Port=True)  // Edge Routers
         {
      Attach (Router[i], Selected_core (Backup_Port));
      Router_Unused [Selected_Router] =0;
         }
   Else //Non-Edge Router
            Attach (Link_Interface, Selected_Core (Backup_Port));
   Update_Available_Places;
} While (Find Spare Router For All Cores);
```

Fig. 15.3 The pseudo code of spare router selection algorithm

15.4.1 Link Interface

In the previous design [14], it is necessary to add one port to all non-edge routers resulting in much hardware redundancy. In order to reduce the overhead, in this section a Link Interface is suggested. After entrance of header flit into this module, destination address is decoded. As an example, if the address shows that connected core is the destination, header and its following flits will be sent to backup network interface of the core, otherwise they are routed to another output towards next router.

This module has been implemented with three processes which run concurrently; therefore it is able to transmit three dataflow as shown in Fig. 15.6. This module has been also designed without using clock pulse that leads us to achieve better response time and power consumption. To this end, as soon as data_ready line is activated; the input port will run its process of management and data control flow to guide flits towards output port. It is worth noting that if two input ports

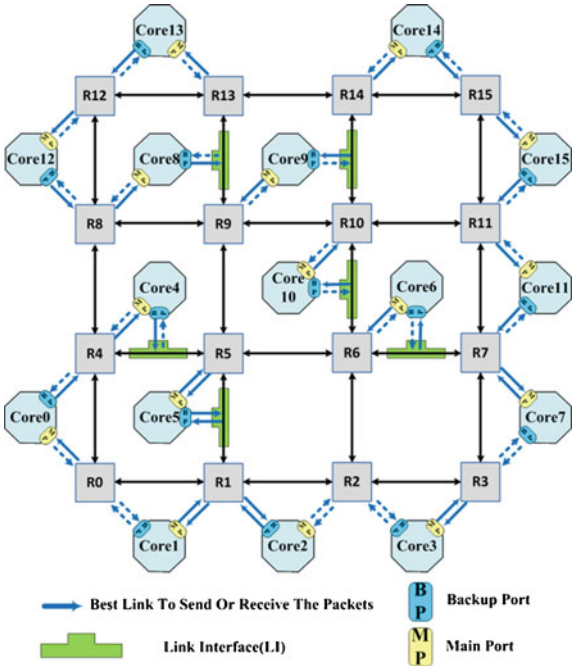

Fig. 15.4 The proposed fault-tolerant architecture in this paper

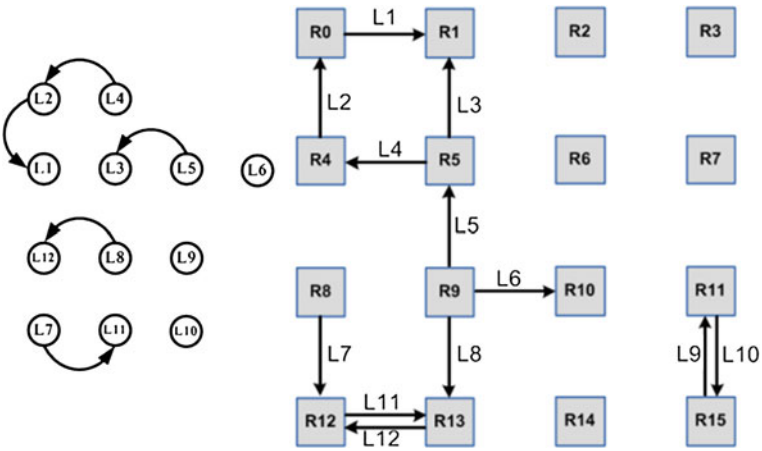

Fig. 15.5 The ASCDG when all routers are working properly

simultaneously request one output port, priority mechanism is used to tackle with this problem (Fig. 15.7). In order to prevent overwriting, a flag has been also considered for each output port to inform its free or busy status.

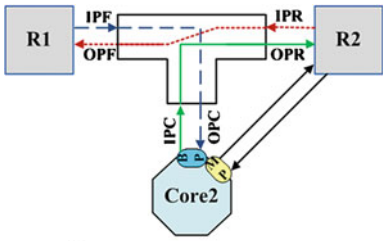

Fig. 15.6 The proposed link interface

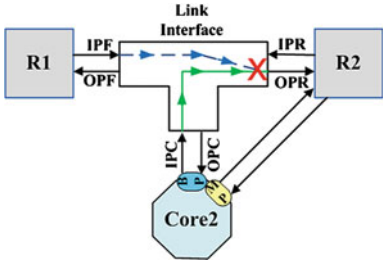

Fig. 15.7 The possible conflict in the link interface

15.5 Experimental Results

In order to compare the average response time and hardware cost of the reliable architecture and traditional mesh, they have been designed in VHDL and synthesized using Xilinx ISE (on FPGA—Xilinx VitexE). Janidarmian et al. [15] shows the effect of mapping algorithm on system reliability, so Video Object Plan Decoder (VOPD) and MPEG-4 as two case studies have been mapped on a 4 × 4 mesh topology using the best mapping technique proposed in [18]. For our experiments, packets are generated according to a uniform distribution with the rates extracting from communication volumes in the VOPD or MPEG-4 core graphs. It means that the number of packets generated in the source cores are derived from the core graph edges and in order to increase the traffic load, communication volumes are multiplied by a traffic factor *i*.

As can be seen in Fig. 15.8, it takes 137,265 cycles to reach all packets to the destination routers in the traditional mesh when the traffic factor *i* is 1 and VOPD application is considered. As mentioned before, the proposed architecture is able to tolerate one router failure and guarantees the 100 % packet delivery.

Because it potentially is possible both main and spare routers are used to send or receive packets by each core, the proposed architecture also significantly decrease the average response time on the faultless and 16 possible faulty routers compared with the mesh architecture as illustrated in Fig. 15.8. It should be pointed that it actually is a great achievement to develop a fault-tolerant NoC design which also has better performance. To explain in details, when all routers are correctly operating, new architecture improves the average response time by 41 % comparing to mesh. The worst observed average response time (123,377 cycles) occurs when the eleventh router fails, and in this case it also improves the

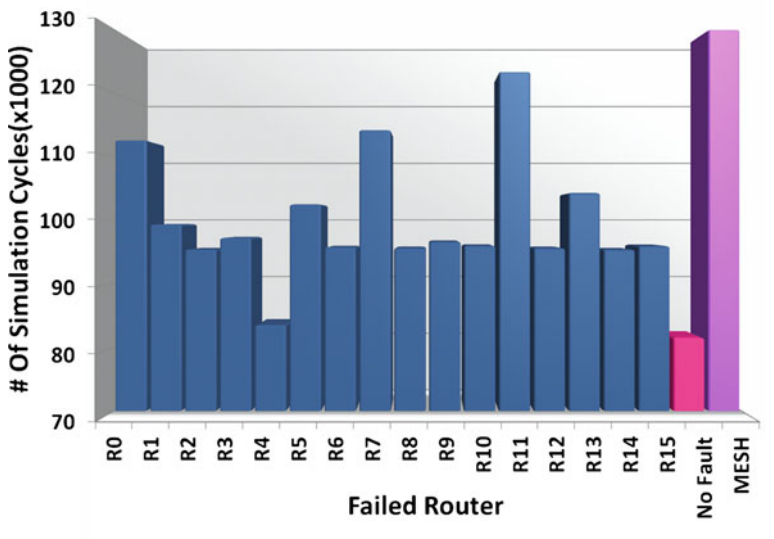

Fig. 15.8 Number of simulation cycles when all packets are received by the destinations (traffic factor = 1)

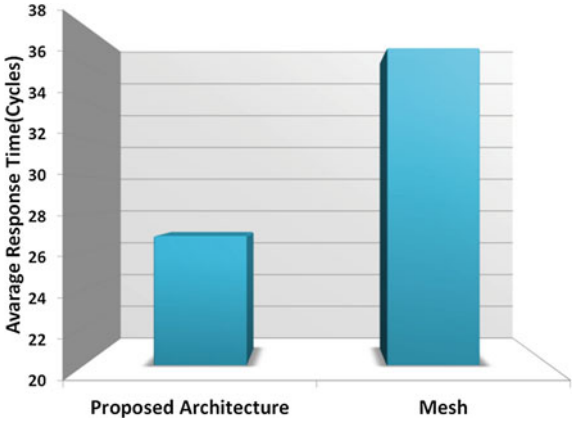

Fig. 15.9 Comparing proposed architecture with mesh in terms of average response time (all situations are considered)

response time by 10 %. Conclusively, we observe that the proposed approach allows to decrease the response time of system by 27 % and tolerate permanent failure of each single router (Fig. 15.9).

To better estimation, the proposed architecture has been simulated in different traffic loads form traffic factor 1–10 for VOPD core graph. As shown in Figs. 15.10 and 15.11, much more improvements are achieved for higher traffic loads even when the worst scenario happens. The worst case (failure of the router 7) signifies that we have the minimum improvements in this situation while comparing performance in all traffic loads.

Traffic Load	Mesh - XY Routing	Best Case (No Fault)		Worst Case (R7 Failed)	
		Proposed Arch	Improvement	Proposed Arch	Improvement
Traffic 1X	**36.79**	**21.83**	**40.66%**	**33.07**	**10.11%**
Traffic 2X	38.03	22.16	41.73%	33.90	10.86%
Traffic 3X	**40.47**	**23.40**	**42.18%**	**35.43**	**12.45%**
Traffic 4X	49.65	27.16	45.30%	42.38	14.64%
Traffic 5X	**58.36**	**29.30**	**49.79%**	**47.79**	**18.11%**
Traffic 6X	72.50	34.23	52.79%	57.58	20.58%
Traffic 7X	**86.92**	**34.82**	**59.94%**	**66.96**	**22.96%**
Traffic 8X	115.92	37.26	67.86%	78.56	32.23%
Traffic 9X	**154.83**	**45.24**	**70.78%**	**102.85**	**33.57%**
Traffic 10X	285.31	53.07	81.40%	173.33	39.25%

Fig. 15.10 Comparing proposed architecture with mesh in terms of average response time in the different traffic loads

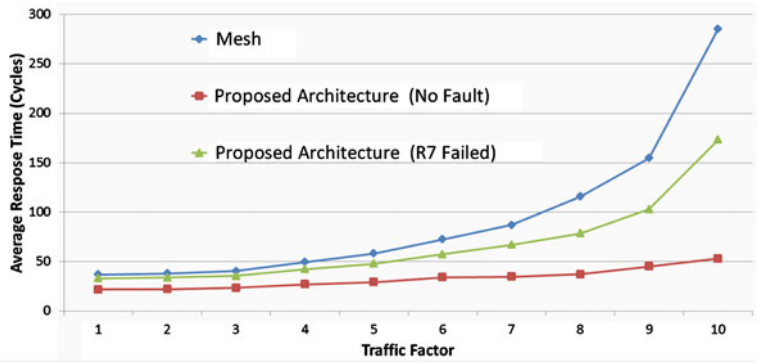

Fig. 15.11 Comparing proposed architecture with mesh in terms of average response time in the different traffic loads

This procedure has been also applied on MPEG-4 core graph shown in Fig. 15.12 and obtained results while traffic factor i is equal to 1 are depicted in Fig. 15.13. Without loss of generality, the edges of the graph are not considered bidirectional and in this case, the improvement of average response time is about 23 % compared to mesh.

To recover from a permanent fault, hardware redundancy is mandatory and reduction of this overhead has always been an important issue in this area. In our design, we do not add any router port contrary to what [14] does and instead a link interface was developed which helps to achieve less hardware overhead.

The router port and LI have been designed and implemented in the VertexE FPGA (v50ecs144-6). Synthesized results (Fig. 15.14) indicate that LI overhead translates to approximately 32 % of the router port area. Therefore, the proposed fault-tolerant architecture (with only 6 LI) introduces better overhead compared to previous work in the literature.

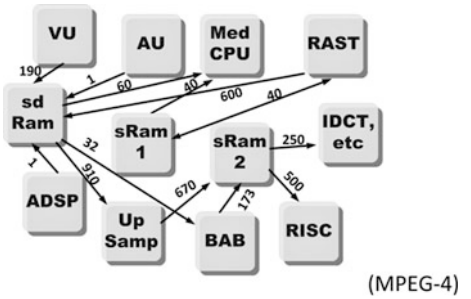

Fig. 15.12 MPEG-4 core graph

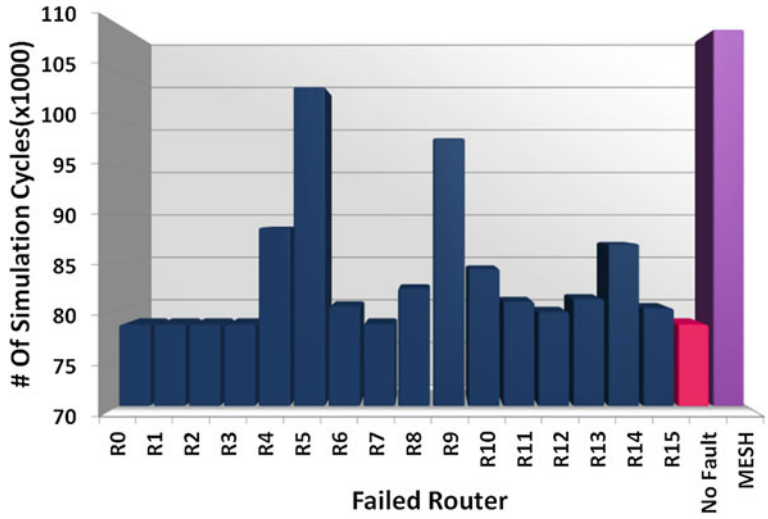

Fig. 15.13 Number of simulation cycles when all packets are received by the destinations (traffic factor = 1)

15.6 Conclusion

In this paper, a new fault-tolerant application-specific network-on-chip was proposed which is able to tolerate one router failure and guarantees the 100 % packet delivery. Considering fault tolerance in designing forces us to accept performance degradation. However, this architecture also improves the average response time of system by 27 and 23 % comparing to traditional mesh for VOPD and MPEG-4 applications, respectively. It was also shown that our architecture obtains more improvements in terms of performance in the higher traffic loads.

Link Interface as a solution for reducing hardware redundancy was suggested and synthesized results demonstrated that each new router port is almost equal to

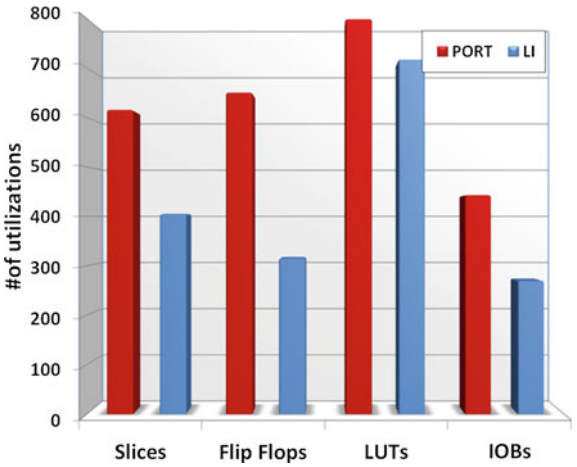

Fig. 15.14 Hardware overhead of the router port and link interface (LI)

three Link Interfaces in terms of hardware overhead. Although the proposed methodology is topology and application agnostic, best mapping algorithm to map Video Object Plan Decoder (VOPD) and MPEG-4 cores onto 2D mesh topology was simulated and investigated.

References

1. Benini L (2006) Application specific NoC design, date. In: Proceedings of the design automation & test in Europe conference, vol 1. p 105
2. Shen W, Chao C, Lien Y, Wu A (2007) A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network, NOCS. In: First international symposium on networks-on-chip (NOCS'07), pp 317–322
3. RoshanFekr A, Janidarmian M, Samadi Bokharaei V, Khademzadeh A (2011) Yield enhancement with a novel method in design of application-specific networks on chips. Electr Eng Appl Comput 90:247–257
4. Furber S (2008) The future of computer technology and its implications for the computer industry. Comput J 51(6):735–740
5. Borkar S (2005) Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. IEEE Micro 25(6):10–16
6. Dumitraş T, Mărculescu R (2003) On-chip stochastic communication, date. In: Design, automation and test in Europe conference and exhibition (DATE'03), vol 1. p 10790
7. Karimi Koupaei F, Khademzadeh A, Janidarmian M (2011) Fault-tolerant application-specific network-on-chip. Lecture notes in engineering and computer science: proceedings of the world congress on engineering and computer science 2011, WCECS 2011, San Francisco, USA, pp 734–738, 19–21 Oct 2011
8. Shivakumar P, Kistler M, Keckler SW, Burger D, Alvisi L (2002) Modeling the effect of technology trends on the soft error rate of combinational logic. dsn. In: International conference on dependable systems and networks (DSN'02), p 389
9. Ali M, Welzl M, Hessler S, Hellebrand S (2007) An Efficient fault tolerant mechanism to deal with permanent and transient failures in a network on chip. Int J High Perform Sys Archit 1(2):113–123

10. Rantala V, Lehtonen T, Liljeberg P, Plosila J (2009) Multi network interface architectures for fault tolerant network-on-chip, isscs. International symposium on signals, circuits and systems. pp 1–4
11. Zou Y, Pasricha S (2010) NARCO: neighbor aware turn model-based fault tolerant routing for NoCs. Embed Sys Lett IEEE 2:85–89
12. Dumitras T, Kerner S, Marculescu R (2003) Towards on-chip fault-tolerant communication. In: Proceedings of the Asia and South Pacific design automation conference
13. Shi Z, You K, Ying Y, Huang B, Zeng X, Yu Z (2010) A scalable and fault-tolerant routing algorithm for NoCs, iscas. International symposium on circuits and systems (ISCAS), 30 May 2010–2 June 2010, pp 165–168
14. Refan F, Alemzadeh H, Safari S, Prinetto P, Navabi Z (2008) Reliability in application specific mesh-based NoC architectures. On-line testing symposium, 2008. IOLTS apos; 08. 14th IEEE international, pp 207–212
15. Janidarmian M, Tinati M, Khademzadeh A, Ghavibazou M, RoshanFekr A (2010) Special issue on a fault tolerant network on chip architecture. AIP Conf Proc 1247:191–204
16. Janidarmian M, Khademzadeh A, Tavanpour M (2009) Onyx: a new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip. IEICE Electron Express 6(1):1–7
17. Janidarmian M, Samadi Bokharaie V, Khademzadeh A, Tavanpour M (2010) Sorena: new on chip network topology featuring efficient mapping and simple deadlock free routing algorithm. 2010 10th IEEE international conference on computer and information technology, pp 2290–2299
18. Janidarmian M, RoshanFekr A, Samadi Bokharaei V (2011) Application-specific networks-on-chips design. IAENG Int J Comp Sci 38(1):16–25
19. Palesi M, Holsmark R, Kumar S (2006) A methodology for design of application specific deadlock-free routing algorithms for NoC systems. Hardware/software codesign and system synthesis, CODES + ISSS '06. In: Proceedings of the 4th international conference, pp 142–147, Oct 2006