A Fault Tolerant NoC Architecture Using Quad-Spare Mesh Topology and Dynamic Reconfiguration

Yu REN[1], Leibo LIU[1*], Shouyi YIN[1], Jie HAN[2], Qinghua WU[1], Shaojun WEI[1]
[1] Institute of Microelectronics and the National Lab for Information Science and Technology, Tsinghua University, Beijing 100084, China
[2] ECE Department, University of Alberta, Edmonton, Canada T6G 2V4
Email: liulb@tsinghua.edu.cn

Abstract

Network-on-Chip (NoC) is widely used as a communication scheme in modern many-core systems. To guarantee the reliability of communication, effective fault tolerant techniques are critical for an NoC. In this paper, a novel fault tolerant architecture employing redundant routers is proposed to maintain the functionality of a network in the presence of failures. This architecture consists of a mesh of 2×2 router blocks with a spare router placed in the center of each block. This spare router provides a viable alternative when a router fails in a block. The proposed fault-tolerant architecture is therefore referred to as a quad-spare mesh. The quad-spare mesh can be dynamically reconfigured by changing control signals without altering the underlying topology. This dynamic reconfiguration and its corresponding routing algorithm are demonstrated in detail. Since the topology after reconfiguration is consistent with the original error-free 2D mesh, the proposed design is transparent to operating systems and application software. Experimental results show that the proposed design achieves significant improvements on reliability compared with those reported in the literature. Comparing the error-free system with a single router failure case, the throughput only decreases by 5.19% and latency increases by 2.40%, with about 45.9% hardware redundancy.

Keywords Fault tolerant; network-on-chip; quad-spare mesh; dynamic reconfiguration

I. Introduction

The advance of microchip technologies has led to a rapid increase in the number of processing elements (PEs) on a single chip. A variety of interconnection schemes have been proposed, including crossbars, rings, buses, and NoC (Network-on-Chip) [1]. The traditional bus is built on well understood concepts and is easy to model. In a highly interconnected multi-core system, however, it suffers from the issue of scalability. As more PEs are added to it, the capacitive load grows rapidly, leading to a dramatic performance degradation. Thus they are not considered appropriate for systems with more than ten nodes [2]. A crossbar overcomes some of the limitations of buses, but it is not ultimately scalable and is only an intermediate solution [3]. The packet-based Network-on-Chip is scalable and has been widely used to decouple communication from computation, thus improving performance. It is considered a promising solution to the interconnection challenges of future SoC designs [4].

The reliability of an NoC is critical to guarantee reliable communication. With increasing fault rates and chip densities, the reliability of an NoC has increasingly become a challenge. Many solutions have been proposed to sustain the reliability of a system, including fault tolerant routing algorithms [5,6] and various topologies for implementing the communication infrastructure [7,8]. These methods, however, do not make use of the good PEs when there is an error in the network. Packets are transported through one or more routers. A faulty network component may prevent healthy PEs from gaining access to the rest of the system or, even worse, prevent the entire system from operating reliably. For example, Fig. 1 shows four healthy PEs isolated by faulty routers and broken links. They cannot communicate with the other part of the network even though they could work correctly, thus leading to a waste of hardware resources because of the higher cost of a PE than that of a router or link.

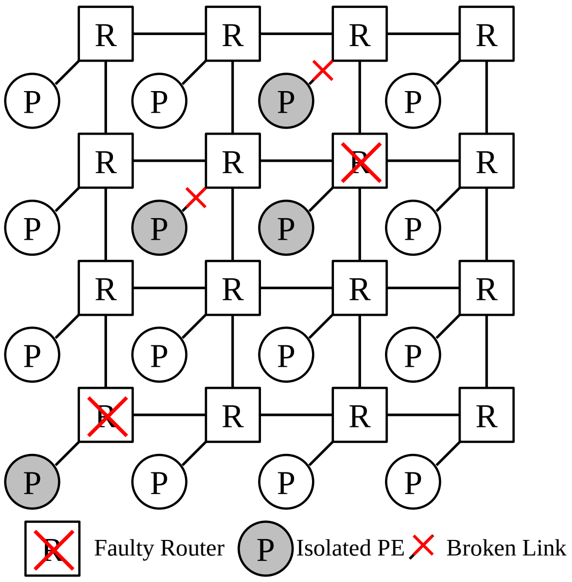


Fig. 1. Healthy PEs are isolated by faulty routers and broken links.

In this paper, a novel router-level redundant scheme, referred to as a quad-spare mesh, is proposed to solve the problem of isolated PEs by maintaining the network connectivity and thus functionality in the presence of faulty routers. In the quad-spare mesh, an NoC is divided into 2×2 blocks. One spare router is placed in the center of each block, providing connectivity between fault-free NoC routers and healthy PEs in the presence of a faulty router. Once a faulty router is detected, network reconfiguration is performed dynamically through control signals without altering the underlying topology. The rest of the system does not have to stop working during reconfiguration. Therefore, this process is transparent to the upper layer, e.g. operating system and application software, so it can be easily implemented in the current hierarchical hardware/software system designs. The design is then evaluated with fault tolerance metrics including system reliability and mean time to failure (MTTF). Simulation results show that this scheme has achieved significant improvements by these metrics compared with those reported in the literature. For a 10×10 NoC, the MTTF of the proposed scheme is 40.0% higher than that of the design in [20]. In an NoC with a single error, the maximum throughput only decreases by 5.19% and the average latency increases by 2.40%, compared with the error-free case. The quad-spare NoC is implemented with TSMC 65nm technology and the result shows that about 45.9% additional hardware resources are needed to implement the proposed fault-tolerant architecture.

The rest of this paper is organized as follows: Section II reviews related prior work on fault tolerant NoC designs. Section III describes the proposed fault tolerant NoC architecture. In

Section IV, evaluation and experimental results are presented. Finally, Section V concludes this paper.

II. Related Work

Faults in semiconductor devices fall into three main categories: permanent, intermittent, and transient [9]. Permanent faults are usually caused by manufacturing defects, aging effects, and/or physical damages to the resources that generate or transport data. One approach to dealing with permanent faults is fault tolerant routing, which involves isolating the entire router [10,11] or a few ports of a router [12]. Another method for permanent fault tolerance is to use spare components to replace defective elements [20]. Intermittent faults usually occur repeatedly at the same location and tend to occur in burst. They are more difficult to handle since subsequent errors can interrupt recovery from the effect of previous ones. Transient faults are usually caused by neutron and alpha particles, power supply and interconnect noise, electromagnetic inference and electrostatic discharge. Error detecting/correcting codes [13,14] are pervasively used to handle these errors. Our work focuses on dealing with permanent faults, which can be detected by add-on mechanisms such as Built-In Self-Test or periodic run-time testing. The proposed scheme could also be combined with error detecting/correcting codes for tolerating transient faults.

Several fault tolerant routing algorithms have been proposed. In [5] and [15], random walk and probabilistic flooding algorithms are presented; faulty links or congested routers are avoided in these algorithms. They are based on generating a number of packet copies in the flooding stage and sending them to the destination. These algorithms increase the load in a network and cause congestion, thus decreasing the throughput. Furthermore, they do not address the aforementioned problem of isolated PEs.

With the ever increasing circuit density and available hardware resources on a chip, redundancy techniques have been widely utilized for achieving fault tolerance. Redundancy technique can be applied at different levels. Previous attempts include introducing microarchitecture-level redundancies [16,17,18], core-level redundancies [19] and router-level redundancies [20]. One simple rule about the use of redundant resources is that the simpler it is, the less likely it is to suffer a failure. The microarchitecture-level redundancy is appropriate for use in multi-core chips (e.g., a quad-core processor) to keep the overhead low. As the size of a mesh increases and the cost of a single router becomes relatively inexpensive compared with the entire NoC, microarchitecture-level redundancy is considered inefficient [19]. As a result, it is reasonable to provide redundancy at the router level [20]. In the router-level redundant design proposed in [20], each column has a common spare router located on the top row, which is shared by the routers of the column. This method works well, but the reconfiguration is complicated and cannot be performed dynamically. Hence, there is still room for improvement. The router-level redundant design proposed in this paper can be dynamically reconfigured and has a better reliability.

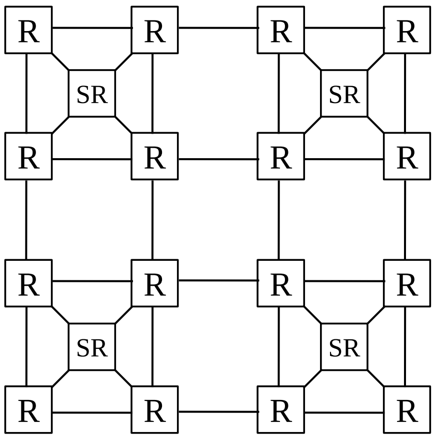
III. Design of the Quad-spare Mesh

In this section, the proposed network topology with spare routers, i.e., the quad-spare mesh, is first presented. The topology reconfiguration algorithm is then developed to maintain the correct function of the system in the presence of faulty routers. This topology reconfiguration can work dynamically, i.e., the faulty routers can be replaced by spare ones while the NoC is working.

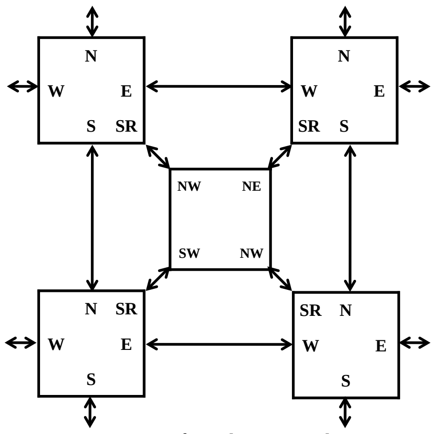
Finally, the routing algorithm based on a bypassing mechanism is introduced.

3.1 Quad-spare Mesh Topology

The topology of the quad-spare mesh is shown in Fig. 2(a). The original 2D mesh is partitioned into blocks. Each block has an array of 2×2 routers, with one spare router placed in the center of the block. The spare router is shared by the four routers in the block and can replace any of them if one becomes faulty. Hence, this design is referred to as a quad-spare mesh. Each router has six directions, namely N/E/S/W/SR/C, representing the north/east/south/west/spare router/PE respectively. The communication path between routers and the spare router in one block is shown in Fig. 2(b).



R-Router; SR-Spare Router



N/E/S/W/SR-Direction of North/East/South/West/Spare router

Fig. 2. (a) Quad-spare mesh topology (b)Communication path between routers

The communication path between a PE and a router/spare router in one block is shown in Fig. 3. Communication channels are divided into input and output channels. For clarity, the input and output channels of PEs in one block are shown in Fig.4 (a) and (b) respectively. The input port of each PE is connected to the original router and spare router by a 2-to-1 MUX. As to the output port, each PE is also connected to the original router and spare router. Because the spare router has only one input port, a 4-to-1 MUX is needed to select the PE whose router is replaced by the spare router.

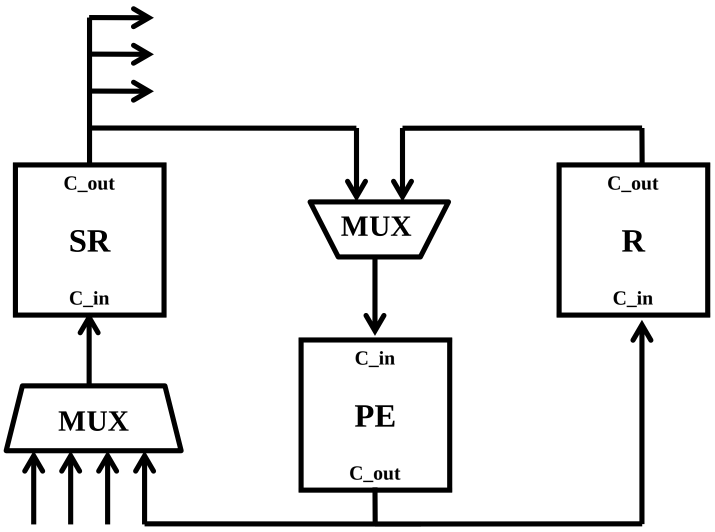


Fig. 3. PE connections in a block

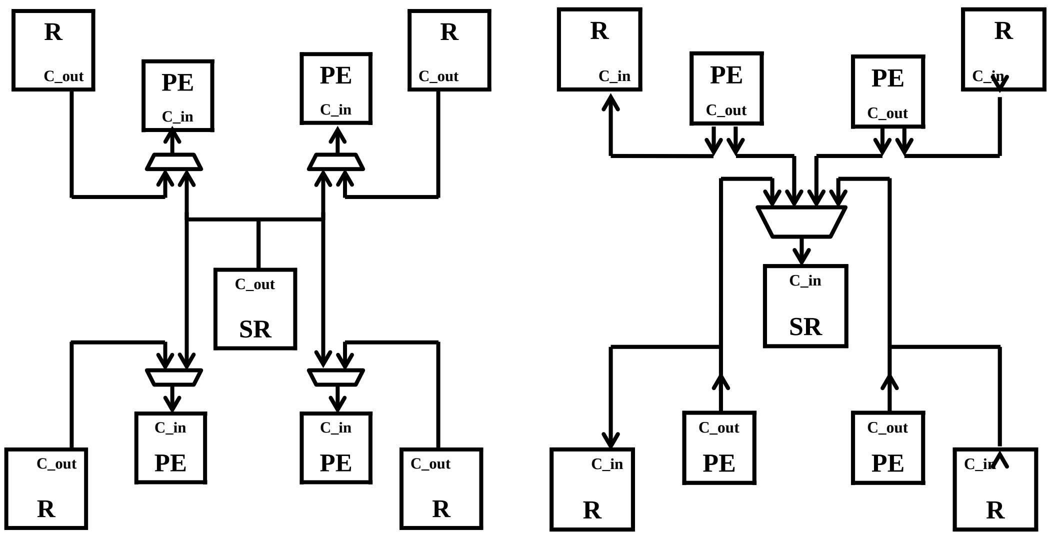


Fig. 4. (a) Input channels of PEs (b) Output channels of PEs

3.2 Topology Reconfiguration

The reconfiguration works dynamically. Faults can be captured by add-on mechanisms such as the Built-In-Self-Test (BIST) or periodic online testing of circuits [21,22]. When an error is detected, this information will be sent to the centralized topology-reconfiguration controller, so that adjacent routers will know which of their neighbors is faulty and then packets will be sent to spare routers. The topology of the NoC and the routing algorithm, which will be described in the next subsection, seem the same to the upper layer, such as operating system and application software. Assuming no delay, the system will not have to stop working to reconfigure the network. As a result, the proposed method is transparent to the upper layer, which is a great advantage.

Fig. 5 shows an example of a 4×4 NoC reconfiguration. In this example, the routers are numbered from 0 to 15. Routers 5 and 14 are faulty and are thus replaced by the spare routers in their blocks respectively. That is, Router 5 is replaced by SR0 and Router 14 is replaced by SR3. For paths between a spare router and routers in other blocks (e.g., SR0 to R6 and R9), a bypass mechanism is used. The controller connects the path from R6 to R5 and R5 to SR0 together, so that packets from R6 can be sent to the spare router, and the path between R9 to SR0 is the same. While for packets sending from SR0 to R6 or R9, a switch is used. The bypass switch is located outside the router. The switch works together with the spare router. The selection signal of the switch is generated by the routing module of the spare router. When an error occurs and a spare router is used, the switch functions as the faulty router. Because there are only two choices when a spare router routes packets to adjacent blocks, a single switch would meet the need. Connecting paths between routers and PEs have been shown in Figs. 3 and 4. They can be reconfigured by changing the selection signals on the 2-to-1 MUX and 4-to-1 MUX by controller.

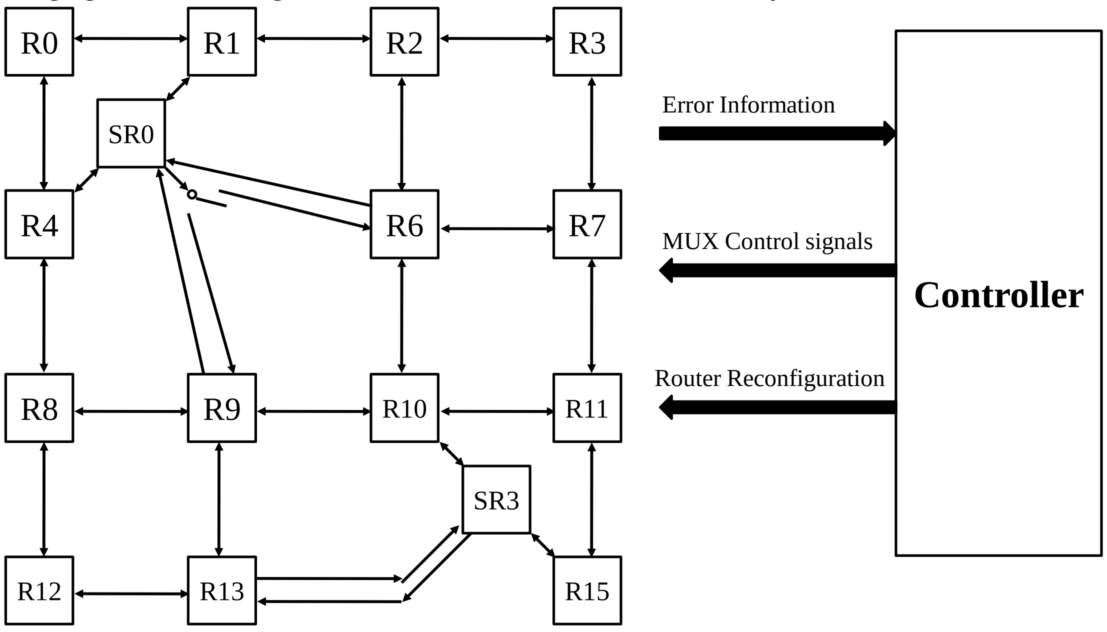


Fig. 5. An NoC reconfiguration example. For clarity, PEs, faulty routers and unused interconnections are not shown in the figure.

3.3 Routing Algorithm

A routing algorithm is indispensible in the topology reconfiguration. In the XY routing algorithm, a routing table can be easily obtained when given the current position and destination of packets, as shown in Table 1.

Table 1. Routing Table*

	$Y_{dest} > Y_{current}$	$Y_{dest} < Y_{current}$	$Y_{dest} = Y_{current}$ $X_{dest} > X_{current}$	$Y_{dest} = Y_{current}$ $X_{dest} < X_{current}$	$Y_{dest} = Y_{current}$ $X_{dest} = X_{current}$
Routing Direction	E	W	S	N	C

* $X_{dest}/X_{current}/Y_{dest}/Y_{current}$ are the coordinates of the destination and current routers.

As discussed in the previous section, when an error is detected in the router, the error information will be sent to the controller, and then the controller will reconfigure the routing tables of adjacent routers, i.e., updating the faulty routing direction to “Spare Router” direction. In addition, the reconfiguration information will also be sent to the spare router. According to the replacing position of the faulty router, namely NW, NE, SW or SE in the block, the routing table of a spare router is obtained as Table 2. If the spare router is routing to the direction of the faulty router, the selection signal for the switch is also generated by the spare router. In this way, packets can be transmitted between original routers and spare routers following the routing algorithm. Hence, the NoC works correctly when faulty routers are present.

Table 2. Routing Table in Spare Router of Different Replacing Directions*

Spare Router Replacing Direction	$Y_{dest} > Y_{current}$	$Y_{dest} < Y_{current}$	$Y_{dest} = Y_{current}$ $X_{dest} > X_{current}$	$Y_{dest} = Y_{current}$ $X_{dest} < X_{current}$
NE	NE	NW	SE	NE
SW	SE	SW	SW	NW
SE	SE	SW	SE	NE
NW	NE	NW	SW	NW

* NE/NW/SE/SW are the port names of the spare router.

IV. Evaluation and Experimental Results

The proposed design is evaluated through mathematical analysis and simulation in terms of reliability, mean time to failure, system throughput, latency and area.

4.1 Effective Reliability Analysis

A system is considered to be reliable if all PEs function properly and exchange data correctly. In a traditional $N \times N$ mesh, if one router fails to work, the data being transmitted may get lost, thus deteriorating the performance. An NoC with spare routers, however, has the advantage of replacing the faulty router with a spare one to maintain the functionality of the network, therefore achieving a higher reliability. With more faulty routers in the system, NoC with spares can either replace all the faulty routers and continues to work, or cannot repair the network and fails to work. Whether it will work or not depends on the pattern of faulty routers.

The reliability of an NoC system is defined as the probability of the system to function correctly, provided that each router works independently and fails with an error probability. Given the mesh size and the error rate of each router, a large number of patterns are randomly generated. The reliability of the NoC is then measured by the statistical outcome of the network working correctly. Another router-level redundant NoC design is presented in [20]. This design uses a spare router in each column, so it is referred to as the column-spare NoC and compared to the proposed quad-spare NoC. Fig. 6 shows the reliability of both quad-spare and column-spare NoCs in different sizes with different router error rates. It can be seen that the reliability of the quad-spare NoC is higher than the column-spare NoC.

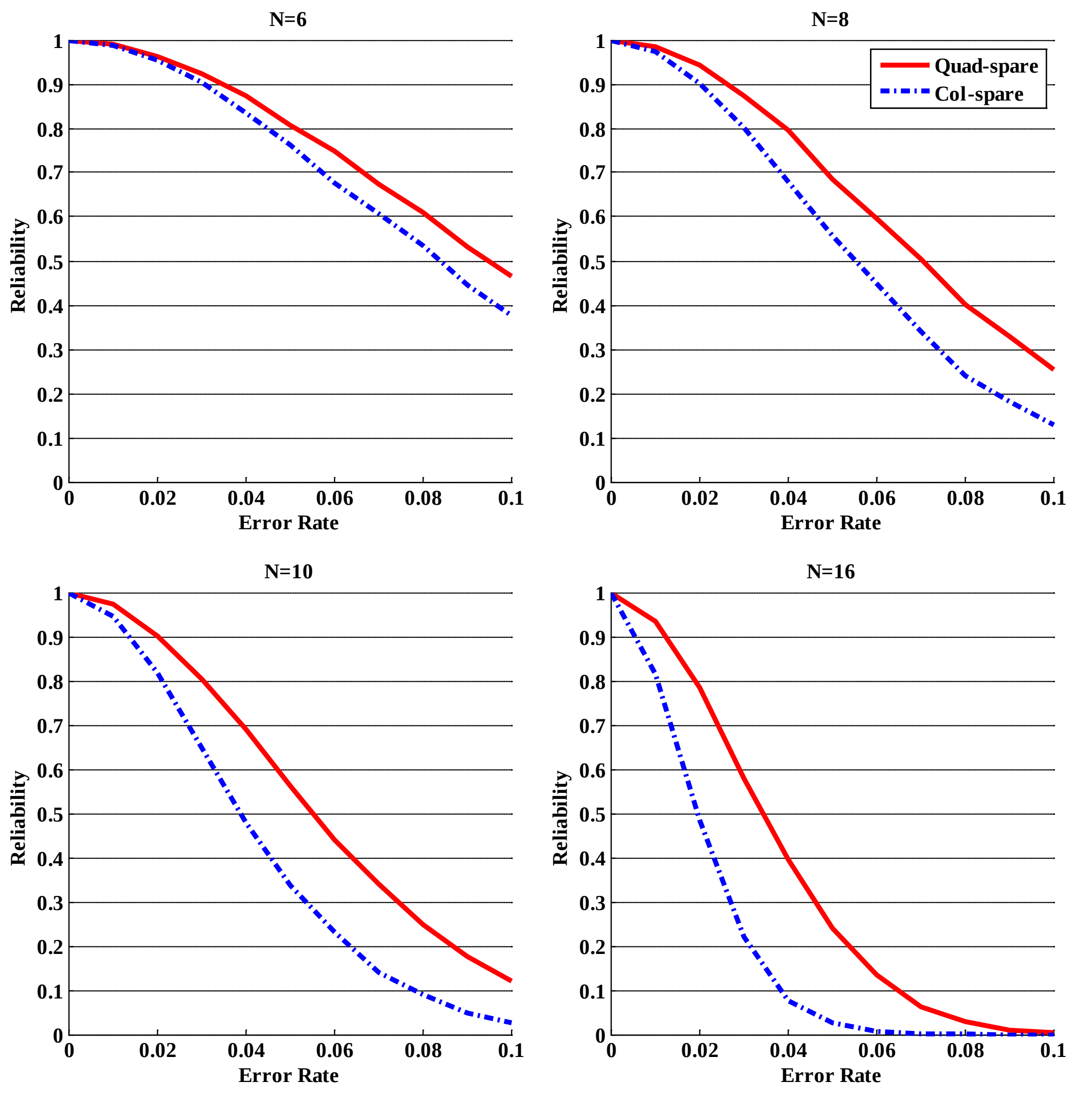


Fig. 6. Reliability of various spare NoCs with different router error rates (the proposed design vs. the design in [20])

For systems of the same size, the proposed design uses more spare routers than that of [20]. A higher reliability is thus expected, as shown in Fig. 6. In this case, reliability may not an appropriate metric. For a fair comparison, therefore, a new metric referred to as effective reliability, is proposed and utilized for evaluation. The effective reliability takes into account the amount of redundancy introduced into a design and evaluates the effectiveness of utilizing redundancy to improve reliability. It is defined as:

$$Eff\_Reliability = Reliabilit\ y \times \frac{Number\ of\ \ Routers\ without\ Redundancy}{Number\ of\ \ Routers\ with\ Redundancy} \qquad (1)$$

The effective reliability of various NoCs with different numbers of error rates is shown in Fig. 7. As can be seen, there is a threshold point, at which the quad-spare mesh technique starts to outperform the column-spare technique. This threshold is dependent on the router error probability and the size of the mesh. The quad-spare mesh technique works better for a larger error rate and/or in a larger network. Hence, it is more effective and provides a better fault-tolerance when more routers are likely to fail in an NoC.

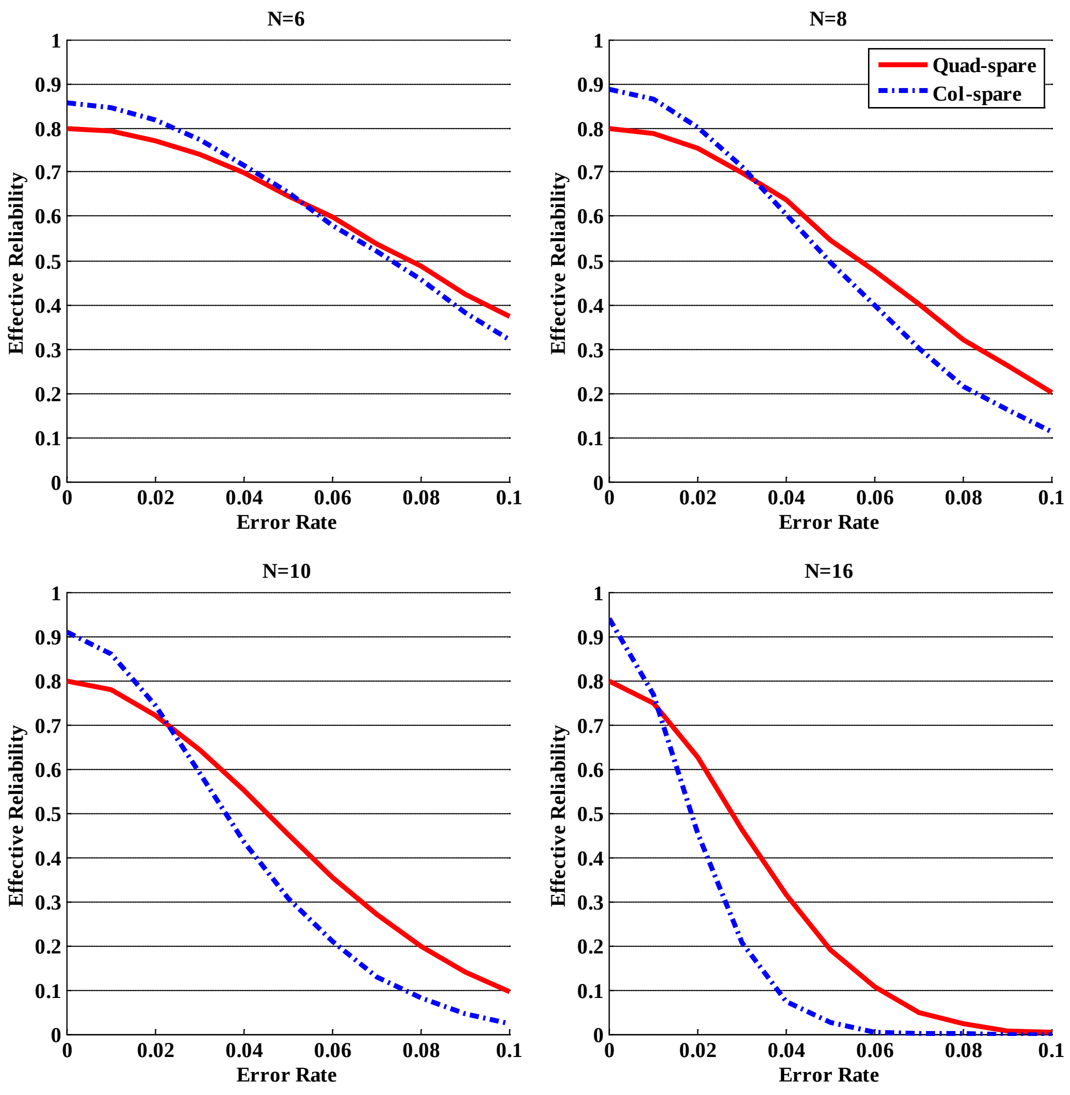


Fig. 7. Effective reliability of various spare NoCs with different router error rates (the proposed design vs. the design in [20])

4.2 *Time Degradation Reliability Analysis*

The performance of hardware may degrade with time. Next, time degradation is taken into consideration. The reliability of an NoC router $R_r(t)$ is defined as the probability that a router works correctly from time 0 to *t*. It is determined by the error rate, $\lambda(t)$, which is a measure in the number of errors per time unit. After the router has been initialized, $R_r(t)$ can be expressed using the exponential failure law [23]:

$$R_r(t) = e^{-\lambda t} \tag{2}$$

The analytical models for the traditional $N \times N$ mesh, $(N+1) \times N$ column-spare mesh and $(N \times N + N \times N/4)$ quad-spare mesh are derived as follows.

In the $N \times N$ mesh, all the routers must work correctly to ensure the system's reliability. As a result, the reliability of the system is the product of all the routers' reliability, shown as follows:

$$R_{sys\_mesh}(t) = (R_r)^{N \times N} \tag{3}$$

The column-spare mesh is spilt into *N* columns. Therefore every column must function correctly to ensure the system's functionality. The reliability of such a system is then the product of each column's reliability R_c. Each column has *N+1* routers. A column works correctly if either all the routers within the column are operational, which gives a reliability of R_r^{N+1}, or *N* of the *N+1* routers are operational. In the latter case, one of the *N+1* routers is faulty, resulting in a reliability of $C_N^{N+1}(1-R_r)(R_r)^N$ .The reliability of the column-spare mesh is given by:

$$R_{sys\_col\_spare}(t) = (R_c)^N = \left[R_r^{N+1} + C_N^{N+1}(1-R_r)(R_r)^N\right]^N \tag{4}$$

As to the quad-spare mesh, a system has $(N / 2)^2$ blocks and every block has to work correctly to ensure the system's reliability. The reliability of a system is then the product of each block's reliability R_{block}. A block works correctly if at least four of the five routers in the block are operational. This leads to the reliability of the quad-spare mesh given by:

$$R_{sys\_quad\_spare}(t) = (R_{block})^{(N/2)^2} = \left[R_r^5 + C_4^5(1-R_r)(R_r)^4\right]^{(N/2)^2} \tag{5}$$

In the reliability analysis using equations (3) (4) and (5), the error rate of a router is assumed to be $\lambda = 0.00315$ (times/year) [24]. The reliability of various meshes with different sizes over 1 to 10 years is shown in Fig. 8. As can be seen, the spare mesh architectures outperform the traditional one and the quad-spare mesh has a higher reliability than the column-spare mesh. As the mesh size increases, the reliability decreases. However, the traditional mesh drops more quickly than the other two designs, while the quad-spare mesh has the most slowly decreasing reliability – it still maintains a reliability of nearly 0.8 in the 10th year.

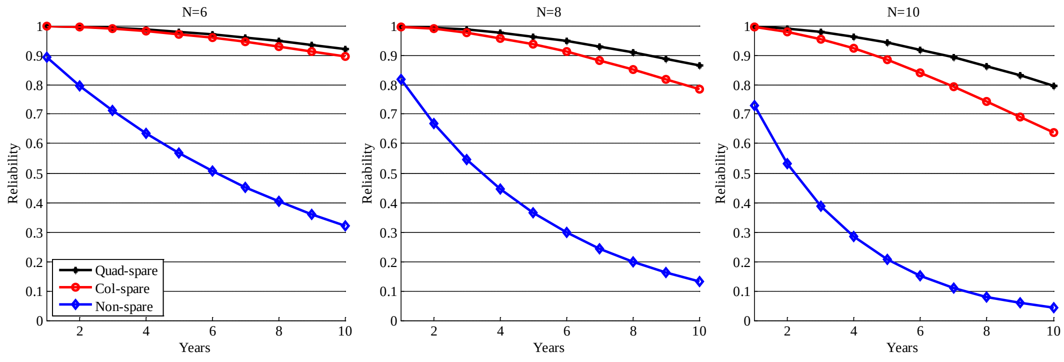


Fig. 8. Reliability in time of the traditional non-spare NoC, the column-spare NoC and the proposed

quad-spare NoC for different sizes.

For a further comparison, the reliability gain is defined as the ratio of the reliability between a spare NoC and a non-spare NoC. The result is shown in Fig. 9. It can be seen that the reliability gain increases with the mesh size and time, and that the quad-spare NoC has a higher reliability gain than the column-spare NoC.

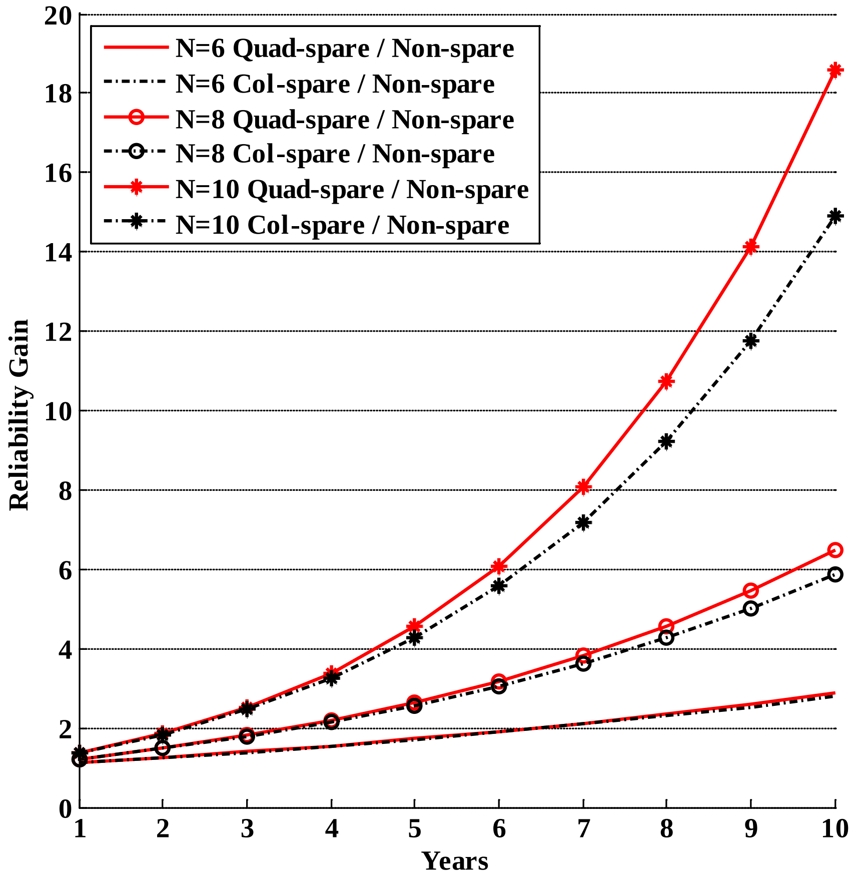


Fig. 9. Reliability gain in time of the traditional non-spare NoC, the column-spare NoC and the proposed quad-spare NoC for systems of different sizes.

4.3 Mean Time to Failure Analysis

The mean time to failure (MTTF) is the average time before a system fails, which is usually given as a function of the reliability in time, i.e.,

$$MTTF = \int_0^\infty R(t)dt \tag{6}$$

Take equation (3) into (6), the MTTF for an $N \times N$ mesh can be expressed as:

$$MTTF_{mesh} = \int_0^\infty e^{-N\times N\lambda t} dt = \frac{1}{N^2\lambda} \tag{7}$$

Similarly, if equations (4) and (5) are substituted into (6), the MTTF of the column-spare and quad-spare mesh are respectively given by:

$$MTTF_{col\_spare} = \int_0^\infty \left[R_r^{\,N+1} + C_N^{N+1}(1-R_r)(R_r)^N\right]^N dt \tag{8}$$

$$MTTF_{quad\_spare} = \int_0^\infty \left[R_r^{\,5} + C_5^{4}(1-R_r)(R_r)^4\right]^{(N/2)^2} dt \tag{9}$$

The MTTF for different mesh sizes is shown in Fig. 10. The gain of MTTF is also depicted to highlight the effect of spare NoC with the NoC size. For a 10×10 NoC, the MTTF of the column-spare mesh is 1.236, while the proposed scheme is 1.731, which is 40.0% higher than the column-spare mesh.

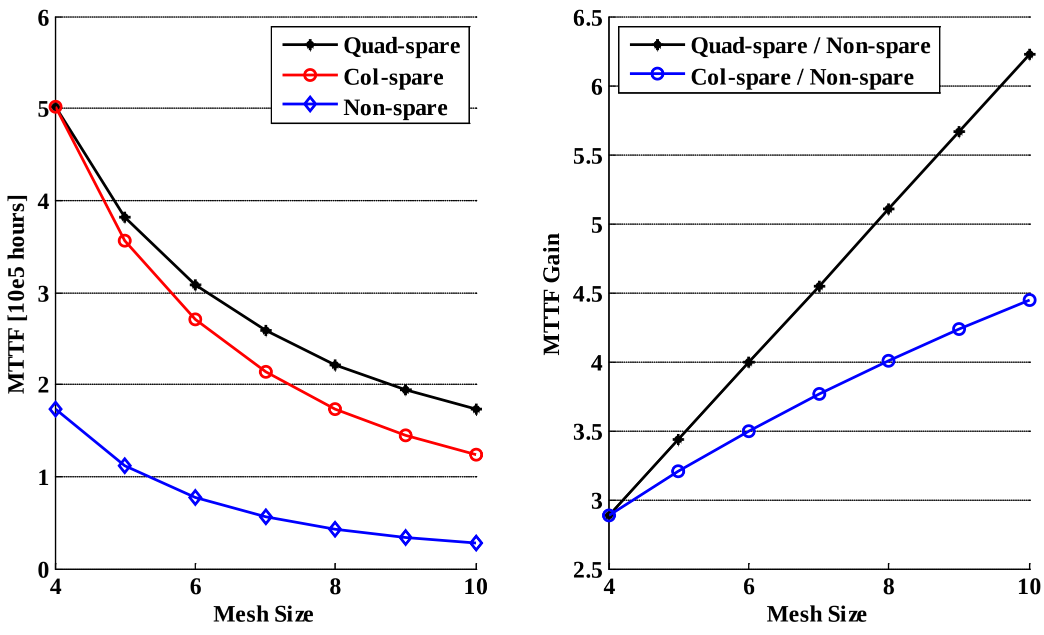


Fig. 10. MTTF and MTTF gain the traditional non-spare NoC, the column-spare NoC and the proposed quad-spare NoC for systems of different sizes.

4.4 Throughput and Latency

A flit-level simulator in C++ is used for measuring the throughput and latency. Wormhole switching is used as the switching technique of the router. Each input port has three virtual channels and each channel has a FIFO buffer to store four flits. Each PE can inject packets independently. The packet length is set to 16 flits, including one header flit. The destination of a packet is randomly determined, resulting in a uniform traffic pattern. The XY routing is used for this network, which routes packets along the X-axis first, and then Y-axis. The performance measures include system throughput and latency.

In the simulation of throughput, 100 different fault patterns are randomly generated. In this evaluation, all PEs are assumed to be healthy, but only one randomly chosen PE is injecting packets. The throughput is the average number of flits received by the other PEs per clock cycle. Figs.11 and 12 show the throughput of 4×4 and 6×6 quad-spare NoCs, averaged over 100 different fault patterns. As revealed in the figures, an error-free network has the highest throughput.

The throughput decreases as the number of faulty routers increases. However, in an $N \times N$ network with only a few faulty routers, our proposed design has the advantage of keeping the throughput almost the same as an error-free network. This is due to the fact that the spare routers can replace some faulty routers, thus maintaining the functionality of the network and the throughput.

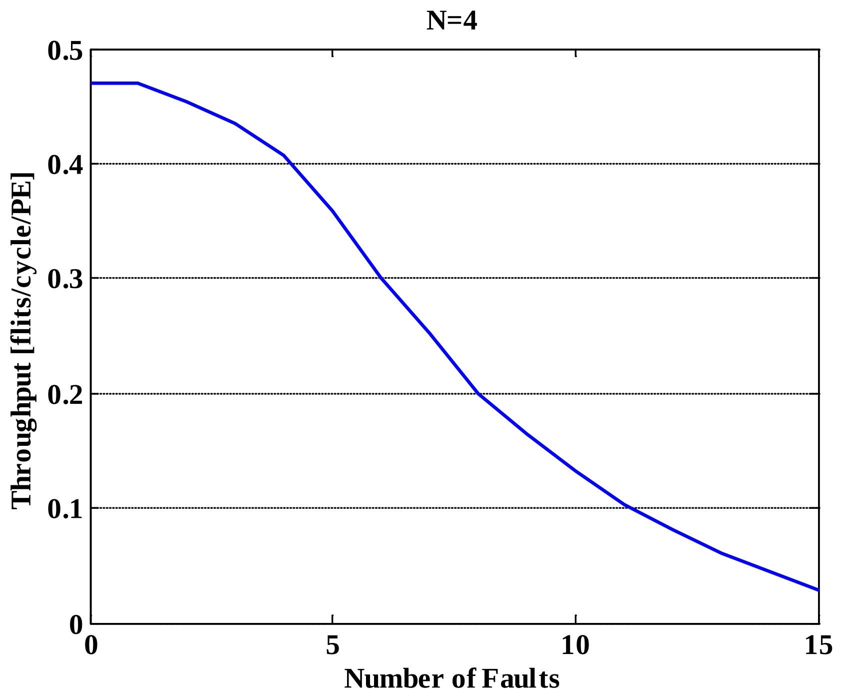


Fig. 11. Average throughput for a network of 16 PEs

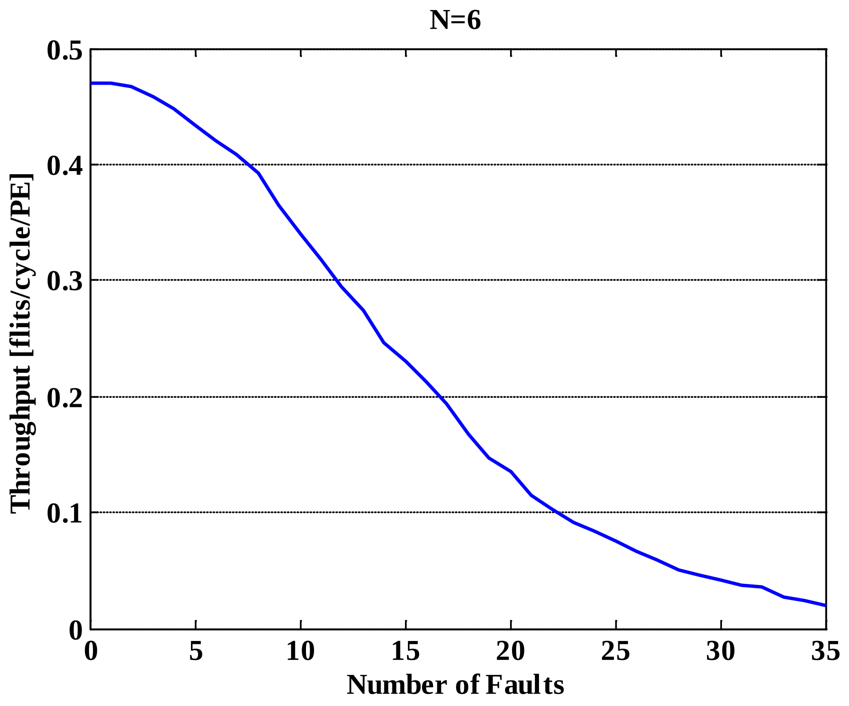


Fig. 12. Average throughput for a network of 36 PEs

A real-time H.264 video-decoding application [25] is implemented using a 4×4 NoC. An H.264 decoder is mainly composed of IDCT-IQ (Inverse Discrete Cosine Transform – Inverse

Quantization), MC (Motion Compensation) and Deblocking blocks. In order to simulate the effectiveness of the proposed architecture on the H.264 decoder, the distribution of data transmission is calculated, as shown in Fig. 13. Each block is mapped onto one PE and another PE serves as a controller. The throughput is shown in Fig. 14.

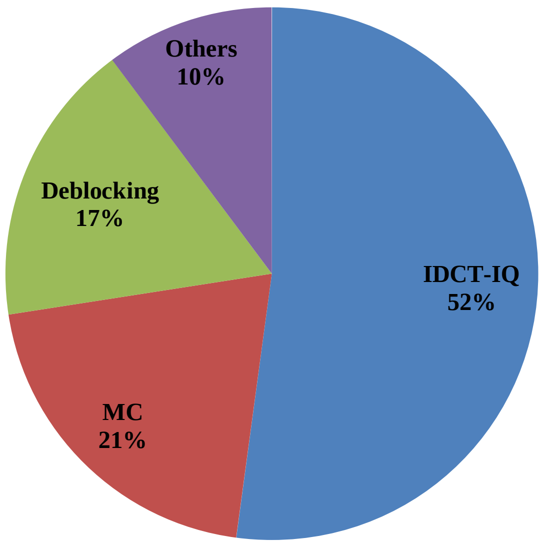


Fig. 13. Distribution of data transmission of each block in an H.264 decoder

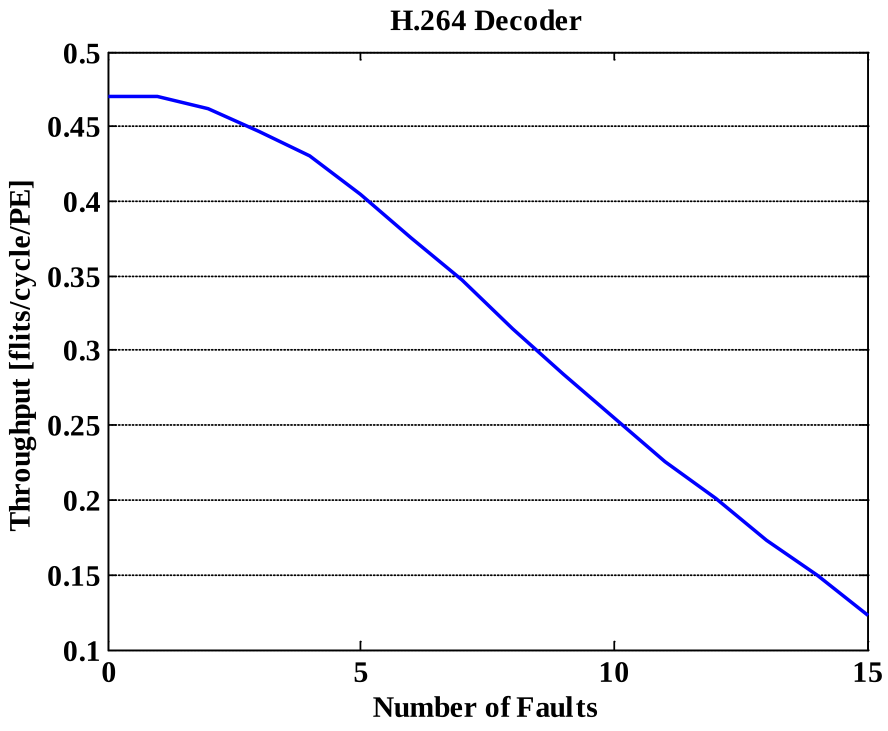


Fig. 14. Throughput of the H.264 decoder implemented using a 4×4 NoC

4.5 Latency

Latency is the number of cycles that elapses from the generation of the first flit of a packet at a node to the time when the packet is received by the destination PE. The latency for a packet to traverse a router is three cycles. In addition, it takes one clock cycle for a flit to cross a link to get to the input port of a neighboring router. A packet may stay longer in the input buffer if there is a contention at the crossbar or the destined output is currently blocked. Virtual cut-through switching is assumed.

We have simulated the NoCs with various sizes and various numbers of faulty routers. Due to limited space here, only the latency for the 4×4 quad-spare NoC is shown in Fig. 15, which demonstrates the typical results for most simulation cases. In the simulation, all the PEs inject packets into the network. Fig. 15 shows the simulation results for a 4×4 mesh with zero and single faulty router. The dots denote the simulation data, while a solid line is a fitting curve of the discrete data. For low traffic densities, the latency is kept around 50 cycles. However, as the density increases, network saturation occurs, resulting in a *latency wall*. When the number of faulty routers increases, the maximum throughput decreases and the average latency increases. The error-free NoC has a maximum throughput of 0.289 (flits/cycle/node) and an average latency of 49.31 (cycles), while single-error NoC has a maximum throughput of 0.274 (flits/cycle/node) and an average latency of 50.52 (cycles). Compared with the error-free NoC, the throughput only decreases by 5.19% and the average latency increases by 2.40%.

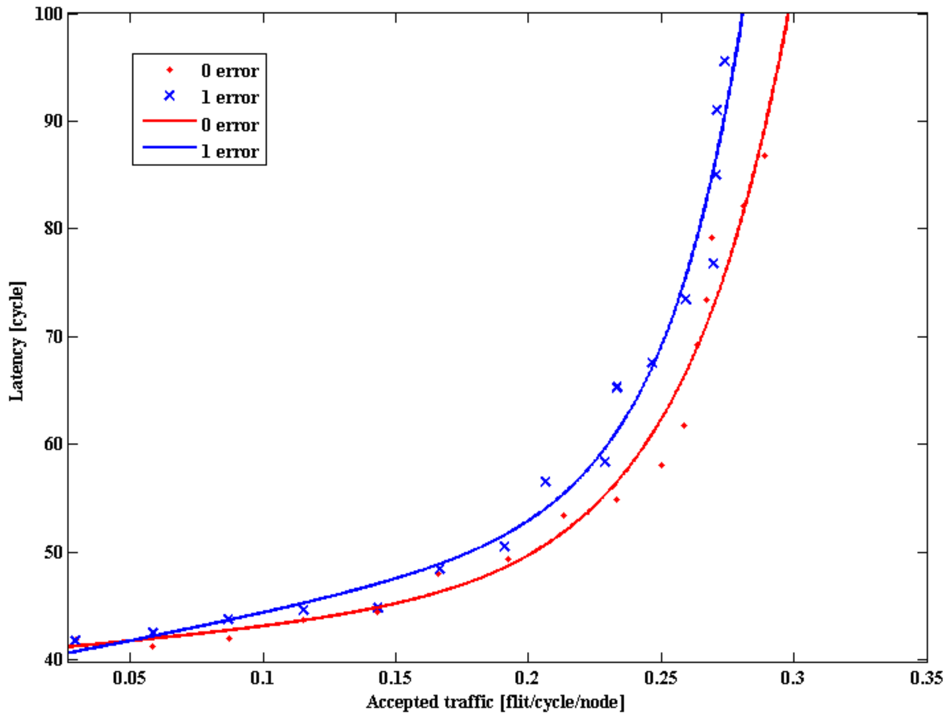


Fig. 15. Latency for 4×4 mesh with zero and single faulty routers. The dots denote the simulation data. A solid line is the fitting curve of the discrete data.

4.6 Hardware Implementation Result

In a quad-spare NoC, one more port is added to each router in the mesh to provide connections to the spare router. Hence, each original router in the mesh has 6 ports, and the spare router has 5 ports. Take a 2×2 block for example. In a traditional non-redundant NoC, the number of ports is 5×4=20. As to the quad-spare NoC, this number is 6×4+5=29, resulting in a 45% increase. As the required hardware resource is approximately proportional to the number of ports, the hardware overhead is estimated to be approximately 50% by taking the additional MUXs and wiring into consideration.

The proposed quad-spare NoC architecture is implemented with TSMC 65nm 1P9M technology (CLN65GPLUS). The non-redundant NoC is also implemented for calculating the area overhead. Fig. 16 shows the layout of a 2×2 block in the non-redundant and quad-spare NoCs. The sizes of the non-redundant and quad-spare NoCs are 221μm×221μm (0.049mm^2) and 266μm×266μm (0.071mm^2) respectively. The results show that the area of the proposed architecture increases by 45.9% compared with the traditional NoC, which is consistent with the above estimated hardware overhead.

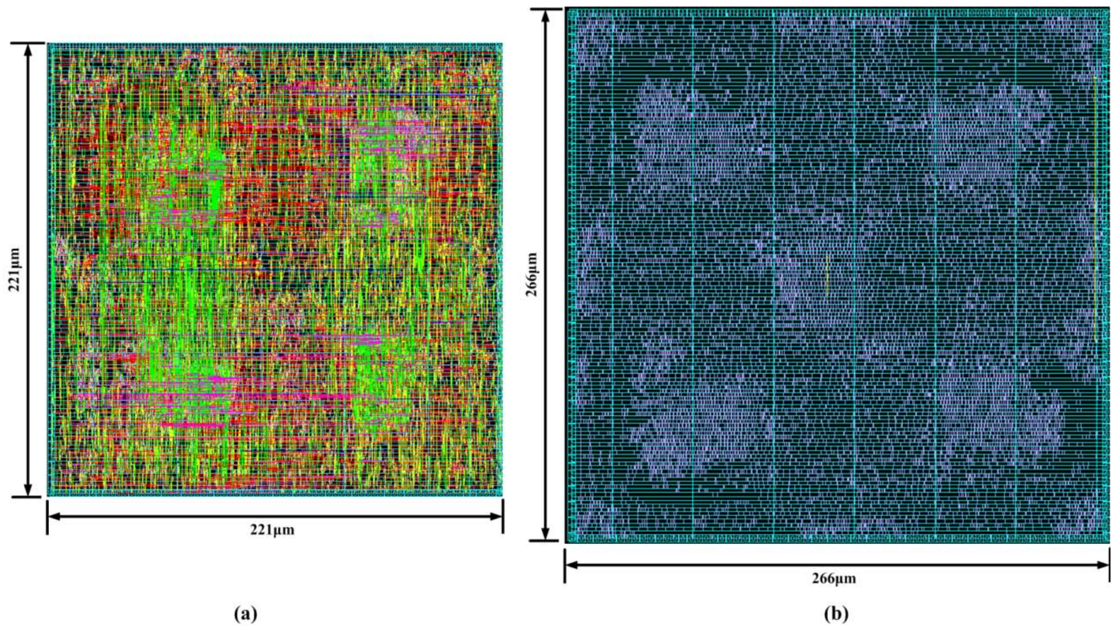


Fig. 16. Implementation of one block in (a) a non-redundant NoC and (b) a quad-spare NoC

V. Acknowledgement

This work is supported in part by the China National High Technologies Research Program (No. 2012AA012701), the Tsinghua Information S&T National Lab Creative Team Project, the International S&T Cooperation Project of China grant (No. 2012DFA11170), the Tsinghua Indigenous Research Project (No.20111080997), the Special Scientific Research Funds for Commonweal Section (No.200903010), the Science and Technology Project of Jiangxi Province (No.20112BBF60050), the NNSF of China grant (No.61274131) and the University of Alberta China Opportunity Fund (Joint Research Lab project).

VI. Conclusion

In this paper, a router-level redundant architecture, referred to as the quad-spare mesh, is proposed for fault tolerant NoC designs. One spare router is located in the center of a 2×2 block, providing an alternative to four other routers. This prevents the problem that faulty router breaks the communication between healthy PEs, thus maintaining the correct function of an NoC with faulty routers. The proposed design significantly improves a system's reliability and its mean time to failure. The throughput only slightly decreases when all faulty routers can be replaced by the spare ones. As the number of faults increases, the performance degrades gracefully. Topology reconfiguration and routing algorithm can be performed dynamically. The NoC after reconfiguration is consistent to the original one, which implies that our design is transparent to the upper layers including operating systems and user applications.

The reliability of an NoC is improved by using spare routers, which are regularly connected to as many original routers as possible. Therefore, the hardware overhead is kept low with a high throughput and a low latency. This idea on the use of redundancy is not restricted in the 2D-mesh topology and it could also be used in NoCs with other types of topologies. It can further be extended to tolerate errors due to faulty links. Hence, the proposed fault tolerant architecture is scalable and is potentially useful in future NoC designs.

References

[1] Krewell, "Multicore Showdown," Microprocessor Report, vol. 19, pp. 41-45, 2005.

[2] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," IEEE Computer, vol. 35, pp. 70-78, 2002.

[3] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip", ACM Computing Surveys, vol. 38, March 2006, Article 1.

[4] W. J. Dally and B. Towles, "Route Packets, not wires: On-chip interconnection networks," in Proc. ACM/IEEE Design Automation Conf., Jun. 2001, pp. 18-22.

[5] M. Pirretti, G. M. Link, et al, "Fault Tolerant Algorithms for Network-On-Chip Interconnect," in Proc. ISVLSI, 2004, pp. 46-51

[6] Z. Zhang, A. Greiner, S. Taktak, "A Reconfigurable Routing Algorithm for a Fault-tolerant 2D-Mesh Network-on-Chip," DAC 2008, pp.441-446

[7] H. Kariniemi, "Fault-Tolerant XGFT Network-On-Chip for Multi-Processor System-on-Chip Circuirs", Proceedings of the International Conference on Field Programmable Logic and Applications, Finland, 2006, pp. 186-190

[8] P. P. Pande, C. Grecu, A. Ivanov, and R. Saleh, "Design of a switch for network-on-chip applications," IEEE International Symposium on Circuits and Systems (ISCAS) 5: 217–220, 2003.

[9] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," in IEEE Micro, 23(4): 14-19, 2003.

[10] D. Fick, et al., "A highly resilient routing algorithm for fault-tolerant NoCs," in Proc. DATE'09, pp. 21-26, Mar. 2009.

[11] M. B. Stensgaard, J. Sparso, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology," Second ACM/IEEE International Symposium on Networks-on-Chip , pp.55-64, April 2008

[12] D. Fick, et al., "Vicis: A reliable network for unreliable silicon," in Proc. DAC'09, pp. 812-817, Jul. 2009

[13] A. Jantsch and H. Tenhunen, Networks on Chip. Kluwer Academic Publishers, Chapter 6, 2003.

[14] V. Praveen, B. Nilanjan, and S. C. Karam, "Quality-of Service and Error Control Techniques for Network-on-Chip Architectures," in Proc. Of the Great Lakes symposium on VLSI, 2004.

[15] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in Proc. Asia and South Pacific Design Automation Conference, 2003

[16] P. Shivakumar, S. W. Keckler, C. R. Moore, and D. Burger, “Exploiting Microarchitectural Redundancy for Defect Tolerance,” in Proc. IEEE Int. Computer Design Conf., Oct. 2003, pp. 481-488

[17] E. Schuchman and T. N. Vijaykumar, “Rescue: A microarchitecture for testability and defect tolerance,” in Proc. IEEE/ACM Int. Symp. Computer Architecture, Jun. 2005, pp. 160-171.

[18] Jie Han, Erin R. Boykin, Hao Chen, Jinghang Liang, and José AB Fortes. "On the reliability of computational structures using majority logic." In IEEE Transactions on Nanotechnology, 10, No. 5 (2011): 1099-1112.

[19] L. Zhang, Y. Han, Q. Xu, X. Li, H. Li, “On Topology Reconfiguration for Defect-Tolerant NoC-Based Homogeneous Manycore Systems,” in IEEE Trans. on VLSI Systems, 17(9): 1173-1168, 2009.

[20] Y. C. Chang, C. T. Chiu, S. Y. Lin, C. K. Liu, “On the Design and Analysis of Fault Tolerant NoC Architecture Using Spare Routers”, ASP-DAC 2011, pp. 431-436.

[21] C. Grecu, P. Pande, A. Ivanov, R. Saleh, “BIST for Netwok-on-Chip Interconnect Infrastructures,” in Proc. 24th IEEE VLSI Test Symposium, 2006, pp. 30-35.

[22] A. M. Amory, E. Briao, E. Cota, M. Lubaszewski, F. G. Moraes, “A scalable test strategy for network-on-chip routers, ” in Proc. IEEE International Test Conference, Nov. 2005, pp. 590-599.

[23] L.T Wang, C.E Stroud, and N.A Touba, System-on-Chip Test Architectures: Nanometer Design for Testability, Morgan Kauffmann, 2008

[24] W.J Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2003

[25] Y. Kim and S. Sair, “Designing Real-Time H.264 Decoders with Dataflow Architectures,” in the third IEEE/ACM/IFIP International Conf. on Hardware/Software Codesign and System Synthesis, 2005, pp. 291-296

Figure Captions

Fig. 1. Healthy PEs are isolated by faulty routers and broken links.

Fig. 2. (a) Quad-spare mesh topology

Fig. 2. (b) Communication path between routers

Fig. 3. PE connections in a block

Fig. 4. (a) Input channels of PEs

Fig. 4. (b) Output channels of PEs

Fig. 5. An NoC reconfiguration example. For clarity, PEs, faulty routers and unused interconnections are not shown in this figure.

Fig. 6. Reliability of various spare NoCs with different router error rates (the proposed design vs. the design in [20])

Fig. 7. Effective reliability of various spare NoCs with different error rates (the proposed design vs. the design in [20])

Fig. 8. Reliability in time of the traditional non-spare NoC, the column-spare NoC and the proposed quad-spare NoC for systems of different sizes.

Fig. 9. Reliability gain in time of the traditional non-spare NoC, the column-spare NoC and the

proposed quad-spare NoC for systems of different sizes
Fig. 10. MTTF and MTTF gain the traditional non-spare NoC, the column-spare NoC and the proposed quad-spare NoC for systems of different sizes
Fig. 11. Average throughput for a network of 16 PEs
Fig. 12. Average throughput for a network of 36 PEs
Fig. 13. Distribution of data transmission of each block in an H.264 decoder
Fig. 14. Throughput of the H.264 decoder implemented using a 4×4 NoC
Fig. 15. Latency for 4×4 mesh with zero and single faulty routers. The dots denote the simulation data. A solid line is the fitting curve of the discrete data.
Fig. 16. Implementation of one block in (a) non-redundant NoC and (b) quad-spare NoC