Exploring Fault-Tolerant Network-on-Chip Architectures*

Dongkook Park, Chrysostomos Nicopoulos, Jongman Kim, N. Vijaykrishnan, Chita R. Das
Department of Computer Science and Engineering, Pennsylvania State University, USA, 16802
{dpark, nicopoul, jmkim, vijay, das}@cse.psu.edu

Abstract

The advent of deep sub-micron technology has exacerbated reliability issues in on-chip interconnects. In particular, single event upsets, such as soft errors, and hard faults are rapidly becoming a force to be reckoned with. This spiraling trend highlights the importance of detailed analysis of these reliability hazards and the incorporation of comprehensive protection measures into all Network-on-Chip (NoC) designs. In this paper, we examine the impact of transient failures on the reliability of on-chip interconnects and develop comprehensive counter-measures to either prevent or recover from them. In this regard, we propose several novel schemes to remedy various kinds of soft error symptoms, while keeping area and power overhead at a minimum. Our proposed solutions are architected to fully exploit the available infrastructures in an NoC and enable versatile reuse of valuable resources. The effectiveness of the proposed techniques has been validated using a cycle-accurate simulator.

1. Introduction

Packet-based interconnection networks, known as Network-on-Chip (NoC) architectures, are increasingly adopted in System-on-Chip (SoC) designs, which support numerous homogeneous and heterogeneous functional modules. Reduced feature sizes into the nanoscale regime, along with increasing transistor densities, have transformed the on-chip interconnect into a deciding factor in meeting the performance and power consumption budgets of the design. A variety of interconnection schemes are currently in use, including crossbars, rings, buses, and NoC's [2]. Of these, the latter two have been dominant in the research community [3, 4]. However, buses suffer from poor scalability; as the number of processing elements increases, performance degrades dramatically. Hence, they are not considered appropriate for systems of more than about 10 nodes [3, 5]. To overcome this limitation, attention has shifted toward NoCs. On-chip networks are scalable, much like traditional macro networks, and are seen as the prime candidate to form the network infrastructure of future SoCs. NoCs, however, pose several design challenges emanating from their inherently stringent resource constraints; namely, area and power limitations. These limitations dictate the choice of routing algorithms and protocols, as well as the architectural implementation.

Aggressive technology scaling has accentuated the issue of reliability due to rapid increase in the prominence of permanent faults; these are mostly caused from accelerated aging effects such as electromigration, and manufacturing and testing challenges. Furthermore, soft upsets caused by cross-talks, coupling noise and transient faults are also a concern to overall reliability. The growing concern about reliability has prompted extensive research in this area. Many researchers [6-13] have proposed solutions for various individual aspects of on-chip reliability, such as soft faults and handling of hard failures within a network. Nevertheless, a comprehensive approach encompassing all issues pertaining to NoC reliability has yet to evolve. In this paper, we propose a comprehensive set of techniques to protect against the most common sources of failures in on-chip interconnects (including link errors, and single-event upsets within the router). The proposed mechanisms incur minimal overhead, while providing fool-proof protection. Moreover, our schemes cleverly employ resource sharing techniques to minimize the overhead imposed by the additional hardware.

To ensure protection from link errors due to crosstalk and capacitive loading, we present a flit-based hop-by-hop (HBH) retransmission scheme, and the corresponding retransmission architecture. With a minimal latency overhead of three clock cycles in the event of an error, this scheme successfully addresses the problems afflicting one of the most vulnerable components of an on-chip network, the inter-router link. In addition to providing link protection, the same architectural framework is also employed in a newly proposed deadlock-recovery scheme. While prior work in deadlock recovery has assumed additional dedicated resources, our technique uses existing retransmission buffers instead. This helps in maximizing resource utilization without incurring additional overhead.

While combinational logic circuits have traditionally been considered less prone to soft errors than memory elements, rapidly diminishing feature sizes and increasing clock frequencies are exacerbating their prominence. In fact, recent studies predict that the soft error rate (SER) per chip of logic circuits will become comparable to the SER per chip of unprotected memory elements by 2011 [14]. This observation would have a profound impact on the reliability of on-chip routers. The lack of protection from logic errors implies that soft errors afflicting a router's logic would escape the error detecting/correcting measures because they do not actually corrupt the data, but, instead, cause erroneous behavior in the functionality of the routing process. Therefore, it is imperative to provide robust protection against such upsets. A recent study [1] has addressed the issues of single-event upsets in the

* This research was supported in part by NSF grants CCR-0208734, EIA-0202007, CCF-0429631, CNS-0509251, CRI-0454123, CAREER 0093085, SRC grant 00541, and a grant from DARPA/MARCO GSRC.

logic of individual hardware components. However, the proposed techniques were applicable to a specific type of router architecture. In this work, we analyze the intricacies of intra-router logic errors, and provide comprehensive solutions relevant to all router architectures. We analyze the possible symptoms of logic errors in each module in the router pipeline and provide detailed recovery mechanisms for each case. In the sequel, we propose a novel Allocation Comparator (AC) unit, which provides full error protection to the virtual channels and switch allocation units at minimal cost.

All the mechanisms proposed in this paper are architected in such a way as to avoid negative impacts on the router's critical path. The mechanisms work in parallel with other vital stages in the router pipeline, without increasing the pipe depth.

This paper is organized as follows. First, a preliminary description of a generic NoC router architecture is given in Section 2. Link error handling techniques and a novel deadlock recovery scheme utilizing retransmission buffers are presented in Section 3. Then, logic soft-error handling techniques are described in Section 4, followed by the concluding remarks in Section 5.

2. Preliminaries

2.1. NoC Router Architecture

A generic virtual-channel-based wormhole router is shown in Figure 1. The router consists of six major components: a Routing Unit (RT), a Virtual Channel Allocator (VA), a Switch Allocator (SA), a crossbar, a retransmission buffer (necessary for fault-tolerance) and handshaking signals (used between neighboring routers; not shown in Figure 1).

Pipelining the router architecture can significantly improve performance by increasing throughput much like the pipeline of a microprocessor, thereby reducing the average latency. As described in [15], there are certain critical components that are best kept intact within a pipeline stage. These atomic modules [15] represent the finest granularity at which efficient pipelining can occur. The RT, VA, SA and crossbar represent the fundamental modules within an NoC router. The

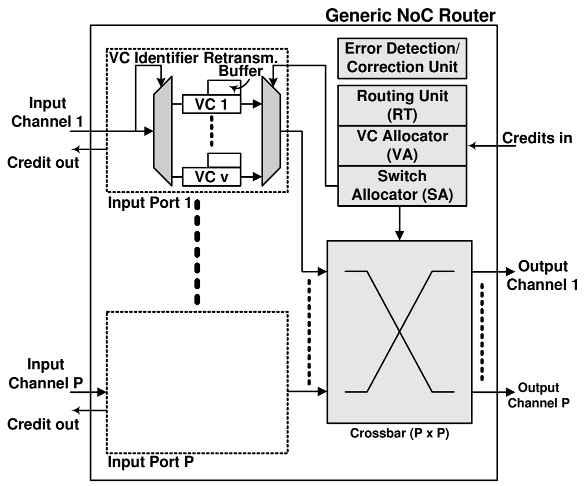


Figure 1. A generic NoC router architecture with *P* PCs and *V* VCs per PC

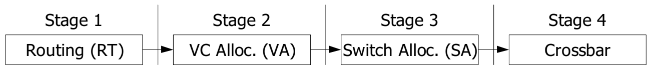


Figure 2. Atomic modules and pipeline of NoC routers

interdependencies of these modules in a pipelined architecture are illustrated in Figure 2. A simple architecture consists of a 4-stage pipeline router, one stage for each module. However, by employing clever techniques such as Speculative Switch Allocation [15] and Look-ahead Routing [16], researchers have been able to break some of these interdependencies by parallelizing operations, thus shortening the router's critical path. This has led to 3-stage and 2-stage router implementations [17]. Recently, [18] proposed a single-stage router, which fully parallelizes the router operation to minimize average latency.

2.2 Simulation Platform

A cycle-accurate network simulator was developed to conduct detailed evaluation of the proposed schemes. The simulator operates at the granularity of individual architectural components, accurately emulating their functionalities. The simulation test-bench models the pipelined routers and their interconnection links. All simulations were performed in a 64-node (8x8) MESH network with 3-stage pipelined routers. Each router has 5 physical channels (PCs) including the PE-to-router channel, and each PC has a set of 3 associated virtual channels (VCs). One message (or packet) consists of four flits. The simulator keeps injecting messages into the network until 300,000 messages (including 100,000 warm-up messages) are ejected. A uniform message injection traffic pattern was used, where a node injects messages into the network at regular intervals specified by the injection rate. For a destination node selection, three distributions are used: normal random (NR), bit-complement (BC), and tornado (TN) [19]. Single link traversal is assumed to complete within one clock cycle, thus eliminating the need for pipelined links (which would incur further power and area penalties).

To evaluate fault-tolerance in the network, various soft faults were randomly generated both within the routers and on the inter-router links. The simulator was also used to calculate the area and power overhead of the proposed architectures. A generic 5-port router architecture along with all proposed modifications were implemented in structural Register-Transfer Level (RTL) Verilog and then synthesized in Synopsys Design Compiler using a TSMC 90 nm standard cell library. The resulting design operates at a supply voltage of 1 V and a clock speed of 500 MHz. Both dynamic and leakage power estimates were extracted from the synthesized router implementation. These power numbers were then imported into the cycle-accurate network simulator and used to trace the power profile of the entire on-chip network. We measured average message latency and energy per packet as the performance and energy parameters.

3. Handling Link Soft Faults

Primarily two types of soft faults could upset the on-chip network infrastructure: link errors occurring during flit

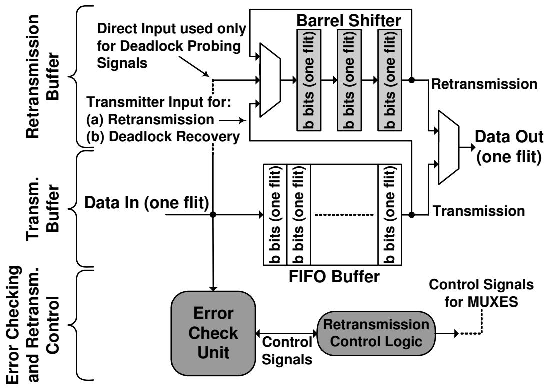


Figure 3. Proposed transmission/retransmission buffer architecture

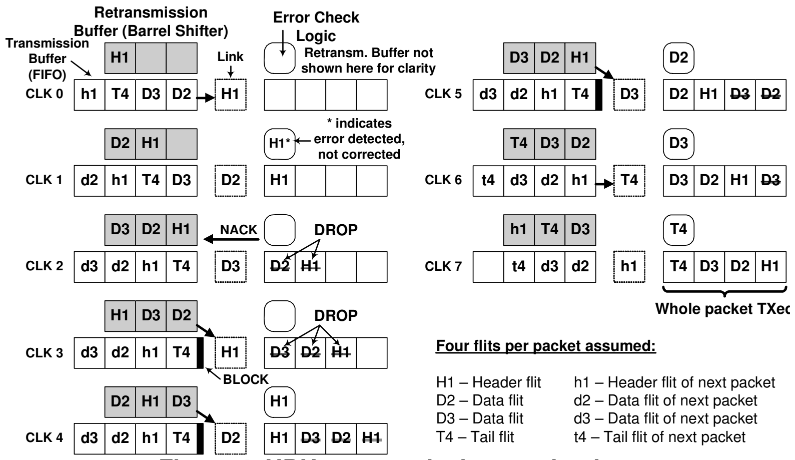

Retransmission Buffer (Barrel Shifter)
Transmission Buffer (FIFO)
Link
Error Check Logic
Retransm. Buffer not shown here for clarity
* indicates error detected, not corrected
NACK
DROP
BLOCK
CLK 0
CLK 1
CLK 2
CLK 3
CLK 4
CLK 5
CLK 6
CLK 7
Whole packet TXed
Four flits per packet assumed:
H1 – Header flit
D2 – Data flit
D3 – Data flit
T4 – Tail flit
h1 – Header flit of next packet
d2 – Data flit of next packet
d3 – Data flit of next packet
t4 – Tail flit of next packet

Figure 4. HBH retransmission mechanism

traversal from router to router, and intra-router errors occurring within individual router components. The latter will be discussed in Section 4. This section focuses on link errors, which are mostly caused by channel disturbances such as cross-talk, coupling noise and transient faults [20]. Link errors have been studied extensively by researchers, since they have so far been considered the dominant source of errors in on-chip network fabrics. They have been tackled within the context of two central themes – correction and retransmission. Some degree of error correction can be achieved through the use of Error Correcting Codes (ECC), as in [21, 22]. These codes achieve what is known as Forward Error Correction (FEC). Similarly, retransmission schemes can also be used to compensate for link errors.

Hybrid techniques [6], which provide both error correction and retransmission, allow for more robust protection of data. Hybrid solutions compensate for the limitations of error correcting codes. For example, Single Error Correction and Double Error Detection (SEC/DED) codes can correct at most one error, but can detect double-bit errors. Therefore, upon detection of a double-bit error, the SEC/DED unit may invoke a retransmission mechanism. Retransmission can occur in two different forms: End-to-End (E2E) or Hop-by-Hop (HBH). In an E2E scheme, the original data is checked only at the destination node, while in an HBH scheme, data is checked in all routers along the path from a source to a destination. Both flavors require dedicated buffers, as opposed to FEC techniques, but they can handle multiple-bit errors, since a clean copy of the data is always maintained.

Both FEC techniques and E2E retransmission schemes suffer severely from errors in the header flit. For example, if the destination node address of a packet is corrupted during the transfer, the packet might be routed to a wrong destination. Even if FEC can correct the error at the (wrong) destination node, the packet should be sent to the correct destination creating additional network traffic. Similarly, E2E schemes need to send a retransmission request from the wrong destination to the source node. Moreover, if the source node address is corrupted, E2E techniques cannot send the retransmission request to the correct source. Thus, it is very important to keep the header information from being contaminated even if the probability of header error is small. [1] addressed this problem by adopting HBH header error checking in both FEC and E2E schemes. Figure 5 shows that E2E schemes suffer from prohibitive latency penalties as error rates increase. E2E schemes also require larger retransmission buffers to account for worst case round-trip delay between a source and destination [1].

Considering all these aspects, HBH retransmission together with FEC seems to be the best choice to handle link faults. To that extend, we propose a minimal-overhead flit-based HBH retransmission scheme. The impact of the additional buffer overhead is mitigated by utilizing these same buffers for a newly-proposed deadlock recovery mechanism, discussed in Section 3.2.

3.1. A Flit-based HBH retransmission scheme

Our proposed minimal overhead HBH retransmission scheme requires a 3-flit-deep retransmission buffer per virtual channel, since a flit should be kept for 3 cycles after it leaves the current node. This 3 cycle delay corresponds to the sum of the link traversal delay (1 cycle), error checking delay at the adjacent receiving node (1 cycle), and the Negative Acknowledgement (NACK) propagation delay (1 cycle). The retransmission buffer is implemented as a barrel-shift register. This way, a flit is stored at the back of the buffer upon transmission on the link, and it moves to the front by the time a possible NACK signal arrives from the receiving node. The simplest type of transmission buffer is a First-In-First-Out (FIFO) buffer. Such an implementation has one input port and one output port, and involves simple control logic. The proposed architecture is shown in Figure 3.

Conceptually, our proposed scheme works similar to the simple retransmission schemes described in [23-25]. However, in terms of implementation, [23, 24] use a single transmission buffer that contains both sent and unsent flits together, and use pointers to track their positions. This requires that every buffer slot has an exit port so that flits can be transmitted from the middle of the buffer. This complicates the logic and incurs wiring overhead. Further, they use both acknowledgement (ACK), as well as NACK signals, whereas our proposed scheme only sends NACK signals when an error is detected. [25] uses link-level retransmission together with the Unique-

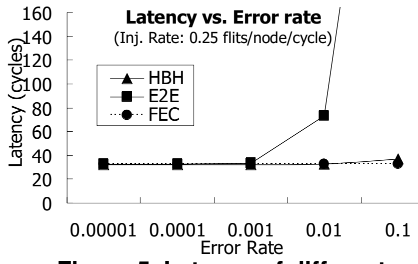


Figure 5. Latency of different error handling techniques

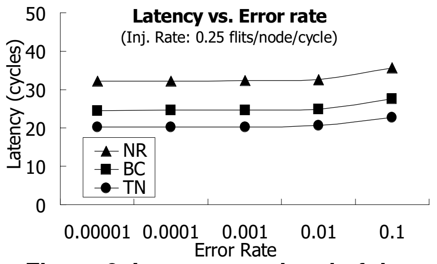


Figure 6. Latency overhead of the HBH retransmission scheme

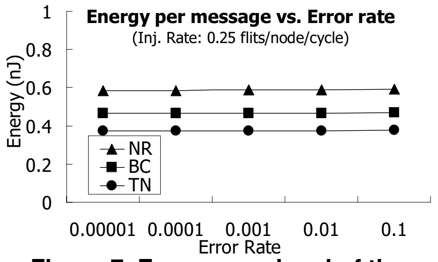


Figure 7. Energy overhead of the HBH retransmission scheme

Token Protocol (UTP) to ensure reliability. However, it requires at least two copies of a packet at all times in the network, increasing buffer occupancy and flow control complexity.

In case of a flit error in the proposed scheme, two subsequently arriving flits must be dropped until the correct flit arrives from the previous node upon retransmission. Once the correct flit is received, all previously dropped flits must then be retransmitted. This scenario is illustrated with a flit-flow example in Figure 4. The example traces the operation of the HBH retransmission mechanism when the header flit H1 is corrupted during link traversal. A clean copy of H1 is stored in the retransmission buffer when H1 is sent to the link. The error check logic detects errors in H1 in the receiving node and sends a NACK signal to the transmitting node in the next clock cycle. As seen in Figure 4, the receiving router drops the subsequent two flits (D2 and D3).

While this may seem an inefficient recovery method, it should be noted that a retransmission event will be highly unlikely under normal operation, since the architecture already employs a single-error correction scheme. While the probability of a double (or higher) error within a single flit may not be insignificant due to crosstalk, it is still low in on-chip networks. Furthermore, the corrected flit (H1 in the example of Figure 4) arrives within 3 clock cycles. This implies that only two flits need to be dropped during a retransmission event. Retransmission of these two flits incurs a latency penalty of two clock cycles. Therefore, a possible latency improvement of two clock cycles does not warrant the implementation of a more complex architecture, which would be able to handle in-situ re-arrangement of flits within each router. While such implementations are very common in macro networks, they are prohibitive in on-chip environments, because the latter have a much stricter area and power budget. The excessive area and power penalty imposed by these modifications, compounded by the increased wiring complexity, clearly overshadow the small improvement in latency during a low-probability retransmission event.

Cycle-accurate simulation of the proposed scheme with the three traffic patterns (NR, BC, TN) in an 8x8 network validates these assertions, as shown in Figure 6. The retransmission scheme is so efficient that average latency remains almost constant even up to 10 % error rate. This behavior is a direct consequence of the minimal latency incurred during a retransmission, as shown in Figure 4. Furthermore, retransmission occurs only between two adjacent hops; this restricts movement of retransmitted flits to a single inter-router link, which, in turn, has minimal impact on overall network traffic. Similarly, Figure 7 illustrates the negligible effect of the proposed scheme on the energy-per-packet metric. Since retransmission is done on a hop-by-hop basis for individual flits, the power overhead of a single-hop flit transmission is insignificant compared to the total power budget for complete packet traversal from a source to a destination.

It should be noted that the retransmission buffer also constitutes an essential component of our proposed deadlock recovery scheme, which is analyzed in detail in Section 3.2. Utilizing the same hardware for both schemes further subsidizes the area and power overhead incurred by the additional circuitry.

3.2. Deadlock Recovery

The concept of deadlock has been extensively researched in the literature. Some researchers use preventive schemes [26, 27], while others propose recovery schemes [28, 29]. However, these methods typically place constraints on resource use, preventing the system to work at full throttle. For example, an adaptive routing algorithm can use escape Virtual Channels (VCs) to recover from deadlocks, as described in [28]. The flits in these escape VCs, however, are managed by a deadlock-free deterministic routing algorithm, thereby limiting adaptivity.

Moreover, many of these techniques cannot guarantee deadlock freedom in a network with hard faults (router or link faults). They all assume a fault-free environment. This assumption, however, no longer seems reasonable in NoC environments where the probability of failure is relatively high. Several techniques have been proposed to address this issue in macro networks [23, 30], but most of them adopt complex algorithms which are not suitable in resource-constrained environments like on-chip interconnects. Thus, it is imperative to provide simple, yet effective solutions to minimize performance degradation.

3.2.1 Proposed Deadlock Recovery Scheme

To address these issues, we propose a scheme, which (1), instead of using additional dedicated resources, utilizes the existing retransmission buffers to break deadlocks, and (2) provides deadlock recovery in both fault-free and faulty environments using a very simple retransmission-buffer management policy. Hence, through efficient resource sharing, we can transform the retransmission buffers into a multifaceted reliability component in our system.

The retransmission buffers can serve a dual purpose, mainly because they are used only when packets are being transferred from one node to another. As network traffic increases, packet blocking increases and, as a result, the utilization of the HBH retransmission buffers will decrease due to decreased flit transmissions. Figure 8 and Figure 9 show the utilization of both transmission and retransmission buffers, respectively, for the adaptive (AD) and deterministic (DT) routing algorithms. In most cases, the utilization of the retransmission buffer does not follow that of the transmission buffer; instead, retransmission buffers are mostly underutilized. Furthermore, if a packet is permanently blocked due to a deadlock, the associated retransmission buffer will be empty, since there has been no data transmission for an extended period of time. Based on these observations, we propose a smart HBH-retransmission-buffer management scheme that exploits these idle buffers for deadlock recovery.

When a deadlock occurs, if any of the packets involved in the deadlock configuration can proceed by one buffer slot, all the other packets also involved in the deadlock can proceed as well. This can be achieved with the presence of a single empty buffer slot; if all packets continue to proceed in this fashion, the deadlock will eventually be broken, since some packets will ultimately move out of the deadlock configuration. In other words, instead of providing a dedicated escape channel to the destination node, as proposed in [28, 29], our scheme gradually shifts flits without breaking the cyclic dependency, until the deadlock is broken.

For example, assuming that four nodes are involved in a deadlock configuration, we have (4x3) retransmission buffers (12 in total) that are empty. Therefore, if each node temporarily moves 3 flits from the normal transmission buffers to the retransmission buffers, it will create an additional available buffer space for the preceding node in the deadlock configuration. As soon as the buffer space becomes available, the flits in the retransmission buffer can be sent to the next router. Thus, flits will be able to advance, and after several iterations, some flits will move out of the deadlock configuration, thereby breaking the deadlock situation.

Figure 10 shows an example of this scenario in detail, where a packet consists of 4 flits and the normal transmission buffer can store up to 4 flits. In step 1, a deadlock is detected and flits are moved to the retransmission buffer, as shown in step 2. The additional buffer space created by this move allows flits in the retransmission buffers to be transmitted to the next nodes. Since the retransmission buffers in our proposed architecture are barrel shifters, transmitted flits also move to the back of the retransmission buffer (flits enclosed by a thick square), as shown in steps 3 to 5. Three clock cycles later, the retransmission buffer will be empty again, as shown in step 7. At this point, the buffer state is exactly the same as in step 1, except that every flit has advanced by 3 buffer slots. This procedure will be repeated until at least one of the packets breaks the deadlock by going out to a direction away from the deadlock configuration. Once the deadlock configuration is broken, each node resumes its normal operation. In the example of Figure 10, we assume that all three nodes involved

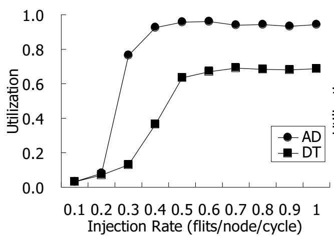


Figure 8. Transmission Buffer utilization

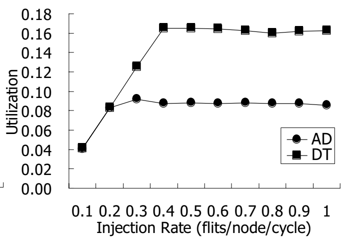


Figure 9. Retransmission Buffer utilization

in the deadlock initiate deadlock recovery action simultaneously for the sake of clarity. However, deadlock recovery need not be synchronized, as long as all nodes eventually start deadlock recovery. The proposed probing technique described below will handle this asynchronous behavior.

The proposed scheme places a lower limit on buffer size to ensure correct functionality. The technique must account for the worst-case scenario, where partially transferred messages prevent other messages from entering the transmission buffers, and thus, absorption of these partially transferred messages is necessary during the deadlock recovery process, as illustrated in Figure 11. Note that no new packets are allowed to enter the transmission buffers that are involved in the deadlock recovery.

To handle the worst-case scenario, the total buffer size (i.e. transmission and retransmission buffers) must be large enough to accommodate the remaining flits of a partial packet and still have at least one empty slot.

Theorem: The proposed scheme ensures deadlock freedom if the buffer size is larger than the lower limit specified in Equation (1).

Lower Limit : $B > M \times \sum_{i=1}^{n} N_i$, where *Eq.(1)*

$$B = \begin{cases} B_1 = T = \sum_{i=1}^{n} T_i & \text{, normal mode} \\ B_2 = T + R = \sum_{i=1}^{n} (T_i + R_i) & \text{, deadlock recovery mode} \end{cases}$$

- B	: Total buffer size (either B_1 or B_2)
- T, (T_i)	: Total size of the Transmission buffer (at node i)
- R, (R_i)	: Total size of the Retransmission buffer (at node i)
- n	: Number of nodes involved in the deadlock
- M	: Number of flits per packet (message)
- N_i	: Maximum number of different packets in a transmission buffer at node i ($= \lceil T_i / M \rceil$)

Proof: When a deadlock is detected, the transmission buffers cannot accommodate any more flits, and therefore, $B = B_1 \leq M \times \sum_{i=1}^{n} N_i$. At this point, the nodes switch to the deadlock recovery mode and if $B = B_2 > M \times \sum_{i=1}^{n} N_i$, then all the messages involved in the deadlock can be absorbed into the buffers (trans. + retrans.) with at least one empty slot still available. Since only packets involved in the deadlock can use this empty buffer slot(s), they can now proceed, and eventually, the network can recover from the deadlock. ■

Examples of the lower limit condition in deadlock recovery mode are shown below for the configurations of Figure 10 and Figure 11. Both examples show that they meet the minimum buffer requirement, and therefore, the deadlock can be broken.

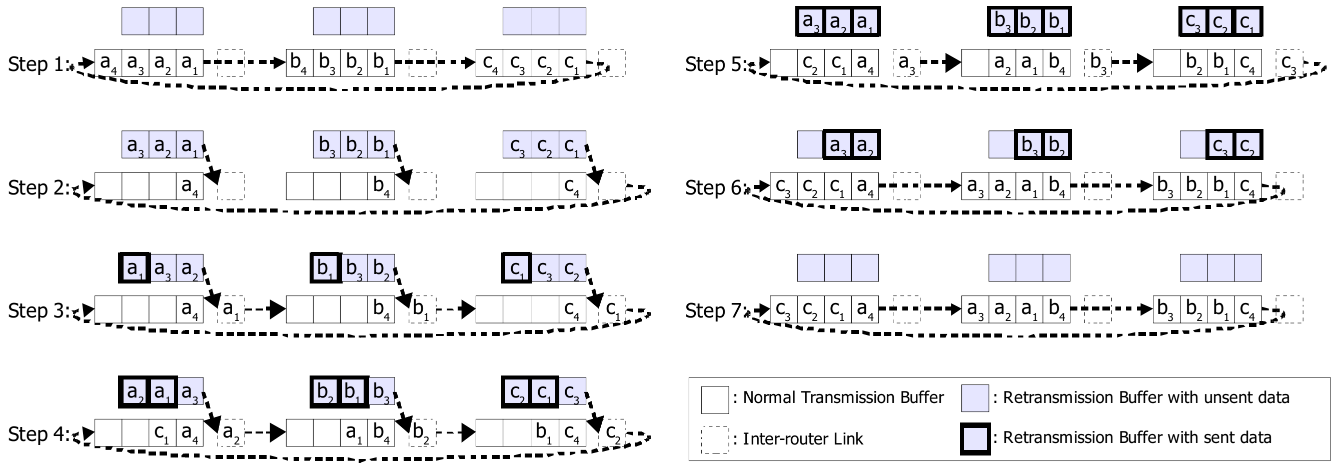


Figure 10. A Deadlock recovery example

- Figure 10: $T_i = 4,\ R_i = 3,\ M = 4,\ N_i = \lceil 4/4 \rceil = 1,\ n = 3$
$\therefore B = B_2 = n \times (4+3) = 21 > 4 \times (n \times 1) = 12$

- Figure 11: $T_i = 6,\ R_i = 3,\ M = 4,\ N_i = \lceil 6/4 \rceil = 2,\ n = 4$
$\therefore B = B_2 = n \times (6+3) = 36 > 4 \times (n \times 2) = 32$

If the retransmission buffers are not to be used for deadlock recovery, then this lower limit for the total buffer size is no longer necessary. Therefore, if we forego deadlock recovery support, only three retransmission buffers will be needed per VC for link error correction (see Section 3.1), regardless of the regular transmission buffer size.

3.2.2 Probing for Deadlock Detection and Recovery

To detect possible deadlocks, most of the previous approaches adopted a threshold value of blocked cycles, after which the router assumed that the blocked flit was involved in a deadlock. This approach is guaranteed to detect all possible deadlocks [28]. However, it can also give false positives, where a node assumes a deadlock even though the flit is simply experiencing long blocking delay. Increasing the triggering threshold value will decrease the number of false positives, but increasing the threshold value arbitrarily will cause the number of blocked flits in the network to increase. In order to predict the most appropriate threshold value, one needs to consider a multitude of parameters, such as the network traffic among nodes, the traffic load, the routing algorithm and the deadlock recovery scheme. This can be a daunting feat, since the exploration space is huge.

To overcome this limitation, we aim to formulate a different methodology, which will detect only actual deadlocks without any false positives; this optimizes network performance, while eliminating the need to precisely identify an optimal threshold value.

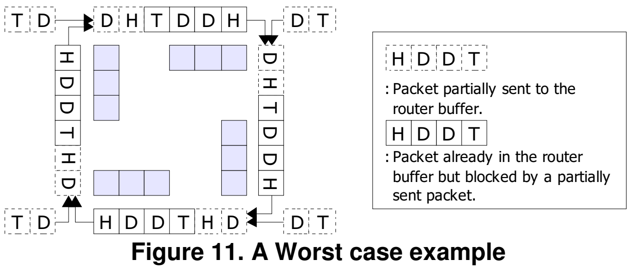


Figure 11. A Worst case example

We propose a probing technique, whereby a compact probing signal is sent along the suspected deadlock path after a flit has experienced more than a predefined number of cycles (***C***) of blocking.

The probe will check whether the flit is involved in a real deadlock or not. While the selection of C_{thres} will also affect network performance (as the threshold value described above), its impact is less pronounced because the probing technique will ensure that no action is prematurely taken. In other words, the threshold itself does not initiate deadlock recovery. The probing technique will first assess the situation to prevent the occurrence of any false positives. Therefore, the value of C_{thres} need not be precisely calculated; its effect on overall network performance will be minimal as long as the value chosen is not excessively high.

The proposed probing technique detects a deadlock based on the following two rules:

Rule 1: *After a flit experiences more than* C_{thres} *cycles of blocking, the router sends a probing signal to the next node specifying the VC buffer of the suspected flit.*

Rule 2: *When a node receives a probing signal, it checks the status of the buffer specified in the probing signal. If the VC buffer is also blocked in the current node or the node is in deadlock recovery mode, it forwards the probing signal to the next node, modifying the VC identifier accordingly. Otherwise, it discards the probing signal.*

If the probing signal returns to the original sender node, then the latter can safely assume that the flit under investigation is involved in a deadlock configuration; this is because the probing signal can return to the sender only if there is a cyclic path dependency and all intermediate nodes also experience blocking (by Rule 2). If increased blocking delay due to a hard failure causes a node to suspect deadlock, the subsequent probing signal will be discarded by the router adjacent to the faulty node, which will redirect blocked flits to another direction using an adaptive routing scheme, breaking the deadlock if any.

After the probe returns, the sender sends an activation signal that triggers the nodes involved in the deadlock to switch to the deadlock recovery mode. The sender node switches to the deadlock recovery mode after the activation signal returns. To handle the case where multiple nodes in the same deadlock configuration send probing signals at the same time, we need two more rules:

Rule 3: *A node will discard an activation signal unless it has received a probing signal from the same sender node before.*

Rule 4: *If a node receives a valid activation signal (as per Rule 3), while it is waiting for its own probe to return, it switches to the deadlock recovery mode and discards its own probe when it finally returns, since the deadlock recovery mode has already been activated by another node involved in the same deadlock configuration.*

To avoid incurring any additional overhead in supporting dedicated probing lines, we propose using a regular flit transmission for the probing signal, which can use the retransmission buffers in each suspected node to propagate. Note that the retransmission buffers are empty in nodes experiencing long blocking. This will ensure that the probing signal itself will not be blocked in an intermediate router. Figure 3 shows how an incoming link can feed the retransmission buffer directly. Since the probing signal is a regular flit, it will also be protected by the error correcting blanket, thus ensuring its safe traversal through the network.

4. Handling Soft Errors in Intra-Router Logic

Until recently, soft errors were tackled within the context of memory cells or registers. This has led to the widespread use of error detection and correction circuits to protect memory arrays. Combinational logic circuits, on the other hand, have been found to be less susceptible to soft errors in equivalent device technologies due to the naturally occurring logical, electrical and latching-window masking effects [31]. However, decreasing feature sizes and higher operating frequencies are rapidly thinning the protective effect of these masking phenomena. As mentioned before, research has indicated an exponential increase in the soft error rate (SER) per chip of logic circuits in the future [14]. Hence, it is crucial that modern router designs account for these events to ensure reliable and uninterrupted operation of the on-chip network. The notion of logic errors resulting from soft error upsets is directly related to the number of pipeline stages within the router. While the proposed measures are the same for all implementations, the recovery process differs depending on the number of pipeline stages present (and, thus, the amount of speculation employed by the architecture). The following sub-sections discuss the effects of soft errors on each router component along with proposed counter-measures. The recovery process for the different pipeline implementations is also analyzed. The latency overhead in the cases of 2-stage and 1-stage routers assumes successful speculative allocation in the recovery phase. Mis-speculation will increase the overhead, but mis-speculation occurs during normal operation as well and is unpredictable.

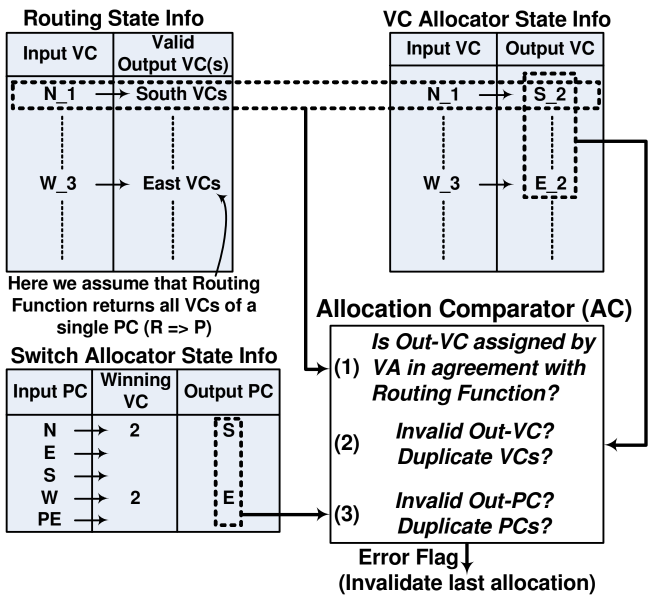

Routing State Info
Input VC
Valid Output VC(s)
N_1 → South VCs
W_3 → East VCs
Here we assume that Routing Function returns all VCs of a single PC (R => P)
VC Allocator State Info
Input VC
Output VC
N_1 → S_2
W_3 → E_2
Allocation Comparator (AC)
(1) Is Out-VC assigned by VA in agreement with Routing Function?
(2) Invalid Out-VC? Duplicate VCs?
(3) Invalid Out-PC? Duplicate PCs?
Switch Allocator State Info
Input PC
Winning VC
Output PC
N → 2 S
E
S
W → 2 E
PE
Error Flag
(Invalidate last allocation)

Figure 12. The Allocation Comparator (AC) unit.
The AC unit uses state information from the three router units (RT, SA, and VA) to perform three computations in parallel.

4.1. Virtual Channel Allocator Errors

The VA, like the routing unit, operates only on header flits. All new packets request access to any one of the valid output VCs, returned by the routing function. The VA arbitrates between all those packets requesting the same output VC. The VA maintains states of all successful allocations through a pairing between input VCs and allocated output VCs. It is this state that effectively opens up the "wormhole" for all subsequent flits of the same packet. Soft errors within the VA may give rise to four different scenarios:

(1) **One input VC is assigned an invalid output VC:** For example, suppose a PC has 3 VCs – designated by 00, 01, and 10. A soft error might cause the assignment of invalid VC 11. Such an assignment will block further traversal of the packet through the network.

(2) **An unreserved output VC is assigned to two different input VCs:** This will lead to packet mixing, and, eventually packet/flit loss. Flits from both packets will follow the same wormhole, since they are seen as one packet by the routers. As soon as the tail flit of one of the two packets releases the wormhole, any subsequent flits of the other packet will essentially be stranded in the network. For example, incoming packets from the North and West both can be assigned the same output VC in the South.

(3) **A reserved output VC is assigned to a requesting input VC:** This case is very similar to case (2) above. The new packet will erroneously follow the existing wormhole,

following a path to a wrong destination. The same consequences will result as above.

(4) **An erroneous, yet unreserved, output VC is assigned to a requesting input VC:** In this scenario, there are two different types of erroneous output VC assignments:
(a) ***The wrong output VC belongs to the intended PC.*** This is a benign case, since the packet will still be forwarded to the same physical direction as originally intended.
(b) ***The wrong output VC belongs to a PC other than the intended one.*** This case is similar to the misdirection situation to be analyzed in Section 4.2. It may lead to deadlock in deterministic routing algorithms. The solution described in Section 4.2 will also protect against this type of error.

The proposed safeguard for VA logic errors is the addition of a compact hardware unit, called the Allocation Comparator (AC). The proposed AC unit is shown in Figure 12. The unit employs purely combinational logic, in the form of XOR gates, to compare the RT state entries, SA state entries, and the VA state entries. The AC unit performs three types of comparisons in parallel, within one clock cycle. It first checks to see if the output VCs assigned by the VA unit are in accordance with the output of the routing function (i.e. RT unit). For instance, if a soft error causes the VA to erroneously assign an output VC in the North PC, while the RT unit had indicated the assignment of a VC in the South PC, the AC unit will trigger an error flag, thus invalidating the VA allocation of the previous clock cycle. This comparison protects against scenario (4b) above. Secondly, the AC unit checks the VA state info to detect both invalid and duplicate output VC assignments. Should any of these cases appear, an error flag is raised. This comparison safeguards against scenarios (1) through (3) above. Finally, the AC unit checks for Switch Allocation errors as discussed in the following sub-section.

The duration of the recovery phase is independent of the pipeline architecture. In all cases except the 4-stage router, parallelization implies that the AC unit will operate in the same stage as the crossbar traversal (i.e. after the VA operation concludes in Figure 2). This means that if an error is detected by the AC unit, a NACK should be sent to all neighboring routers to ignore the previous transmission. Then the previous VA allocations are repeated in the current router, thus incurring single-clock latency overhead. In a 4-stage router, the AC unit will detect the error by the end of stage 3 (i.e. before crossbar traversal); therefore, no erroneous transmission will occur. The latency delay is still one clock cycle.

While adding additional hardware increases the overall area and power consumption of the router, the proposed unit was deliberately architected to be as small and efficient as possible. First, the number of state entries to be compared is equal to PV, where P is the number of input/output ports and V is the number of VCs per port. For a typical 5-port mesh NoC router (North, East, South, West, PE) with 4 VCs per PC, the number of entries is 5x4=20. The size of the entries is minimal, since the VC IDs are only a few bits long (e.g. 2 bits for 4 VCs per PC). Thus, the data to be compared is very small. To validate the architecture's compactness and efficiency, we synthesized the comparator unit in 90nm technology. The area and power budgets of the unit, as compared to the total budget for a generic NoC router (see Figure 1), are shown in Table 1. It is evident that the AC unit imposes a minimal area and power penalty on the overall design, while providing full protection from logic errors. Moreover, the AC unit is also used to protect against Switch Allocator errors, described in Section 4.3, further subsidizing its small additional overhead.

It should be noted that almost all the overhead in the proposed protective scheme comes from the AC unit alone; the VA and SA are only slightly modified to accept invalidation signals from the AC. It is the AC unit which monitors the results of the VA and SA.

Additional hardware components to combat faults bring in the possibility of a fault within the new component. In this work, we assume single event upsets, i.e., only one fault could happen at any given time. This implies that there might be a fault in the VA, or SA, or the AC unit at a given time, but not in more than one of them. Hence, if there is a soft error in the VA or SA unit, it will definitely be caught by the AC unit. If, on the other hand, there is a false positive due to a fault in the AC unit, then the consequence is benign; all that the AC unit does is invalidate the allocations in the previous clock cycle. Therefore, a false positive will simply waste one clock cycle in arbitrations.

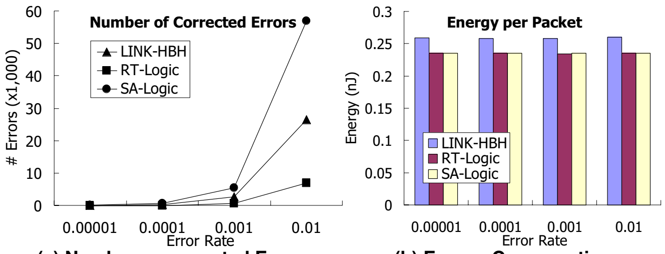
4.2. Routing Unit Errors

A transient fault in the routing unit logic could cause a packet to be misdirected. Since the subsequent virtual channel allocation and switch allocation would be performed based on the misdirection, no data corruption will occur. The erroneous direction, however, may be blocked, either because of a link outage (hard fault), or a network edge in various topologies (e.g. mesh). This will be caught by the VA, which maintains the state information for its adjacent routers. The VA is able to detect such erroneous behavior, because the allocator is aware of blocked links or links which are not permitted due to physical constraints (e.g. a network edge); they are either pre-programmed in the allocator's state table or they are dynamically specified through incoming state information from adjacent routers. The recovery, however, depends on whether look-ahead routing is used or not. If such a routing is employed, then the error will be caught by the VA of the following router and reported to the previous router through an appropriate NACK message. This will invalidate the erroneous decision and force the routing unit to repeat the routing process on the specific packet.

Note that the header flit is still in the previous router's retransmission buffer (as described in Section 3). The whole recovery process will take 3 clock cycles (one for the NACK

Table 1. Power and Area Overhead of the AC Unit

Component	Power	Area
Generic NoC Router (5 PCs, 4 VCs per PC)	119.55 mW	0.374862 mm^2
Allocation Comparator (AC)	2.02 mW (+1.69% overhead)	0.004474 mm^2 (+1.19% overhead)



(a) Number or corrected Errors (b) Energy Consumption

Figure 13. Impact of soft-error correcting schemes

propagation to the previous router, one for the new routing process, and one for the retransmission) in a 2-stage router. It would take 2 clock cycles in a single-stage router (one for the NACK and one for the new routing process and retransmission). To increase the efficiency in non-minimal, adaptive routing schemes, the current router may reset the invalid direction and assign a new direction without informing the previous router. If current-node routing is used (i.e. 4-stage and 3-stage routers), then the recovery phase is simpler, since the error is caught by the VA in the same router, which will inform the routing unit before the transmission occurs. This will incur a single-cycle delay for re-routing.

Misdirection to a non-blocked, functional path, however, will not be caught by the VA, since its state information will not raise an error flag (i.e. an error signal, as shown in Figure 12). It could potentially cause deadlock in deterministic routing algorithms. In such algorithms, however, the error will be detected in the router that receives the misdirected flit. A NACK to the sending router would then fix the problem. The latency overhead is dependent on the number of pipeline stages (n) within the router. The delay penalty is equal to 1 + n, (NACK + re-routing and retransmission). In adaptive routing schemes, the error cannot be detected. However, in such schemes, a misdirection fault is not catastrophic; it simply delays the flit traversal.

4.3. Switch Allocator Errors

A switch allocator error could give rise to the following four different problems, some of which would lead to packet/flit loss: (a) A soft error in the control signals of the switch allocator could prevent flits from traversing the crossbar. This case is the least problematic, since the flits will keep requesting access to the crossbar until they succeed. (b) If a data flit is mistakenly sent to a direction different from the header flit, it would cause flit/packet loss, because it would deviate from the wormhole created by its header flit. (c) A soft error could cause the allocator to direct two flits to the same output. This will lead to a corrupt flit, which will be detected by the error detection code in the next router. A NACK will be sent and the correct flits retransmitted from the retransmission buffer. Regardless of the number of pipeline stages, this error recovery process will incur two cycles (NACK + retransmission) latency overhead. (d) An error could cause the allocator to send a flit to multiple outputs (multicasting). If the flit is a data flit, the same error will occur as case (b) above. If the flit is a header flit, then multiple virtual channels will mistakenly be reserved in all the receiving routers (essentially opening multiple wormholes for the same message). Those wormholes will stay permanently reserved, thus reducing the effective buffer space in those routers.

The most challenging cases are (b) and (d). To prevent such scenarios, we propose use of the Allocation Comparator (AC) Unit, which was introduced to protect against VA errors. As shown in Figure 12, the AC unit also checks for invalid SA allocations (such as multicasting) and duplicate SA allocations; upon detection of an erroneous behavior, the AC unit will invalidate the SA allocation in the previous clock cycle. In this case, the overhead involved does not depend on the number of pipeline stages of a router. In all cases, an SA error will be caught by the AC unit after the SA stage finishes. This implies that the AC unit will be operating in the same stage as crossbar traversal. Therefore, a NACK signal must be sent to all adjacent routers to ignore the previous transmission, and a new SA process will commence; this amounts for single-clock latency overhead.

We examined the impact of our proposed solutions by simulating three types of error situations. These are routing logic errors (RT-Logic), switch allocator logic errors (SA-Logic) and link errors (LINK-HBH). Each one of the cases was simulated independently by varying the error rate and measuring the number of errors corrected and energy consumption per message. Figure 13 (a) illustrates the number of errors corrected by the proposed measures. Errors in the routing unit are significantly less than errors in the SA, since routing errors occur only in header flits. The SA, however, operates on every flit, and many flits often undergo multiple arbitrations before winning access to the switch. Link traversal, on the other hand, only occurs once for each flit per hop, thus the link errors detected in this experiment were less than the SA errors. Figure 13 (b) depicts the energy consumed per packet under the different error schemes. As shown, link errors induce more energy overhead because of retransmissions. Nevertheless, even with retransmissions, the overhead is still minimal, thereby validating our previous assertions.

4.4. Crossbar Errors

A transient fault within the crossbar would produce single-bit upsets, not entire flits being misdirected as in the switch allocator case. Single-bit upsets are taken care of by the error detection and correction unit employed within each router, thus eliminating the problem.

4.5. Retransmission Buffer Errors

A single soft error in the retransmission buffer would be corrected by the error-correcting unit in the receiving router. A double (or more) error, however, would yield an endless retransmission loop since the original data itself is now corrupt. Given that a double bit-flip is highly unlikely, such a scenario can be ignored. However, a fool-proof solution would be to use duplicate retransmission buffers. This will double the buffer area and power overhead.

4.6. Handshaking Signal Errors

Every router has several handshaking signal lines with neighboring routers to facilitate proper functionality and synchronization. Transient faults on these lines would disrupt the operation of the network. Since the number of handshaking signal lines is small, Triple Module Redundancy (TMR) can be used, in which three lines and a voter are used to ensure protection against soft errors. There is a slight area and power overhead increase, but the area occupied by these lines is negligible compared to the area of the other router components.

5. Conclusions

In this paper we presented a comprehensive plan of attack on various types of reliability hindrances in on-chip networks. We have tackled most common failure types by proposing a series of architectural techniques, which work in tandem to protect the interconnect infrastructure.

A new hop-by-hop retransmission scheme was presented to combat link errors. The scheme was shown to be very efficient in terms of both latency and power even under high error rates. The retransmission buffers required by this mechanism were also used in a newly proposed deadlock recovery technique, which utilizes existing resources to break deadlocks, thus minimizing the incurred overhead. Finally, a detailed analysis of possible symptoms resulting from intra-router logic errors was also presented, along with an array of protective measures and their effectiveness in various router architectures.

More importantly, all the mechanisms proposed in this work kept the critical path of the NoC router intact. For on-chip networks, ultra-low latencies are an absolute necessity; thus, any reliability solution which inflicts significant burden on latency is not well suited. Our schemes work in parallel with the critical components without deepening the router pipeline.

6. References

[1] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, "Design and analysis of an NoC architecture from performance, reliability and energy perspective," in Proc. of the Symposium on Architecture for networking and communications systems (ANCS), 2005.

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

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

[4] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in Proc. of the Design Automation Conference (DAC), 2001.

[5] T. D. Richardson, C. Nicopoulos, D. Park, V. Narayanan, X. Yuan, C. Das, and V. Degalahal, "A Hybrid SoC Interconnect with Dynamic TDMA-Based Transaction-Less Buses and On-Chip Networks," in Proc. of the International Conference on VLSI Design, pp. 657-664, 2006.

[6] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," Design & Test of Computers, IEEE, vol. 22, pp. 434-442, 2005.

[7] K. Constantinides, S. Plaza, J. Blome, Z. Bin, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "BulletProof: A Defect-Tolerant CMP Switch Architecture," in Proc. of the High-Performance Computer Architecture (HPCA), pp. 3-14, 2006.

[8] A. Lorena and N. Michael, "Cost reduction and evaluation of temporary faults detecting technique," in Proc. of the Design, Automation and Test in Europe (DATE), 2000.

[9] D. Bertozzi, L. Benini, and G. De Micheli, "Low power error resilient encoding for on-chip data buses," in Proc. of the Design, Automation and Test in Europe Conference (DATE), pp. 102-109, 2002.

[10] R. Marculescu, "Networks-on-chip: the quest for on-chip fault-tolerant communication," in Proc. of the symposium on VLSI, pp. 8-12, 2003.

[11] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in Proc. of the International Conference on Computer-Aided Design (ICCAD), pp. 524-531, 1996.

[12] A. Krstic, J. Yi-Min, and C. Kwang-Ting, "Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, pp. 416-425, 2001.

[13] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 225-232, 2003.

[14] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in Proc. of the Dependable Systems and Networks (DSN), pp. 389-398, 2002.

[15] L. S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in Proc. of the High Performance Computer Architecture (HPCA), pp. 255-266, 2001.

[16] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," in Proc. of the Hot Interconnects Symposium IV, 1996.

[17] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in Proc. of the Design Automation Conference (DAC), pp. 559-564, 2005.

[18] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in Proc. of the International Symposium on Computer Architecture (ISCA), pp. 188-197, 2004.

[19] S. Arjun, W. J. Dally, A. K. Gupta, and B. Towles, "GOAL: a load-balanced adaptive routing algorithm for torus networks," in Proc. of the International Symposium on Computer Architecture (ISCA), pp. 194-205, 2003.

[20] R. S. Srinivasa and R. S. Naresh, "Coding for system-on-chip networks: a unified framework," in Proc. of the Design Automation Conference (DAC), 2004.

[21] 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.

[22] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in Proc. of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 188-193, 2003.

[23] W. J. Dally and B. Towles, Principles and practices of interconnection networks: Morgan Kaufmann, 2003.

[24] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs," in Proc. of the International Conference on Computer Design (ICCD), pp. 536- 539, 2003.

[25] W. J. Dally, L. R. Dennison, D. Harris, K. Kan, and T. Xanthopoulos, "Architecture and implementation of the reliable router," in Proc. of the Hot Interconnects II, pp. 197-208, 1994.

[26] W. Jie, "A deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes based on odd-even turn model," in Proc. of the International Conference on Supercomputing (ICS), pp. 67-76, 2002.

[27] N. Ted and S. L. Johnsson, "ROMM routing on mesh and torus networks," in Proc. of the Symposium on Parallel Algorithms and Architectures (SPAA), 1995.

[28] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," Parallel and Distributed Systems, IEEE Transactions on, vol. 4, pp. 1320-1331, 1993.

[29] K. V. Anjan and T. M. Pinkston, "An efficient, fully adaptive deadlock recovery scheme: DISHA," in Proc. of the International Symposium on Computer Architecture (ISCA), pp. 201-210, 1995.

[30] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection networks: An engineering Approach.," Los Alamitos, Calif., IEEE Computer Society, 1997.

[31] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson, "On latching probability of particle induced transients in combinational networks," in Proc. of the Symposium on Fault-Tolerant Computing (FTCS), pp. 340-349, 1994.