Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 37627, 11 pages
doi:10.1155/2007/37627

Research Article

A Method for Routing Packets Across Multiple Paths in NoCs with In-Order Delivery and Fault-Tolerance Gaurantees

Srinivasan Murali,[1] David Atienza,[2,3] Luca Benini,[4] and Giovanni De Micheli[3]

[1] *Computer Systems Lab, Stanford University, Stanford, CA 94305-9040, USA*
[2] *Departamento Arquitectura de Computadores y Automatica, Universidad Complutense de Madrid, 28040 Madrid, Spain*
[3] *Laboratoire des Systèmes Intégrés, Ecole Polytechnique Federale de Laussanne, 1015 Lausanne, Switzerland*
[4] *Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, 40126 Bologna, Italy*

Received 16 October 2006; Revised 21 January 2007; Accepted 6 February 2007

Recommended by Maurizio Palesi

Networks on Chips (NoCs) are required to tackle the increasing delay and poor scalability issues of bus-based communication architectures. Many of today's NoC designs are based on single path routing. By utilizing multiple paths for routing, congestion in the network is reduced significantly, which translates to improved network performance or reduced network bandwidth requirements and power consumption. Multiple paths can also be utilized to achieve spatial redundancy, which helps in achieving tolerance against faults or errors in the NoC. A major problem with multipath routing is that packets can reach the destination in an out-of-order fashion, while many applications require in-order packet delivery. In this work, we present a multipath routing strategy that guarantees in-order packet delivery for NoCs. It is based on the idea of routing packets on partially nonintersecting paths and rebuilding packet order at path reconvergent nodes. We present a design methodology that uses the routing strategy to optimally spread the traffic in the NoC to minimize the network bandwidth needs and power consumption. We also integrate support for tolerance against transient and permanent failures in the NoC links in the methodology by utilizing spatial and temporal redundancy for transporting packets. Our experimental studies show large reduction in network bandwidth requirements (36.86% on average) and power consumption (30.51% on average) compared to single-path systems. The area overhead of the proposed scheme is small (a modest 5% increase in network area). Hence, it is practical to be used in the on-chip domain.

1. INTRODUCTION

Future *Systems on Chips (SoCs)* will have multiple processing cores and memories with each core having high performance and frequency of operation. There is a growing trend towards integrating multiple applications onto the same SoC, so that the user needs to carry only one device that performs a host of operations. As the computational load of the SoC increases, so does the load on the communication architecture. Scalable micro networks (or *Networks on Chips (NoCs)*) are needed to provide high bandwidth communication infrastructure for the SoCs [1–7]. With technology scaling, on-chip wires are increasingly susceptible to various error sources such as crosstalk, coupling noise, ground bounce, soft errors, process variations, and interconnect failures. The use of NoCs facilitate the application of network error resiliency techniques to tolerate such transient and permanent errors in the interconnects.

The routing scheme used in the NoC can be either static or dynamic in nature. In static routing, one or more paths are selected for the traffic flows in the NoC at design time. In the case of dynamic routing, the paths are selected based on the current traffic characteristics of the network. Due to its simplicity and the fact that application traffic can be well characterized for most SoC designs, static routing is widely employed for NoCs [8]. When compared to static single-path routing, the static multipath routing scheme improves path diversity, thereby minimizing network congestion and traffic bottlenecks. When the NoC is predesigned, with the NoC having a fixed operating frequency, data width, and hence bandwidth (bandwidth available on each network link is the product of the link data width and the NoC operating frequency), reducing congestion results in improved network performance. For most SoC designs, the NoC operating frequency can be set to match the application requirements. In this case, reducing the traffic bottlenecks leads to lower

Table 1: Reorder buffer overhead.

Design	Area (sq mm)	Power (mW)
(1) Base NoC	1.04	183.75
(2) 2 buffers/core	1.14	201
(3) 10 buffers/core	1.65	281.25

required NoC operating frequency, as traffic is spread evenly in the network, thereby reducing the peak link bandwidth needs. A reduced operating frequency translates to a lower power consumption in the NoC. As an example, consider an MPEG video application mapped onto a 4×3 mesh NoC. Detailed analysis of the application and the performance of traditional single path schemes and the proposed multipath scheme are presented later in this work (in Section 7.2). When the NoC operating frequency for the schemes is set so that both schemes provide the same performance level (same average latency for traffic streams), the multipath scheme results in 35% reduction in network operating frequency, leading to 22.22% reduction in network power consumption (after accounting for the overhead involved in the multipath scheme). Another important property of the multipath routing strategy is that there is spatial redundancy for transporting a packet in the on-chip network. A packet can be sent across multiple paths for achieving resiliency against transient or permanent failures in the network links.

Many of today's NoC architectures are based on static single path routing. This is because, with multipath routing, packets can reach the destination in an out-of-order fashion due to the difference in path lengths or due to difference in congestion levels on the paths. For many applications, such out-of-order packet delivery is not acceptable and packet reordering is needed at the receivers. As an example, in chip multiprocessor applications for maintaining coherency and consistency, packets reaching the destination need to be in-order. In video and other multimedia applications, packet ordering needs to be maintained for displays and for many of the processing blocks in the application.

With multipath routing, packet reorder buffers can be used at the receiver to reorder the arriving packets. However, the reorder buffers have large area and power overhead and deterministically choosing the size of them is infeasible in practice. In Table 1, the area and power consumption of a 4×3 mesh NoC (the area-power values include the area power of the switches, links, Network Interfaces (NIs)) with different numbers of packet buffers in the receiving NIs is presented. The network operating frequency is assumed to be 500 MHz with 50% switching activity at the subcomponents. The flit size is assumed to be 16 bits with the base switch/NI having 4 flit queuing buffers at each output. The base NoC component area, power values are obtained from synthesizing the component designs that are based on the ×pipes architecture [9] (refer to Section 7 for a description of the architecture) using UMC 0.13 μm technology library. As seen from the table, a NoC design with 10 packet reorder buffers/core has 59% higher NoC area and 43% higher NoC power consumption when compared to the base NoC without reorder buffers. Another important point is that at design time it is not possible to size the reorder buffers to prevent packets from being dropped at the receiver. As an example, if a packet travels a congested route and takes an arbitrarily long time to reach the destination, several subsequent packets that take a different route can reach the destination before this packet. In this case, the reorder buffers, unless they have infinite storage capacity, can be full for a particular scenario and can no longer receive packets. This leads to dropping of packets to recover from the situation and requires end-to-end ACK/NACK protocols for resuming the transaction.

End-to-end ACK/NACK protocols are used in most macro networks for error recovery and in such networks, these protocols are extended to handle this packet buffering problem as well [10]. However, such protocols have significant overhead in terms of network resource usage and congestion. Thus, they are not commonly used in the NoC domain [10, 11]. Moreover, the performance penalty to recover from such a situation can be very high and most applications cannot tolerate such variations in performance. This motivates the need to find efficient solutions to the packet reordering problem for the on-chip domain.

To the best of our knowledge, this is the first work that presents a multipath routing strategy with guaranteed in-order packet delivery (without packet dropping) for on-chip networks. It is based on the idea of routing packets on *non-intersecting* paths and rebuilding packet order at *path reconvergent* nodes. By using an efficient flow control mechanism, the routing strategy avoids the packet dropping situation that arises in the traditional multipath routing schemes. We present algorithms to find the set of paths in a NoC topology to support the routing strategy and present a method to split the application traffic across the paths to obtain a network with minimum power consumption. We explore the use of temporal and spatial redundancy during multipath routing to provide resilience against temporary and permanent errors in the NoC links. When sending multiple copies of a packet, it is important to achieve the required reliability level for packet delivery with minimum data replication. We integrate reliability constraints in our multipath design methods to provide a reliable NoC operation with least increase in network traffic. Experiments on several benchmarks show large power savings for the proposed scheme when compared to traditional single-path schemes and multipath schemes with reorder buffers. The area overhead of the proposed scheme is small (a modest 5% increase in network area). Hence, it is practical to be used in the on-chip domain.

2. PREVIOUS WORK

Several researchers have been focusing on NoC design issues and a variety of NoC architectures and platforms have been proposed [1–7]. Many works on mapping of applications onto NoC architectures have considered the routing problem during the NoC design phase [8, 12–15]. In [16], a low latency router architecture for supporting dynamic routing is presented. In [17], a routing scheme that switches between

deterministic and adaptive modes, depending on the application requirements, is presented. All these works assume that the architectural support needed for such routing schemes (such as packet reorder buffers) are available in the NoC.

Several works in the multiprocessor field have focused on the design of efficient routing strategies [18]. In the Avici router [19], packets that need to be in-order at the receiver are grouped together into a *flow*. Packets of a single flow follow a single path, while different flows can use different paths. In the IBM SP2 network [20], source-based oblivious routing is used for a multistage interconnection network. In [21], the authors present a source-based dynamic routing algorithm for multistage networks. In both works, a single path is used for packets that require in-order delivery and multiple paths are used for packets that do not require ordering.

Several research works have focused on designing reliable NoC systems [11, 22–28]. In [24], fault-tolerant stochastic communication for NoCs is presented. The use of nonintersecting paths for achieving fault-tolerant routing has been utilized in many designs, such as the IBM Vulcan [18]. The use of temporal and spatial redundancy in NoCs to achieve resilience from transient failures is presented in [28].

Unlike earlier works, we assume that each packet can be routed on different paths. We present algorithms and design methods to support the multipath routing strategy. We explore the use of temporal and spatial redundancy during multipath routing to provide resilience against temporary as well as permanent errors in the NoC links. The routing methods can either be applied after NoC topology mapping and design or during the mapping process. In this work, we only present the details of the design of the routing strategy. We refer the interested readers to existing works [8, 12–15] on mechanisms to integrate different static routing methods with NoC topology mapping. The basic multipath-routing strategy has been presented by us in [29]. In this work, we integrate the routing mechanism into a design methodology which makes it effective for applying the scheme to NoCs.

3. MULTIPATH ROUTING WITH IN-ORDER DELIVERY

In this section, we present the conceptual idea of the multipath routing strategy with in-order packet delivery. For analysis purposes, we define the NoC topology by the NoC topology graph.

Definition 1. The topology graph is a directed graph $G(V, E)$ with each vertex $v_k \in V$ representing a switch/NI in the topology and the directed link (or edge) $e_l \in E$ representing a direct communication between two switches/NIs. We represent the traffic flow between a pair of cores in the NoC as a commodity i with the source switch/NI of the commodity being s_i and the destination of the commodity being d_i. Let the total number of commodities be I. The rate of traffic transferred by commodity i is represented by r_i.

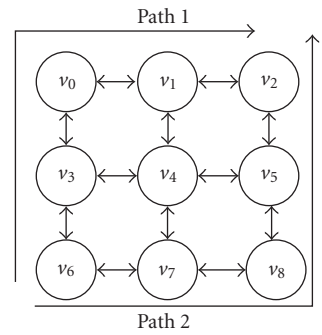


Figure 1: Example mesh topology.

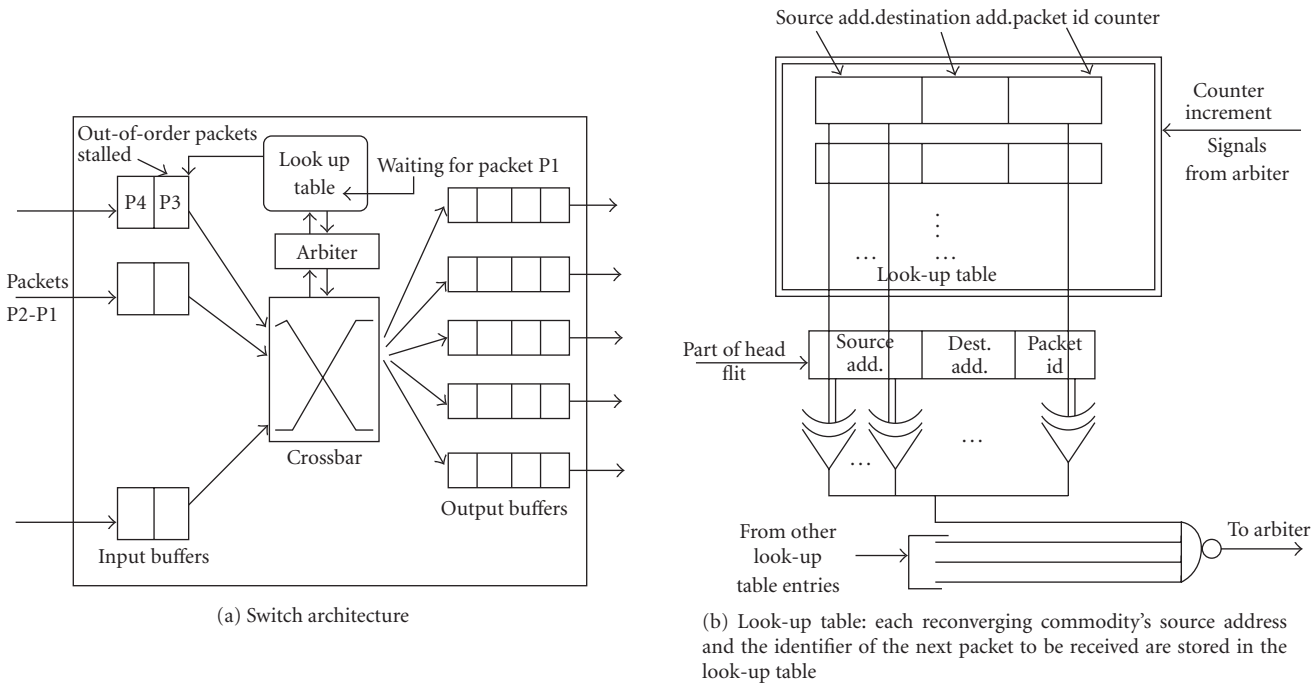
An example NoC topology graph for a 3×3 mesh NoC is shown in Figure 1. Since in the mesh network each NI is connected to only one switch, we represent the topology graph by using only the switches. For topologies where each NI is connected to multiple switches, the NIs are also taken as part of the topology description.

The traffic rate for each commodity (r_i) can either be the average rate of communication between the source and destination of the commodity or can be obtained in an efficient manner that considers the *Quality-of-Service (QoS)* provisions for the application. In this work, we assume that the efficient numbers for the rates are obtained as presented in [14]. The traffic rates are computed considering the peak and average bandwidth needs for the commodity, burstiness in traffic, deadlines and slacks associated with the bursts [14]. We define the paths for the traffic flow of a commodity as follows.

Definition 2. Let the set SP_i represent the set of all paths for the commodity i, for all $i \in 1 \cdots I$. Let P_i^j be an element of SP_i, for all $j \in 1 \cdots |\mathrm{SP}_i|$. Thus P_i^j represents a single path from the source to destination for commodity i. Each path P_i^j consists of a set of links.

Example paths for the commodity (traffic flow) from source vertex v_6 to destination v_2 are shown by the solid lines in Figure 1. We define a set of paths to be *nonintersecting* if the paths originate from the same source vertex but do not intersect each other in the network, except at the destination vertex. The two paths shown in Figure 1 are *nonintersecting*. Consider packets that are routed on the two *nonintersecting* paths. Note that with worm-hole flow control [18], packets of a commodity on a particular path are in-order at all time instances. However, packets on the two different paths can be out-of-order. As an example, if packets 1, 2 are sent on *path* 1 and packets 3, 4 are sent on *path* 2 and if *path* 2 is faster (either because it is shorter or because of lower congestion), then packets 3, 4 can reach the destination before packets 1, 2. Therefore, we need a mechanism to reorder the packets at the *reconvergent* nodes to maintain the packet ordering.

To implement the reordering mechanism at network reconvergent nodes, the following architectural changes to the



(a) Switch architecture

(b) Look-up table: each reconverging commodity's source address and the identifier of the next packet to be received are stored in the look-up table

Figure 2: Switch design to support multipath routing with in-order packet delivery.

switches/NIs of the NoC are required (shown in Figure 2). We assume that the packet is divided into multiple flow control units called *flits*. The first flit of the packet (known as the *header* flit) has the routing information for the packet. To support multipath routing, individual packet identifiers are used for packets belonging to a single commodity. At the *reconvergent* switch, we use a look-up table to store the identifier of the next packet to be received for the commodity. Initially (when the NoC is reset), the identifiers in the look-up tables are set to 1 for all the commodities. When packets arrive at the input of the *reconvergent switch*, the identifier of the packet is compared with the corresponding look-up table entry. If the identifiers match, the packet is granted arbitration and the look-up table identifier value for this commodity is incremented by 1. If the identifiers do not match, then this is an out-of-order packet and access to the output is not granted by the arbiter circuit, and it remains at the input buffer.

As the packets on a particular path are in-order, the mechanism only stalls packets that would also be out-of-order if they reach the switch. Due to the disjoint property of the paths reaching the switch, the actual packet (matching the identifier on the look-up table) that needs to be received by the switch is on a different path. As a result, such a stalling mechanism (integrated with *credit-based* or *on-off* flow control mechanisms [18]) does not lead to packet dropping, which is encountered in traditional schemes when the reorder buffers at the receivers are full. Note that routing-dependent deadlocks that can occur in the network can be avoided using virtual channel flow control [18]. Other routing dependent deadlock free methods, such as presented in [30], can also be used in conjunction with the methods presented in this work.

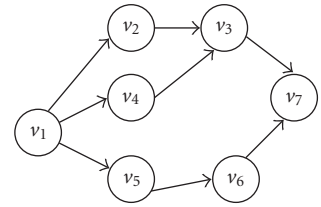
4. PATH SELECTION ALGORITHM

In this section, we describe the algorithms that can be used to efficiently find *nonintersecting paths* for each commodity of the NoC. As in general, the number of paths between a source and destination vertex of a graph is exponential, we present heuristic algorithms to compute the paths [31]. For each commodity, we first find the set of all possible paths for the commodity. Then, from the chosen paths, we find those paths that are nonintersecting. We use such a two-phase approach to achieve fast heuristic solutions to tackle the exponential problem complexity.

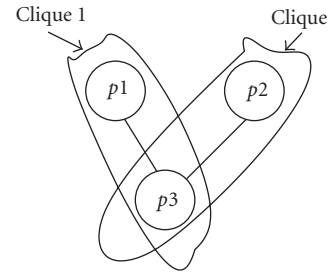
Consider a source vertex s_i (which corresponds to the source core/NI that sends a packet) and destination vertex d_i of a commodity i. Algorithm 1 is used to find the set of possible paths between the two vertices. Example 1 presented below illustrates how the working algorithm works. The objective of the algorithm is to find maximum number of paths possible, so that large path diversity is available for the traffic flow. In the algorithm, after finding a path, we remove one of the edges of the path so that the same path is not considered in further iterations. As most NoC vertices have only a small degree, we remove one of the middle edges (instead of the edges at the source and destination), because it helps in increasing the number of paths found. As in each iteration of the algorithm we remove an edge, the number of iterations (and hence the maximum number of paths found) is at most $|E|$ for one pair of source and destination vertices.

(1) Choose a path from the source to destination of the commodity using Depth First Search (DFS).
(2) Remove one of the middle edges of the chosen path.
(3) Repeat the above steps until there are no paths between the vertices.

Algorithm 1: Path selection algorithm for a single commodity.



(a) Example graph for path selection



(b) Compatibility graph for the example

Figure 3: Path selection and compatibility graph generation.

Example 1. Consider the NoC topology graph presented in Figure 3(a). The vertices represent switches/NIs in the NoC. Let v_1 and v_7 be the source and destination vertices of a traffic flow. In the first iteration of the algorithm, one of the paths (e.g., the path $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$) is chosen and the middle edge (edge from $v2 \rightarrow v3$) is removed. In the next iteration of the algorithm, another path ($v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$) is chosen and the edge $v4 \rightarrow v3$ is removed. In the last iteration $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ is chosen and the edge $v5 \rightarrow v6$ is removed, after which no more paths exist from $v1$ to $v7$. Note that if we had removed the edge $v3 \rightarrow v7$ in the first iteration, we would have obtained only two paths (instead of three paths).

The paths resulting from the algorithm may converge at one or more vertices. In order to obtain *nonintersecting* paths, we form a *compatibility graph* with each vertex of the graph representing a path. An edge between two vertices in the graph implies that the corresponding paths do not intersect. An example *compatibility graph* for the paths from Example 1 is shown in Figure 3(b). The objective is to obtain the maximum number of nonintersecting paths from the set of paths. This is equivalent to finding the maximum size clique[1] in the *compatibility graph*, which is a well-known NP-Hard problem [31]. We use a commonly used heuristic algorithm for finding the maximum clique (see Algorithm 2) [32]. The working of the algorithm is illustrated in Example 2. We repeat the two algorithms for all the commodities in the NoC. When applying Algorithm 1 for each commodity, we start with the original topology graph.

Example 2. The compatibility graph for the 3 paths from Example 1 is shown in Figure 3(b). The vertex $p1$ represents the path $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$, $p2$ represents the path $v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$, and $p3$ represents the path $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$. As the paths $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ do not intersect each other, there is an edge between $p1$ and $p3$ in the compatibility graph. Similarly, for the paths $v1 \rightarrow v4 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$, there is an edge between $p2$ and $p3$. There are two maximum size cliques in the graph (formed by $p1, p3$ and $p2, p3$) and one of them is arbitrarily chosen (say, $p1, p3$). Thus, the paths $v1 \rightarrow v2 \rightarrow v3 \rightarrow v7$ and $v1 \rightarrow v5 \rightarrow v6 \rightarrow v7$ are used for the traffic flow between vertices $v1$ and $v7$.

[1] Clique of a graph is a fully connected subgraph.

The time complexity of each iteration in Algorithm 1 is dominated by the *Depth-First Search (DFS)* procedure and is $O(|E| + |V|)$ [31]. The number of iterations is $O(|E|)$. The time complexity of the maximum clique calculation step is $O(|E|^2)$. The algorithms are repeated once for each commodity. Therefore, the run time of the nonintersecting path finding algorithms is $O(|I||E|(|E| + |V|))$. In practice, the run time of the algorithms is less than few minutes for all the experimental studies we have performed (presented in Section 7).

In cases where we are interested in having as many minimum paths as possible, we can modify the call to DFS in Algorithm 1 to a call to Dijkstra's shortest path algorithm, choosing shorter paths first. Then, Algorithm 2 can also be modified to first choose the minimum paths that are nonintersecting and then choosing nonminimum paths that are nonintersecting with each other and with the chosen minimum paths. Note that in situations where we only need few paths for each commodity (to have small route look-up tables), we can select the needed number of paths from the above algorithms. Similarly, for networks that require deadlock avoidance using restricted routing functions, we can use turn models to select the paths [8, 18], with only a marginal increase in the complexity of the presented algorithms.

5. MULTIPATH TRAFFIC SPLITTING

Once we have obtained the set of nonintersecting paths for each commodity, we need to determine the amount of flow of each commodity across the paths that minimizes congestion. Then, we can assign probability values for each path of every commodity, based on the traffic flow across that path for the commodity. At run time, we can choose the path for each packet from the set of paths based on the probability values assigned to them. To achieve this traffic splitting, we

(1) Build a compatibility graph for the paths and initialize the set MAX_CLIQUE to NULL.
(2) Add vertex with largest degree to MAX_CLIQUE.
(3) From remaining vertices, choose vertex that is adjacent to all vertices in set MAX_CLIQUE and add it to the set.
(4) Repeat the above step until no more vertex can be added.

Algorithm 2: Determining non-intersecting paths for a single commodity.

use a *Linear-Programming (LP)*-based method to solve the corresponding multicommodity flow problem. The objective of the LP is to minimize the maximum traffic on each link of the NoC topology, satisfying the bandwidth constraints on the links, and routing the traffic of all the commodities in the NoC. Our LP is represented by the following set of equations:

$$\min : \ t, \tag{1}$$

$$\text{s.t} \sum_{\forall j \in 1 \cdots |\ \text{SP}_i|} f_i^j = r_i, \quad \forall i, \tag{2}$$

$$\sum_{\forall i} \sum_{\forall j, e_l \in P_i^j} f_i^j = \text{flow}_{e_l}, \quad \forall e_l,$$
$$\text{flow}_{e_l} \leq \text{bandwidth}_{e_l}, \quad \forall e_l, \tag{3}$$

$$\text{flow}_{e_l} \leq t \quad \forall e_l \in P_i^j, \ \forall i, j, \tag{4}$$

$$f_i^j \geq 0. \tag{5}$$

In the objective function we use the variable t to represent the maximum flow on any link in the NoC (equations (1), (4)). Equation (2) represents the constraint that the NoC has to satisfy the traffic flow for each commodity with the variable f_i^j representing the traffic flow on the path P_i^j of commodity i. The flow on each link of the NoC and the bandwidth constraints are represented by (3). Other objectives (such as minimizing the sum of traffic flow on the links) and constraints (like latency constraints for each commodity) can also be used in the LP. As an example, the latency constraints for each commodity can be represented by the following equation:

$$\frac{\sum_{\forall j \in 1 \cdots |\text{SP}_i|} \left(f_i^j \times l^j\right)}{\sum_{\forall j \in 1 \cdots |\text{SP}_i|}} f_i^j \leq d_i, \tag{6}$$

where d_i is the hop delay constraint for commodity i and l^j is the hop delay of path j. Once the flows on each path of a commodity are obtained, we can order or assign probability values to the paths based on the corresponding flows.

6. FAULT-TOLERANCE SUPPORT WITH MULTIPATH ROUTING

The errors that occur on the NoC links can be broadly classified into two categories: transient and permanent errors.

6.1. Resilience against transient errors

To recover from transient errors, error detection or correction schemes can be utilized in the on-chip network [11]. Forward error correcting codes such as Hamming codes can be used to correct single-bit errors at the receiving NI. However, the area-power overhead of the encoders decoders, and control wires for such error correcting schemes increases rapidly with the number of bit errors to be corrected. In practice, it is infeasible to apply forward error correcting codes to correct multi-bit errors [11]. To recover from such multi-bit errors, switch-to-switch (link-level) or end-to-end error detection and retransmission of data can be performed. This is applicable to normal data packets. However, control packets such as interrupts carry critical information that need to meet real-time requirements. Using retransmission mechanisms can have significant latency penalty that would be unacceptable to meet the real-time requirements of critical packets. Error resiliency for such critical packets can be achieved by sending multiple copies of the packets across one or more paths. At the receiving switch/NI, the error detection circuitry can check the packets for errors and can accept an error free packet. When sending multiple copies of a packet, it is important to achieve the required reliability level for packet delivery with minimum data replication. We formulate the mathematical models for the reliability constraints and consider them in the LP formulation presented in previous section, as follows.

Definition 3. Let the transient *Bit-Error Rate (BER)* encountered in crossing a path with maximum number of hops in the NoC be β_t. Let the bit-width of the link (also equal to the flit-width) be W.

The maximum probability of a single-bit error when a flit reaches the destination is given by

$$P(\text{Single-bit error in a flit}) = C_1^W \times \beta_t^1 \times (1 - \beta_t)^{W-1}. \tag{7}$$

That is, any one of the W bits can have an error, while the other bits should be error free.

We assume a single-bit error correcting Hamming code is used to recover from single-bit errors in the critical packets and packet duplication is used to recover from multi-bit errors. The probability of having two or more errors in a flit received at the receiving NI is given by

$$P(\geq 2 \text{ errors}) = \gamma_t = \sum_{k=2}^{W} C_k^W \times \beta_t^k \times (1 - \beta_t)^{W-k}. \tag{8}$$

We assume that the error rates on the multiple copies of a flit are statistically independent in nature, which is true for many transient noise sources such as soft errors (for those transient errors for which such statistical independence cannot be assumed, we can apply the method for recovering from permanent failures presented later in this section). When a flit is transmitted n_t times, the probability of having

two or more errors in all the flits is given by

$$\theta_t = \gamma_t^{n_t}. \tag{9}$$

As in earlier works [11, 22, 23], we assume that an undetected or uncorrected error causes the entire system to crash. The objective is to make sure that the packets received at the destination have a very low probability of undetected/uncorrected errors, ensuring the system operates for a predetermined *Mean Time To Failure (MTFF)* of few years. The acceptable residual flit error-rate, defined as the probability of one or more errors on a flit that can be undetected by the receiver is given by the following equation:

$$\mathrm{Err}_{\mathrm{res}} = \frac{T_{\mathrm{cycle}}}{(\mathrm{MTTF} \times N_c \times \mathrm{inj})}, \tag{10}$$

where T_{cycle} is the cycle time of the NoC, N_c is the number of cores in the system and inj is the average flit injection rate per core. As an example, for 500 MHz system with 12 cores, with an average injection rate of 0.1 flits/core and MTTF of 5 years, the $\mathrm{Err}_{\mathrm{res}}$ value is 1.07×10^{-17}. Each critical packet should be duplicated as many times as necessary to make the θ_t value to be greater than the $\mathrm{Err}_{\mathrm{res}}$ value, that is,

$$\theta_t = \gamma_t^{n_t} \geq \mathrm{Err}_{\mathrm{res}}, \quad \text{i.e., } n_t \geq \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_t)}. \tag{11}$$

The minimum number of times the critical packets should be replicated to satisfy the reliability constraints is given by

$$n_t = \left\lceil \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_t)} \right\rceil. \tag{12}$$

To consider the replication mechanism in the LP, the traffic rates of the critical commodities are multiplied by n_t and (2) is modified for such commodities as follows:

$$\sum_{\forall j \in 1 \cdots |\mathrm{SP}_i|} f_i^j = n_t \times r_i \quad \forall i, \text{critical}. \tag{13}$$

6.2. Resilience against permanent errors

To recover from permanent link failures, packets need to be sent across multiple nonintersecting paths. The nonintersecting nature of the paths makes sure that a link failure on one path does not affect the packets that are transmitted on the other paths. As in the transient error recovery case, the critical packets can be sent across multiple paths. For noncritical packets, we assume that hardware mechanisms such as presented in [20] exist to detect and inform the sender of a permanent link failure in a path. Then, the sender does not consider the faulty path for further routing and retransmits the lost flits across other nonintersecting paths. The probability of a path failure in the NoC is given by

$$P(\text{path failure}) = \gamma_p = \sum_{k=1}^{W} C_k^W \times \beta_p^k \times (1 - \beta_p)^{W-k}, \tag{14}$$

where β_p is the maximum permanent bit-error rate of any path in the NoC.

The maximum number of permanent path failures for each commodity (denoted by n_p) can be obtained similar to the derivation of n_t, and is given by

$$n_p = \left\lceil \frac{\ln(\mathrm{Err}_{\mathrm{res}})}{\ln(\gamma_p)} \right\rceil. \tag{15}$$

Let the total number of paths for a commodity i be denoted by $n_{\mathrm{tot},i}$. Once the number of possible path failures is obtained, we have to model the system such that for each commodity, any set of $(n_{\mathrm{tot},i} - n_p)$ paths should be able to support the traffic demands of the commodity. Thus, even when n_p paths fail, the set of other paths would be able to handle the traffic demands of the commodity and proper system operation would be ensured. We add a set of $n_{\mathrm{tot},i}!/(n_p! \times (n_{\mathrm{tot},i} - n_p)!)$ linear constraints in place of (2) for each commodity in the LP with each constraint representing the fact that the traffic on $(n_{\mathrm{tot},i} - n_p)$ paths can handle the traffic demands of the commodity. As an example, when $n_{\mathrm{tot},i}$ is 3 and n_p is 1 (which means that any 1 path can fail from the set of 3 paths) for a commodity i, we need to add the following 3 constraints:

$$\begin{aligned} f_i^1 + f_i^2 &\geq r_i, \\ f_i^2 + f_i^3 &\geq r_i, \\ f_i^1 + f_i^3 &\geq r_i. \end{aligned} \tag{16}$$

Thus, the paths of each commodity can support the failure of n_p paths for the commodity, provided more than n_p paths exist. When we introduce these additional linear constraints, the impact on the run time of the LP is small (for our experiments, we did not observe any noticeable delay in the run time). This is due to the fact that the number of paths available for each commodity is usually small (less than 4 or 5) and hence only few tens of additional constraints are introduced for each commodity. Note that we can modify the mapping procedures to ensure that each commodity has more than n_p paths available for data transfer. In cases where the mapping procedure cannot produce more than n_p paths for some commodities, we can introduce additional links between switches to get such additional paths for the commodity. Modifying the NoC mapping and topology design processes to achieve these effects is beyond the scope of this paper.

7. SIMULATION RESULTS

7.1. Area, power, and timing overhead

Even though the methodology presented in this paper is general, for illustrative purposes we assume that the component designs are based on the ×pipes NoC architecture. The backbone of the NoC is composed of switches, whose main function is to route packets from sources to destinations. Switches provide buffering resources to lower congestion and improve performance; in ×pipes, output buffering is chosen, that is, FIFOs are present on each output port. Switches also handle

flow control issues and resolve conflicts among packets when they overlap in requesting access to the same physical links.

An NI is needed to connect each IP core to the NoC. NIs convert transaction requests/responses into packets and vice versa. In ×pipes, two separate NIs are defined, an *initiator* and a *target* one, respectively associated to system masters and system slaves. A master/slave device will require an NI of each type to be attached to it. The interface among IP cores and NIs is point-to-point as defined by the OCP 2.0 [33] specification, guaranteeing maximum reusability. NI *Look-Up Tables (LUTs)* specify the path that packets will follow in the network to reach their destination (source routing).

The estimated power overhead, based on gate count and synthesis results for switches/NIs, to support the multipath routing scheme for the 4×3 mesh network considered earlier (in Table 1) is found to be 18.09 mW, which is around 5% of the base NoC power consumption. For the power estimation, without loss of generality, we assume that 8 bits are used for representing the source and destination addresses and 8-bit packet identifiers are utilized. The baseline architecture already supports the use of source address in the header flit. With the use of 4-flit packets, with 32-bit for each flit in the baseline architecture, the multipath scheme results in a 13% increase in packet size. The power overhead accounts for this increase in packet size and for the look-up tables and combinational logic associated with the multipath routing scheme. The numbers assume a 500 MHz operating frequency for the network, using UMC 0.13 μm technology library. The estimated area overhead (from gate and memory cell count) for the multipath routing scheme is low (less than 5% of the NoC component area). The maximum possible frequency estimate of the switch design with support for the multipath routing tables is above 500 MHz.

7.2. Case study: MPEG decoder

We assume that the tasks of the MPEG application are assigned to processor/memory cores and the best mapping (minimizing the average communication hop delay) onto a mesh network is obtained using the tool presented in [15]. The communication between the various cores and the resulting mapped NoC are presented in Figures 4, 5. The various paths obtained using the algorithms presented earlier, for some of the commodities, are presented in Table 2. Applying the LP procedure to split the traffic across the obtained paths results in 35% reduction in the bandwidth requirements for the application when compared to single-path routing (both dimension-ordered and minimum-path routing). The bandwidth savings translate to 35% reduction in the required NoC operating frequency. For 16-bit link data width, the multipath routing requires 300 MHz frequency for the NoC to support the application traffic, while the single-path routing schemes require a NoC frequency of 405 MHz. The NoC frequency reduction leads to 22.22% reduction in power consumption of the NoC for the multipath scheme compared to single-path schemes, after accounting for the power overhead of the multipath scheme. The average packet latencies incurred for the MPEG NoC for *dimension ordered (Dim)*, *minimum path (Min)* and our proposed *multipath (Multi)* strategy for the MPEG NoC is presented in Figure 6(a). The simulations are performed on a flit-accurate NoC simulator designed in C++. The multipath routing strategy results in reduced frequency requirements to achieve the same latency as the single-path schemes for a large part of the design space.

When compared to the multipath routing scheme with reorder buffers (10 packet buffers/receiver), the current scheme results in 28.25% reduction in network power consumption.

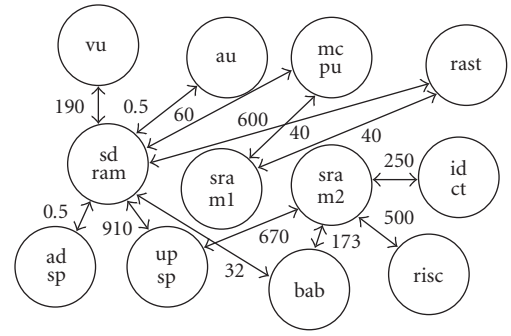


Figure 4: A MPEG decoder application.

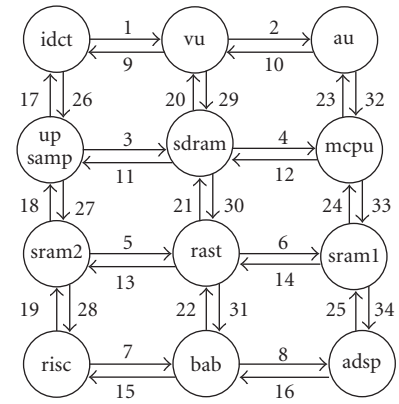


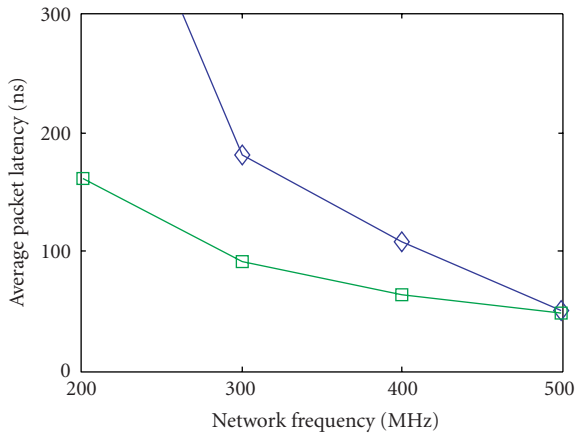
Figure 5: Mapped onto a mesh NoC. The edges of the mesh are numbered.

Table 2: Sample paths.

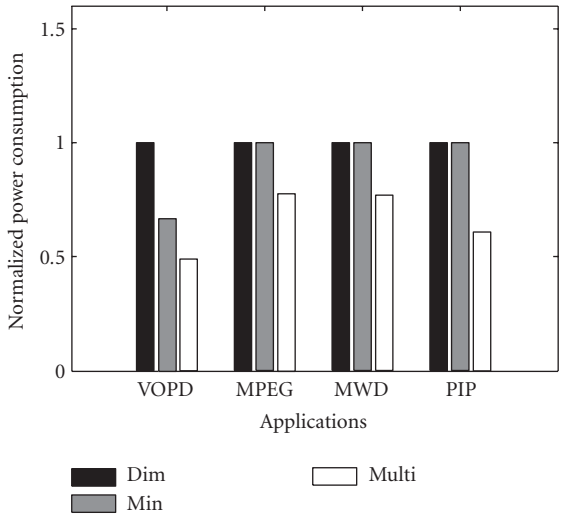
Comm.	Source	Dest.	Paths (edges traversed)
1	au	sdram	32-12, 10-29
2	mcpu	sdram	12, 23-10-29, 33-14-21
3	upsamp	sram2	27, 3-30-13
4	risc	sram2	19, 7-22-13

7.3. Comparisons with single-path routing

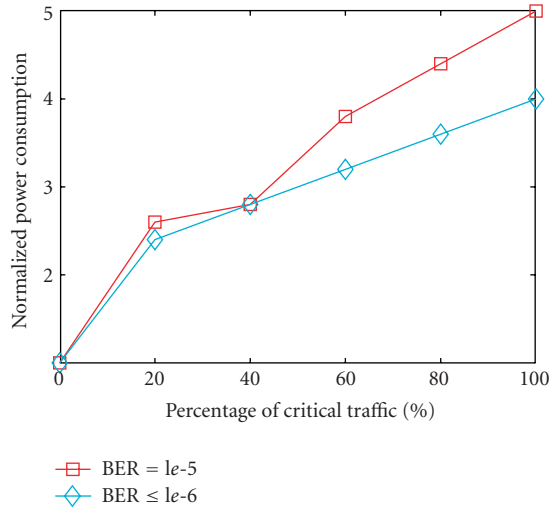
The network power consumption for the various routing schemes for different applications is presented in Figure 6(b).



(a) Average latency for MPEG NoC



(b) Routing effects on applications



(c) Effect of error resiliency

FIGURE 6: (a) Performance of routing schemes for MPEG NoC. (b), (c) Effect of routing and fault-tolerance on NoC power consumption.

The numbers are normalized with respect to the power consumption of dimension-ordered routing. We use several benchmark applications for comparison: *Video Object Plane Decoder (VOPD-mapped onto 12 cores)*, *MPEG decoder (MPEG-12 cores)*, *Multi-Window Display application (MWD-12 cores)* and *Picture-in-Picture (PIP-8 cores) application*. Description of the traffic characteristics of the applications is presented in [15]. Without loss of generality, we assume that the applications are mapped onto mesh topologies using the tool from [15], although the multipath routing strategy can be used for any topology. By using the proposed routing scheme, on average we obtain 33.5% and 27.52% power savings compared to the *dimension ordered* and *minimum path routing*, respectively. The total run time for applying our methodology (includes the run time for path selection algorithms for all commodities and for solving the resulting LP) is less than few minutes for all the benchmarks, when run on a 1 GHz Sun workstation.

7.4. Effect of fault-tolerance support

Adding support for resiliency against a single-path permanent failure for each commodity of the MPEG NoC resulted in a 2.33× increase in power consumption of the base NoC. Please note that the power overhead reported is for the worst-case scenario, where every communication flow experiences a single path failure. The amount of power overhead incurred in achieving fault-tolerance against temporary errors depends on the transient bit-error rate (β_t) of each link and the amount of data that is critical and needs replication. The effect of both factors on power consumption for the MPEG decoder NoC is presented in Figure 6(c). The power consumption numbers are normalized with respect to the base NoC power consumption (when no fault-tolerance support is provided). As the amount of critical traffic increases, the power overhead of packet replication is significant. Also, as the bit-error rate of the NoC increases (higher BER value in the figure, which imply a higher probability of bit-errors happening in the NoC), the amount of power overhead increases. We found that for all BER values lower than or equal to 1*e*-6, having a single duplicate for each packet was sufficient to provide the required MTTF of 5 years.

8. CONCLUSIONS

Routing packets across multiple paths can reduce network congestion and bottlenecks, which translates to reduced NoC frequency and power requirements or improved performance. Traditional multipath schemes require large packet reorder buffers at the receivers to provide in-order delivery. The reorder buffers have large area, power overhead, and deterministically sizing them is infeasible in practice. In this work, we have presented a multipath routing strategy that guarantees in-order packet delivery at the receiver. We introduced a methodology to find paths for the routing strategy and to split the application traffic across the paths to obtain a network operation with minimum power consumption. With technology scaling, reliable operation of on-chip

wires is also rapidly deteriorating and various transient and permanent errors can affect them. With the proposed multi-path routing strategy, we explored the use of spatial and temporal redundancy to tolerate transient as well as permanent errors occurring on the NoC links. Our method results in large NoC power savings for several SoC designs when compared to traditional single-path systems. In the future, we plan to extend the methods to implement source-based dynamic routing strategies that do not require reorder buffers at the receiver.

ACKNOWLEDGMENTS

This work is supported by the US National Science Foundation (NSF, contract CCR-0305718), the Swiss National Science Foundation (FNS, Grant 20021-109450/1), and Spanish Government Research Grant (TIN2005-5619). It is also supported by a Grant by STMicroelectronics for University of Bologna. The authors would also like to thank Alexandru Susu of EPFL for his useful comments on the work.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] D. Wingard, "MicroNetwork-based integration for SOCs," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 673–677, Las Vegas, Nev, USA, June 2001.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[4] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Rădulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '05)*, vol. 2, pp. 1182–1187, Munich, Germany, March 2005.

[5] S. Kumar, A. Jantsch, J.-P. Soininen, et al., "A network on chip architecture and design methodology," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '02)*, pp. 105–112, Pittsburgh, Pa, USA, April 2002.

[6] F. Karim, A. Nguyen, S. Dey, and R. Rao, "On-chip communication architecture for OC-768 network processors," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 678–683, Las Vegas, Nev, USA, June 2001.

[7] D. Bertozzi, A. Jalabert, S. Murali, et al., "NoC synthesis flow for customized domain specific multi-processor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[8] H. Jingcao and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 688–693, Munich, Germany, March 2003.

[9] F. Angiolini, P. Meloni, S. Carta, L. Benini, and L. Raffo, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *Proceedings of Design, Automation and Test in Europe (DATE '06)*, vol. 1, pp. 124–129, Munich, Germany, March 2006.

[10] V. Karamcheti and A. A. Chien, "Do faster routers imply faster communication?" in *Proceedings of the 1st International Workshop on Parallel Computer Routing and Communication (PCRCW '94)*, vol. 853 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, Seattle, Wash, USA, May 1994.

[11] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[12] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 2, pp. 896–901, Paris, France, February 2004.

[13] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS '05)*, pp. 75–80, Jersey City, NJ, USA, September 2005.

[14] S. Murali, L. Benini, and G. De Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the 12th Asia and South Pacific Design Automation Conference (ASP-DAC '05)*, pp. 27–32, Shanghai, China, January 2005.

[15] S. Murali and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," in *Proceedings of Design Automation Conference (DAC '04)*, pp. 914–919, San Diego, Calif, USA, June 2004.

[16] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 559–564, Anaheim, Calif, USA, June 2005.

[17] H. Jingcao and R. Marculescu, "DyAD - smart routing for networks-on-chip," in *Proceedings of the 41st Design Automation Conference (DAC '04)*, pp. 260–263, San Diego, Calif, USA, June 2004.

[18] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, Calif, USA, 2003.

[19] W. J. Dally, P. P. Carvey, and L. R. Dennison, "Architecture of the avici terabit switch/router," in *Proceedings of Hot-Interconnects VI*, pp. 41–50, Stanford, Calif, USA, August 1998.

[20] C. B. Stunkel, D. G. Shea, B. Abali, et al., "The SP2 communication subsystem," Tech. Rep., IBM, Yorktown Heights, NY, USA, August 1994.

[21] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali, "Adaptive source routing in multistage interconnection networks," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 258–267, Honolulu, Hawaii, USA, April 1996.

[22] R. Hegde and N. R. Shanbhag, "Towards achieving energy-efficiency in presence of deep submicron noise," *IEEE Transactions on VLSI Systems*, vol. 8, no. 4, pp. 379–391, 2000.

[23] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, 2005.

[24] R. Marculescu, "Networks-on-chip: the quest for on-chip fault-tolerant communication," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '03)*, pp. 8–12, Tampa, Fla, USA, February 2003.

[25] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '03)*, pp. 188–193, Newport Beach, Calif, USA, October 2003.

[26] F. Worm, P. Thiran, P. Ienne, and G. De Micheli, "An adaptive low-power transmission scheme for on-chip networks," in *Proceedings of the 15th International Symposium on System Synthesis (ISSS '02)*, pp. 92–100, Kyoto, Japan, October 2002.

[27] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI '04)*, pp. 46–51, Lafayette, La, USA, February 2004.

[28] S. Manolache, P. Eles, and Z. Peng, "Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 266–269, Anaheim, Calif, USA, June 2005.

[29] S. Murali, D. Atienza, L. Benini, and G. De Micheli, "A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip," in *Proceedings of the 43rd ACM/IEEE Design Automation Conference (DAC '06)*, pp. 845–848, San Francisco, Calif, USA, July 2006.

[30] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for NoC systems," in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '06)*, pp. 142–147, Seoul, Korea, October 2006.

[31] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, Mass, USA, 1990.

[32] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, NY, USA, 1994.

[33] http://www.ocpip.org/.