Application Specific Routing Algorithms for Networks on Chip

Maurizio Palesi, *Member*, *IEEE*, Rickard Holsmark,
Shashi Kumar, *Senior Member*, *IEEE*, and Vincenzo Catania

Abstract—In this paper, we present a methodology to develop efficient and deadlock-free routing algorithms for Network-on-Chip (NoC) platforms that are specialized for an application or a set of concurrent applications. The proposed methodology, called the Application-Specific Routing Algorithm (APSRA), exploits the application-specific information regarding pairs of cores that communicate and other pairs that never communicate in the NoC platform to maximize communication adaptivity and performance. The methodology also exploits the known information regarding concurrency/nonconcurrency of communication transactions among cores for the same purpose. The methodology does not require virtual channels to guarantee deadlock freedom. We demonstrate, through analysis of adaptivity and simulation-based evaluation of latency and throughput, that algorithms produced by the proposed methodology give significantly higher performance as compared to other deadlock-free algorithms for both homogeneous and heterogeneous 2D mesh topology NoC systems. Since the APSRA methodology is topology agnostic, the most appropriate general implementation of the routing function within the router is using a table. A table-based implementation of the router is costlier as compared to an algorithm-based implementation. We also propose a technique to compress the routing table in a mesh topology NoC to very small sizes with very little negative effect on routing adaptivity.

Index Terms—Networks on Chip, adaptive routing, deadlock-free routing, application specific, table-based router, routing table compression, router architecture.

1 Introduction

Single-chip-based embedded systems are becoming increasingly complex and heterogeneous. One of the most important key factors that characterize such Systems on Chip (SoCs) is the seamless integration of numerous Intellectual Property (IP) cores performing different functions and operating at different clock frequencies. In just the last few years, Network on Chip (NoC) has emerged as a dominant paradigm for the synthesis of multicore SoCs [1]. They are generally viewed as the ultimate solution for the design of modular and scalable communication architectures and provide inherent support for the integration of heterogeneous cores through the standardization of the network boundary. A large number of different NoC architectures have been proposed by different research groups [2], [3], [4], [5] based on this paradigm. The network topology and routing algorithm used in the underlying on-chip communication network are the two most important aspects that distinguish various proposed NoC architectures [6], [7]. In this paper, we deal with routing algorithms.

Many deadlock-free routing algorithms have been proposed in literature in the context of multiprocessing systems ([8], [9], [10], [11], and [12] represent just a short list). Whereas some of them have been applied as is in the NoC domain (e.g., Odd-Even has been used in [13]), others have inspired the design of new routing algorithms targeted for NoC architectures (e.g., the turn-prohibition algorithm in [14] has been built on the turn model [9]). In the general context of multiprocessing systems, an important issue in the design of a routing algorithm is to conjugate deadlock freedom with high adaptiveness. A highly adaptive routing algorithm has a potential of providing high performance (low latency, low packet drop, and high throughput), fault tolerance, and uniform utilization of network resources. Unfortunately, freedom from deadlock is often achieved at a high loss of adaptivity. On the other hand, generally, routing algorithms providing high adaptiveness guarantee freedom from deadlock by means of virtual channels (VCs).

In general, for arbitrary network topologies, a relatively large number of VCs could be required to provide deadlock freedom, high adaptivity, and shortest paths [6]. However, the use of VCs introduces some overhead in terms of both additional resources (e.g., buffers, replication of the routing logic, etc.) and mechanisms for their management (e.g., complex arbitration). To limit such overhead, some proposals try to minimize the number of VCs required to obtain a high degree of adaptivity [15].

The aforementioned problems are much more marked in the context of NoCs. In the NoC scenario, the use of VCs has an important impact on both complexity and power dissipation of the router as reported in the BONE Project [16] and in many other industrial experiences and scientific work [17], [18], [19], [20], [21], [22]. For these reasons, we believe that reducing as much as possible the number of VCs required to provide deadlock-free routing and, at the same time, guaranteeing a high degree of

- *M. Palesi and V. Catania are with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Università di Catania, V.le Andrea Doria, 95125 Catania, Italy. E-mail: {mpalesi, vcatania}@diit.unict.it.*
- *R. Holsmark and S. Kumar are with the Department of Electronics and Computer Engineering, School of Engineering, Jönköping University, 551-11 Jönköping, Sweden. E-mail: {Rickard.Holsmark, Shashi.Kumar}@jth.hj.se.*

Manuscript received 19 Feb. 2008; revised 3 June 2008; accepted 4 June 2008; published online 19 June 2008.
Recommended for acceptance by B. Parhami.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-02-0066.
Digital Object Identifier no. 10.1109/TPDS.2008.106.

adaptiveness must be considered one of the primary concerns in the NoC domain.

In this paper, we present a methodology to design highly adaptive routing algorithms for NoC architectures that do not require the use of any VCs. The basic idea behind the proposed approach can be summarized as follows: Traditionally, in the general-purpose domain, a routing algorithm is designed based solely on the network topology. As there is no information about the application that will be mapped on the system, the routing algorithm must conservatively guarantee any two network nodes to communicate. On the contrary, in the embedded system domain, the designer generally has an in-depth knowledge of the application (or the set of applications) that will be mapped on the system. If we consider the canonical design flow of an NoC-based embedded system [23], after the task mapping phase, we have the information about the set of pairs of cores that communicate and other pairs that never communicate. After the task scheduling phase of system development, we can also have information about the set of communication transactions that are concurrent and others that are nonconcurrent. We take advantage of these characteristics to "extract" adaptivity for free.

In this paper, we formalize the rationale behind this idea by extending Duato's theory [24], [25] to exploit information about an application's static and dynamic communication topology in order to develop efficient, highly adaptive, deadlock-free, and topology-agnostic routing algorithms. We call algorithms developed using this information as *Application-Specific Routing Algorithms* (APSRAs). Given a mapped and scheduled application (or a set of concurrent applications), we describe a methodology and algorithms to design an APSRA for it with the objective of maximizing adaptivity. We will show that the APSRA methodology is general, in the sense that it is not designed for a given/fixed network topology but can be used on any network topology. That is, routing algorithms generated by APSRA are topology agnostic.

We will demonstrate, through analysis of adaptivity and simulation-based evaluation of latency and throughput, that algorithms produced by this methodology give significantly higher performance as compared to other deadlock-free algorithms for both homogeneous and heterogeneous 2D mesh topology NoC systems. It is important to note that although the proposed methodology does not require the use of VCs as a mechanism to deal with deadlocks, it does not mean that it is in contrast with VC-based designs. The presence of VCs is definitely a design opportunity for global network performance optimization. In fact, the possibility to design highly adaptive routing algorithms in a transparent fashion with respect to the actual VC budget (if any) makes it possible to fully exploit the available VCs for optimization/managing purposes like QoS, prioritization, head-of-line blocking mitigation, etc.

The rest of the paper is organized as follows: In Section 2, we briefly describe some background concepts related to the paper and also discuss related research work. In Section 3, we define relevant terminology and give some results that form the theoretical basis for the APSRA methodology. In Section 4, we first introduce APSRA design methodology through a simple example and later discuss various algorithmic and implementation issues for it. In Section 5, we compare the performance (adaptivity, latency, and throughput) of routing algorithms designed using the APSRA methodology with general-purpose routing algorithms for both regular mesh and irregular mesh topology NoC architectures. The architectural implications of APSRA are discussed in Section 6. Finally, Section 7 presents some conclusions regarding the APSRA methodology and lists possibilities of future work.

2 Related Work

The overall performance of a network depends on several network properties such as topology, routing algorithm, flow control, and switching technique. In this paper, we mainly focus on routing algorithms. A routing algorithm determines the path selected by a packet to reach its destination.

Hu and Marculescu in [13] propose a routing scheme called *DyAD*, which combines the advantages of both deterministic and adaptive routing schemes. The router works in deterministic mode when the network is not congested and switches to adaptive mode when the network becomes congested.

In [26], Ye et al. present a contention-look-ahead on-chip routing scheme that is similar to [27]. It is nonminimal routing in the sense that based on the value of two delay penalty indices, the router chooses whether to send the packet toward a *profitable route* (minimal route) or a *misroute* (nonminimal route). In [28], Ascia et al. present a new selection strategy named *Neighbors-on-Path*, developed with the aim to choose the channel that will allow the packet to be routed to its destination along a path that is as free as possible of congested nodes.

Many routing algorithms for wormhole-switched networks have been proposed in the literature [9], [10], [11], [12]. Glass and Ni in [9] propose a turn model for designing wormhole routing algorithms for mesh and hypercube topology networks that are deadlock and livelock free. Prohibiting just enough turns to avoid the formation of any cycle produces routing algorithms that are deadlock free, livelock free, and highly adaptive. This model has been later utilized by Chiu in [12] to develop the *Odd-Even* adaptive routing algorithm for meshes without VCs. The model restricts the locations where some turns can be taken so that deadlock is avoided. In comparison with the turn model, the degree of routing adaptiveness provided by the model is greater even for different source-destination pairs. A nonminimal deadlock-free routing algorithm is described for an irregular mesh topology NoC with regions in [29]. This algorithm is biased in favor of some area of the network as compared to the other area.

Many other routing algorithms have been proposed in the literature, especially in the domain of high-performance computing based on a cluster of PCs. Most of them provide fault tolerance by the use of topology-agnostic routing, possibly combined with a reconfiguration process. Some examples, which use no VCs and can be adapted to be used in the NoC domain, are lturn [30], smart routing [31], FX [32], and segment-based routing [33].

3 Terminology and Definitions

In an embedded system scenario, the communication traffic between different cores of a system on chip is usually well characterized. In particular, after the task mapping phase of the NoC design flow, we have complete knowledge about the pairs of cores that communicate and other pairs that never communicate. This information can be incorporated in Duato's theory for the systematic design of deadlock-free routing algorithms for communication networks [24]. In this section, we define the concept of *Application-Specific Routing*.

3.1 Basic Definitions

Given a directed graph $G(V, E)$, where V is the set of vertices, and E is the set of edges, we indicate with $e_{ij} = (v_i, v_j)$ the directed arc from vertex v_i to vertex v_j. Given an edge $e \in E$, we indicate with $src(e)$ and $dst(e)$, respectively, the source and the destination vertices of the edge [e.g., $src(e_{ij}) = v_i$ and $dst(e_{ij}) = v_j$].

Definition 1. *A* Communication Graph $CG = G(T, C)$ *is a directed graph where each vertex t_i represents a task and each directed arc $c_{ij} = (t_i, t_j)$ represents the communication from t_i to t_j.*

Definition 2. *A* Topology Graph $TG = G(P, L)$ *is a directed graph where each vertex p_i represents a node of the network and each directed arc $l_{ij} = (p_i, p_j)$ represents a physical unidirectional channel (link) connecting node p_i to node p_j.*

Definition 3. *A* Mapping Function $M : T \rightarrow P$ *maps a task $t \in T$ on a node $p \in P$.*

Let $L_{in}(p)$ and $L_{out}(p)$ respectively be the set of input channels and output channels for node p. Mathematically

$$L_{in}(p) = \{l | l \in L \wedge dst(l) = p\},$$
$$L_{out}(p) = \{l | l \in L \wedge src(l) = p\}.$$

Definition 4. *A* Routing Function *for a node $p \in P$ is a function $R(p) : L_{in}(p) \times P \rightarrow \wp(L_{out}(p))$. $R(p)(l, q)$ gives the set of output channels of node p that can be used to send a message received from the input channel l and whose destination is $q \in P$. We assume that $R(p)(l, q) = \emptyset$ if q is not reachable from p.*

The $\wp$ indicates a power set. We indicate with R the set of all routing functions:

$$R = \{R(p) : p \in P\}.$$

To make the paper self-contained, we briefly report two fundamental definitions and one theorem given by Duato [24], [25] that are extensively used in the rest of the paper.

Definition 5. *Given a topology graph $TG(P, L)$ and a routing function R, there is a* direct dependency *from $l_i \in L$ to $l_j \in L$ if l_j can be used immediately after l_i by messages destined to some node $p \in P$.*

Definition 6. *A* Channel Dependency Graph $CDG(L, D)$ *for a topology graph TG and a routing function R is a directed graph. The vertices of CDG are the channels of TG. The arcs of CDG are the pairs of channels (l_i, l_j) such that there is a direct dependency from l_i to l_j.*

The following theorem due to Duato is a straightforward extension of Dally and Seitz's theorem [34] for adaptive routing functions.

Theorem 1 (Duato's theorem [24]). *A routing function R for a topology graph TG is deadlock free if there are no cycles in its channel dependency graph CDG.*

3.2 Incorporating Communication Topology Information

As we can observe, the above definitions and theorem do not make any reference to the communication traffic because they implicitly assume that all the nodes of the network can communicate with each other. As stated at the beginning of the section, in an embedded system scenario, the designer often knows which nodes of the network communicate and which do not. This information can be captured by a communication graph CG. In the rest of this section, we extend Duato's theory by incorporating this communication topology information.

Definition 7. *Given a communication graph $CG(T, C)$, a topology graph $TG(P, L)$, a mapping function M, and a routing function R, there is an* application-specific direct dependency *from $l_i \in L$ to $l_j \in L$ iff*

$$dst(l_i) = src(l_j), \tag{1}$$

$$\exists\, c \in C : l_j \in R(dst(l_i))(l_i, M(dst(c))). \tag{2}$$

Condition (1) states that there exists a possibility for a message to use l_j immediately after l_i. Condition (2) states that there exists a communication that will actually use l_j immediately after l_i.

Definition 8. *An* Application-Specific Channel Dependency Graph $ASCDG(L, D)$ *for a given CG, a topology graph TG, and a routing function R is a directed graph. The vertices of $ASCDG$ are the channels of TG. The arcs of $ASCDG$ are the pairs of channels (l_i, l_j) such that there is an application-specific direct dependency from l_i to l_j.*

In this paper, we assume minimal routing while constructing $ASCDG$.

Theorem 2. *A routing function R for a topology graph TG and for a communication graph CG is deadlock free if there are no cycles in its application-specific channel dependency graph $ASCDG$.*

Proof. As the $ASCDG$ is acyclic, it is possible to establish an order between the channels of L. Suppose that there is a deadlocked configuration for R. Let l_i be a channel of L with a nonempty queue such that there are no channels less than l_i with a nonempty queue. The following two cases are possible:

Case 1. l_i is minimal. In this case, as proved in [24], l_i is a sink, i.e., a channel such that all the flits that enter on it reach the destination in a single hop. Then, the flit at the queue head can reach its destination in a single hop, and there is no deadlock.

Case 2. l_i is not a minimal. For each $l_j \in L$ such that $l_i > l_j$, there are no flits in the queue for channel l_j. Thus, the flit at the head of the queue for l_i is not blocked, regardless of whether it is a header or a data flit, and there is no deadlock. □

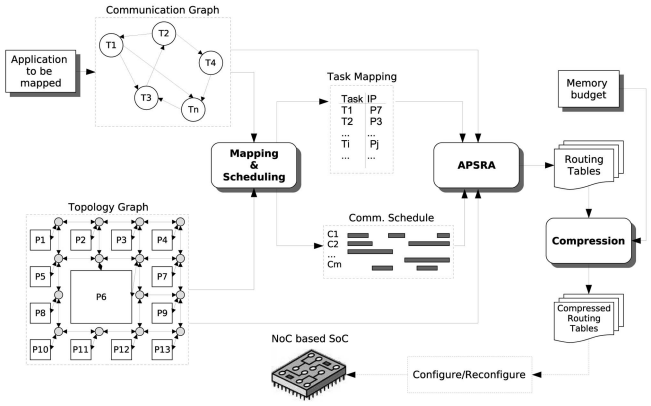


Fig. 1. Overview of the APSRA design methodology.

3.3 Routing Adaptivity

Many metrics have been used for estimating, evaluating, and comparing the performance of various routing algorithms. These metrics include different versions of latency (spread, minimum, maximum, average, and expected), various versions of throughput, jitter in latency, jitter in throughput, etc. Adaptivity is another measure used in this context and refers to ability of the algorithm to offer alternative paths for a packet. High adaptivity increases the chances that packets may avoid hot spots or faulty components and reduces the chances that packets are continuously blocked. Higher adaptivity facilitates network management and generally leads to higher performance for the applications.

In this paper, we use the definition of adaptivity given by Glass and Ni in [9], which is also used by Chiu in [12] and by many other researchers. We define adaptivity, $\alpha(c)$, (sometimes also referred as degree of adaptiveness) for a communication c as the ratio of the number of allowed minimal paths to the total number of minimal paths between the source node and the destination node. The average adaptivity α is the average of the degree of adaptiveness for all the communications.

4 APSRA Design Methodology

The APSRA design methodology is depicted in Fig. 1. It gets as inputs the application modeled as a task graph (or as set of concurrent task graphs) and the network topology modeled as a topology graph. We assume that the tasks in the application have already been mapped and scheduled on the available NoC resources (such steps are not new to the CAD community, as they have been addressed in the area of hardware/software codesign and IP reuse [35]). Using this information, APSRA generates a set of routing tables (one for each router of the NoC) that not only guarantee both the reachability and the deadlock freeness of communication among tasks but also try to maximize routing adaptivity. A compression technique can be used to compress the generated routing tables. Finally, the compressed routing tables are uploaded to the physical chip (NoC configuration). Of course, all the aforementioned steps are carried out offline.

Let us start with a brief overview of the APSRA methodology by means of an example.

4.1 APSRA by Example

For the sake of an example, let us consider the communication graph and the topology graph depicted in Figs. 2a and 2b, respectively. Although for this example, the topology is mesh based, the approach is general and can be applied to any network topology without any modification. As the mapping function, let us consider $M(T_i) = P_i$, $i = 1, 2, \ldots, 6$. The channel dependency graph (CDG) [25] for a minimal fully adaptive routing algorithm is shown in Fig. 2c. Since it contains six cycles, Duato's theorem [25] cannot assure deadlock freeness of the minimal fully adaptive routing for this topology. The $ASCDG$ is shown in Fig. 2d. It has been obtained from the CDG by removing all the false dependencies as follows: For the sake of an example, let us consider the dependency $l_{1,2} \rightarrow l_{2,3}$. Such a dependency is present in the CDG, but it is not present in the $ASCDG$. In fact, channels $l_{1,2}$ and $l_{2,3}$ can be used in sequence only for the communications $T_1 \rightarrow T_3$, $T_1 \rightarrow T_6$, and $T_4 \rightarrow T_3$, which are not present in the CG. Although the number of cycles is reduced to two for the $ASCDG$, also in this case, we cannot assure the deadlock freeness. At any rate, we can simply break the cycles as follows: The application-specific channel dependency $l_{4,1} \rightarrow l_{1,2}$ is due to the communication $T4 \rightarrow T2$. Such communication can be realized by both paths $P4 \rightarrow P5 \rightarrow P2$ and $P4 \rightarrow P1 \rightarrow P2$. If the routing function is restricted in such a way that the latter path is prohibited, the application-specific channel dependency $l_{4,1} \rightarrow l_{1,2}$ does not exist any longer. In a similar way, it is possible to break the second cycle, removing, for instance, the dependency $l_{1,4} \rightarrow l_{4,5}$ due to communication $T1 \rightarrow T5$.

However, this restriction reduces the degree of adaptiveness of the routing. Now, suppose that we have some

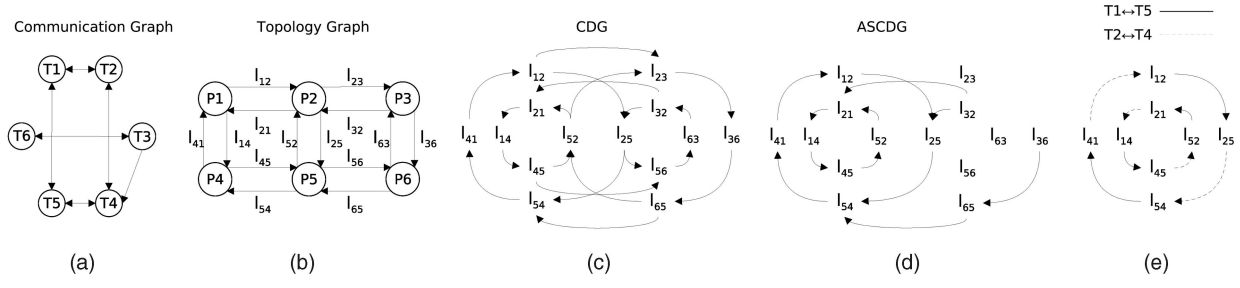


Fig. 2. Comparison of cyclic dependencies without and with the APSRA methodology.

```
1  APSRA(in: CG, TG, M; out: RT)
2  {
3     R ← MinimalFullyAdaptiveRouting(CG,TG,M);
4     BuildASCDG(CG,TG,M,R,ASCDG);
5     GetCycles(ASCDG,C);
6     RemoveCycles(C,ASCDG,CG,TG,M,R,success);
7     if (success)
8        ExtractRoutingTables(R,RT);
9     else
10       RT ← ∅;
11 }
```

Fig. 3. APSRA main algorithm.

knowledge about communication concurrency and suppose that communication $T1 \rightarrow T5$ and communication $T2 \rightarrow T4$ do not overlap in time. Fig. 2e highlights the dependencies due to such communications. Since these communications are not concurrent, the associated dependencies are not concurrently active too. The result is that the two cycles are actually false cycles. In conclusion, for this latter case, a minimal fully adaptive routing is deadlock free.

4.2 Main Algorithm

The main algorithm that implements the APSRA methodology is shown in Fig. 3. It gets as inputs a communication graph CG, a topology graph TG, and a mapping function M and returns the set of routing tables. The algorithm starts by initializing R with a minimal fully adaptive routing and then calls the procedure `BuildASCDG`, which builds the $ASCDG$. The procedure `GetCycles` returns the set C of cycles in the $ASCDG$. `RemoveCycles` tries to remove all the cycles C from $ASCDG$ with the objective of minimizing the loss in adaptivity and with the constraint to guarantee the reachability between all communicating pairs (cf. Section 4.3). If it succeeds, it returns *true*, and the procedure `ExtractRoutingTables` is used to extract routing tables from R (cf. Section 4.4); otherwise, an empty set is returned.

The construction of the $ASCDG$ (procedure `BuildASCDG`) involves the annotation of each minimum path between any source/destination pair, as defined in the communication graph. The basic assumption is that we start from a minimal fully adaptive routing algorithm. If we consider a mesh-based topology, the complexity to annotate all the minimal paths for a given source-destination pair is $O(2^n)$, where n is the dimension of the quadrant containing the source and destination nodes. It means that as the NoC size increases, the approach could become infeasible if some nodes located far from each other need to communicate. It should be pointed out, however, that this is the worst case condition, and in any case, it can be managed efficiently considering the following. First, any topological mapping algorithm (like [36], [37], and [38]) tries to map the most frequent and most critical communications in such a way as to minimize the physical distance between the source and destination nodes. This leads to mapped architectures that seem to mimic a kind of small-world phenomenon [39] in which there are many communications following short paths and only a few communications that require long paths. Second, the long-distance communications, which determine the complexity of the `BuildASCDG` procedure, can be treated in a more practical way. That is, for these long-distance communications, one can consider a subset of all the minimal paths. To limit the number of minimal paths to be annotated, an idea could be to fix a budget of minimal paths that can be used for any communication. In this way, the complexity of the `BuildASCDG` procedure can be tuned by simply modifying the budget, which can be considered as a user-defined parameter.

The `RemoveCycles` procedure tries to remove all the cycles C of the $ASCDG$. As will be discussed in Section 4.3, the order in which the cycles in $ASCDG$ get treated determines both the overall adaptivity of the generated routing algorithm and the routability for all the communications. More precisely, with regard to the second point, certain cycle removal sequences might make some communications unroutable. In our implementation, we used a backtracking mechanism in which removing sequences are generated by performing a depth-first search of the solution space.

It should be pointed out that all the steps listed in Fig. 3 are carried out offline. In addition, some of these steps can be modified and customized on a case-by-case basis. For instance, to cope with the out-of-order packet delivery problem, which characterizes adaptive routing algorithms, the simple reordering mechanism at network reconvergent nodes proposed in [40] can be used. In this case, it is sufficient that R, in the procedure shown in Fig. 3, is restricted by removing all the *intersecting* paths for each source/destination pair.

4.3 Cutting Edge with Minimum Loss

As discussed above, the procedure `RemoveCycles` tries to remove all the cycles from $ASCDG$ with the objective of minimizing the loss in adaptivity and with the constraint to guarantee the reachability between all communicating pairs. Before we start presenting the heuristic used to select the application-specific channel dependency to be removed to break a cycle, some definitions should be given.

Given a communication $c \in C$ we denote with $\Phi(c)$ the set of all permissible minimal paths from node $M(src(c))$ to node $M(dst(c))$. We indicate with $\phi_i(c)$ the ith path of $\Phi(c)$. A *Path* from a node p_s to a node p_d is a succession of channels $\{l_1, l_2, \ldots, l_n\}$, $l_i \in L$, such that $dst(l_i) = src(l_{i+1})$, $i = 1, 2, \ldots, n-1$, $p_s = src(l_1)$, and $p_d = dst(l_n)$.

Definition 9. *For an edge d of the $ASCDG$ the* Pass-through set $A(d)$ *is the set of pairs (c, j), where $c \in C$ is a communication whose jth path contains both channels associated to d. Formally*

$$A(d) = \{(c, j) |\ c \in C,\ j \in \mathbb{N} :\ src(d) \in \phi_j(c) \wedge dst(d) \in \phi_j(c)\}.$$

For an edge d of the $ASCDG$, we indicate as $A(d)_{|c}$ the restriction of $A(d)$ for communication $c \in C$, that is

$$A(d)_{|c} = \{(c', \cdot) \in A(d) : c' = c\}.$$

Theorem 3. *Given an $ASCDG(L, D)$ and $d = (l_i, l_j) \in D$, then $A(d) \neq \emptyset$.*

Proof. $\Rightarrow d = (l_i, l_j) \in D$; then, $\exists\, c \in C$ such that $l_j \in R(dst(l_i))(l_i, c)$, that is, there exists a communication c

that has a path that contains both l_i and l_j. This path belongs to $\Phi(c)$, and suppose it is the jth path of $\Phi(c)$, named $\phi_j(c)$. Then, the pair (c, j) belongs to $A(d)$ because $src(d) = l_i \in \phi_j(c)$, and $dst(d) = l_j \in \phi_j(c)$.

$\Leftarrow$ Let $a = (c, j) \in A(d)$; then, $\exists \phi_j(c) \in \Phi(c)$ that contains both $src(d) = l_i$ and $dst(d) = l_j$. Condition (1) is satisfied by construction because $d \in D$. The existence of the path $\phi_j(c)$ states that a communication c travelling on l_i can immediately use l_j. This means that the routing function at node $dst(l_i)$ allows this turn. Therefore, $l_i \in R(dst(l_i))(l_i, M(dst(c)))$, and (2) is satisfied too. □

Therefore, let $D_c = \{d_1, d_2, \ldots, d_n\} \subseteq D$ be a cycle in the $ASCDG(L, D)$. To break the cycle, a dependency d_i must be removed. If d_i is removed, based on Theorem 3, $A(d_i) = \emptyset$. To make $A(d_i) = \emptyset$, the set of admissible paths for some communications has to be restricted. This, however, has an impact on the degree of adaptiveness of the routing function. The heuristic has to select the dependency d_i to be removed in such a way that it minimizes the impact on the degree of adaptiveness.

Let α be the current degree of adaptiveness and α_d be the degree of adaptiveness when we remove a dependency $d \in D_c$. The objective is to minimize the difference $\alpha - \alpha_d$ or, equivalently, maximize α_d:

$$\max_{d \in D_c} \alpha_d = \max_{d \in D_c} \frac{1}{|C|} \sum_{c \in C} \frac{|\Phi_d(c)|}{TMP(c)}, \tag{3}$$

where with $\Phi_d(c)$ we indicated the set of paths for communication c when the dependency d is removed, that is

$$\Phi_d(c) = \Phi(c) \setminus \Big\{ \phi_j(c) |\ (c, j) \in A(d)_{|c} \Big\},$$

we have

$$\begin{aligned} |\Phi_d(c)| &= |\Phi(c)| - \Big| \Big\{ \phi_j(c) |\ (c, j) \in A(d)_{|c} \Big\} \Big| \\ &= |\Phi(c)| - \Big| A(d)_{|c} \Big|. \end{aligned}$$

Substituting into (3), we have

$$\max_{d \in D_c} \frac{1}{|C|} \left(\sum_{c \in C} \frac{|\Phi(c)|}{TMP(c)} - \sum_{c \in C} \frac{\Big| A(d)_{|c} \Big|}{TMP(c)} \right),$$

which is equivalent to

$$\min_{d \in D_c} \sum_{c \in C} \frac{|A(d)_{|c}|}{TMP(c)} = \min_{d \in D_c} \sum_{(c, \cdot) \in A(d)} \frac{1}{TMP(c)}.$$

In short, the heuristic states that to minimize the impact on adaptiveness, we have to select a dependency $d \in D_c$ to be removed that satisfies the following reachability constraint:

$$\bigwedge_{c \in C(d)} |\Phi(c)| > \Big| A(d)_{|c} \Big|, \tag{4}$$

and minimize the quantity:

$$\sum_{(c, \cdot) \in A(d)} \frac{1}{TMP(c)}, \tag{5}$$

where with $C(d)$ we indicate the set of communications having at least a path that contains both channels associated to d. Inequality (4) ensures that all the communications that use the links $src(d)$ followed by $dst(d)$ will have alternative paths after d is removed. The removal of a dependency d impacts on $\Phi(c)$ as follows:

$$\forall (c, j) \in A(d) \Rightarrow \Phi(c) = \Phi(c) \setminus \big\{ \phi_j(c) \big\}.$$

It is easy to show that our heuristic results in an optimum adaptivity when there is a single cycle in $ASCDG$. Optimality is not guaranteed in the case of multiple cycles. For a globally optimal solution, we need to consider all the cycles simultaneously.

Restricting the routing functions in various network nodes may also affect the reachability of certain communications. The order in which the cycles in $ASCDG$ get treated may finally decide if the constraint (4) can be met for all cycles or not. This implies that if we look at cycles in one order only, then we may not get a routing path for some communications. In fact, in the worst case, we may have to exhaustively consider all possible combinations of dependencies, one from each cycle in $ASCDG$, to be removed to find a feasible minimal routing for all communicating pairs.

4.4 Routing Tables

For each node $p \in P$ and for each input channel $l \in L_{in}(p)$, there is a routing table $RT(p, l)$ in which each entry consists of 1) a *destination address* $d \in P$ and 2) a set of output channels $O \in \wp(L_{out}(p))$ that can be used to forward a message received from channel l and destined to node d. Formally,

$$RT(p, l) = \{(d, O) |\ d \in P,\ O = R(p)(l, d) \wedge O \neq \emptyset\}.$$

The routing table of a node $p \in P$ is the union of routing tables of each input channel of p:

$$RT(p) = \bigcup_{l \in L_{in}(p)} RT(p, l).$$

Although in general, a table-based implementation of the routing function is more costly in terms of silicon area as compared to a custom logic implementation, it can be a blessing. In fact, the table-based implementation allows configurability (and even dynamic reconfigurability [41], [42]) of the routing algorithm to allow modifications in communication requirements in the running applications. However, the routing table size can be drastically reduced, as we will see in Section 6.1.

4.5 Exploiting Communication Concurrency

After the task mapping and scheduling step of a system-level design, it is possible to know whether two communications can be concurrent or not. The APSRA design methodology can be extended to exploit the communication concurrency information [43]. The basic idea is to apply the APSRA methodology to each *communication scenario*[1] (CS) obtaining a set of routing

1. A communication scenario is a pair (CG, Δ), where CG is a communication graph, and Δ is a time interval.

tables (one for each CS). The theory presented in [41] and [42] can be used to switch from one routing table to another (i.e., by router reconfiguration) when the communication scenario changes. For each communication scenario CS, the APSRA procedure is invoked and acts on the communication graph associated to CS.

The process of defining time intervals or communication scenarios is the responsibility of the task scheduling phase of NoC specialization. The task scheduler can either use the CS period as a constraint or produce communication scenario periods as output while optimizing some objective function. The granularity of the interval will affect routing adaptivity, as well as routing performance. Large intervals will effectively increase the communication density, leading to a decrease in adaptivity (cf. Section 5.1), but will have lower reconfiguration overheads. On the other hand, routing algorithms generated by APSRA will have very high adaptivity for very small intervals. The drawback is that the performance improvement due to this could be lost due to the heavy overheads of frequent reconfiguration. Deciding optimal communication scenario intervals is an interesting future research problem.

4.6 Some Notes about VC-Based Designs

Traditionally, highly adaptive routing algorithms guarantee freedom from deadlock by means of VCs. If the underlying routing infrastructure does not allow VC transitions [44] (i.e., the possibility for a packet traveling in a virtual layer to be switched onto another virtual layer), it needs to implement different routing algorithms at each VC level. This, however, has a negative impact on both router cost and complexity since each VC requires its own routing logic (e.g., routing table). On the other hand, if VC transitions are allowed, it is possible to use the same routing algorithm but perform a VC transition every time the packet will cross a routing restriction [15]. However, both solutions require a relatively large number of VCs to obtain maximum adaptivity, shortest path routing, and deadlock freedom [15].

Aoyama and Chien [45] showed about a 30 percent penalty in router speed for each extra VC. This cost comes from the additional delay introduced into the critical path due to the addition of arbitration and multiplexing logic and due to the slowdown of the crossbar controller in implementing alternate routing algorithms. In [46], the authors found that the benefits of an increased number of VCs may not compensate for the loss of performance due to the slower speed of the router.

We believe that due to the continuously increasing complexity of applications that will be mapped in NoC-based systems, along with the demand of guaranteed services that such applications require, VC resources should be exploited for optimization purposes instead of deadlock avoidance problems. The methodology we propose does not require the presence of VCs. As will be shown in the next section, although we do not use VCs to tackle the problem of deadlock, it is still possible to obtain highly adaptive minimal deadlock-free routing algorithms by exploiting information about an application that comes for free. It will allow available VC resources to be exploited for other important issues such as QoS, traffic prioritization, network virtualization, etc., [22], [47], [48].

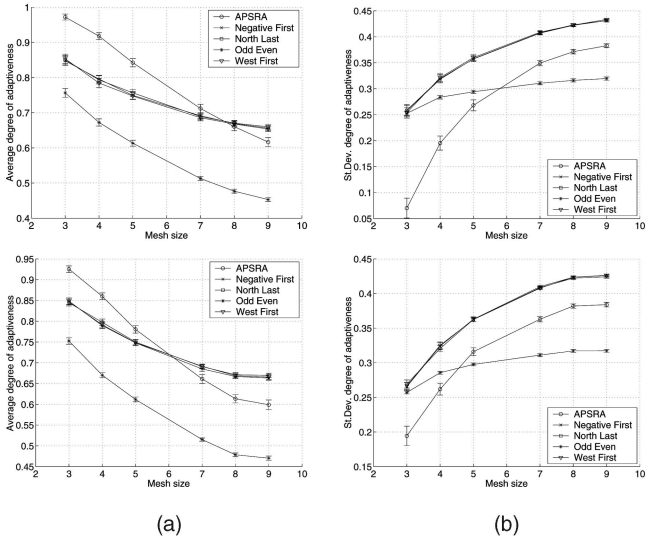


Fig. 4. Degree of adaptiveness (a) average and (b) standard deviation for different NoC sizes and for randomly generated communication graphs with (top) $\rho = 2$ and (bottom) $\rho = 4$.

5 Evaluation of APSRA

In this section, we analyze the performance of APSRA in terms of both adaptivity and simulation-based evaluation of latency and throughput for both homogeneous and non-homogeneous mesh NoCs with regions.

5.1 Adaptivity Analysis

We evaluate the adaptiveness provided by APSRA using three different communication scenarios: two synthetic and one that models a real multimedia application. In both synthetic communication graphs, the number of nodes (tasks) is fixed, whereas the number of edges (communications) is a parameter that characterize the communication scenario. We define the *communication density* ρ as the ratio between the number of communications and the number of tasks. The synthetic communication graphs are generated randomly (but, in any case, with the satisfaction of a given communication density) based on two different assumptions. In the first synthetic communication graph, each task can communicate with every other task with equal probability. In the second one, tasks communicate with a probability depending on the distance of the nodes where they are mapped on. More precisely, we define the *one-hop probability ohp* as the communication probability between two tasks mapped on two nodes distant by one hop from each other. The communication probability of two tasks mapped on two nodes distant by $h \geq 2$ hops is computed as $CP(h) = (1 - \sum_{i=1}^{h-1} CP(i))/2$. The mapping function is defined as $M(t_i) = p_{i \,\%\, |P|}$, where $|P|$ is the number of network nodes, and % is the module operator.

We compare APSRA with adaptive routing algorithms based on the *turn model* [9] and with the *Odd-Even* turn model [12]. Fig. 4a shows the average degree of adaptiveness for different NoC sizes and for $\rho = 2$ (top) and $\rho = 4$ (bottom). Each point of the graph has been obtained by evaluating 100 random communication graphs and reporting the mean value and the 90 percent confidence interval. The algorithms based on the turn model outperform ASPRA in degree of adaptiveness for large mesh

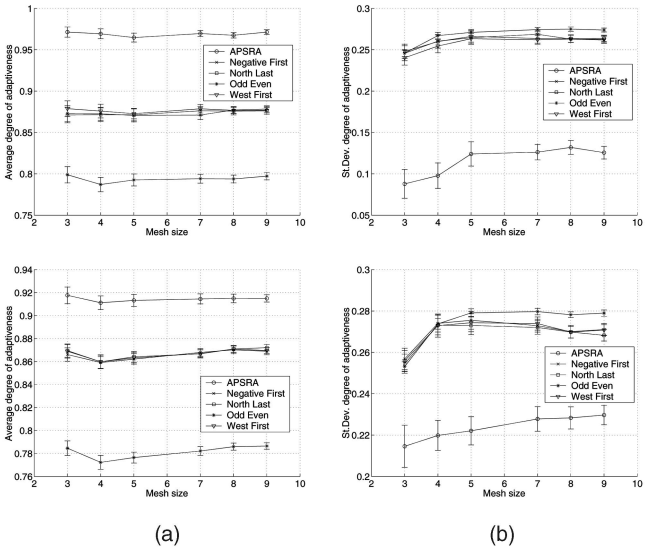


Fig. 5. Degree of adaptiveness (a) average and (b) standard deviation for different NoC sizes and for randomly generated communication graphs with (top) $\rho = 2$ and (bottom) $\rho = 4$ and $ohp = 0.4$.

sizes. Unfortunately, the degree of adaptiveness provided by the turn model is highly uneven [12]. This is because at least half of the source-destination pairs are restricted to having only one minimal path, while full adaptiveness is provided for the rest of the pairs. This is confirmed by the high standard deviation values these algorithms exhibit (Fig. 4b). On the other hand, Odd-Even is the worst one in terms of average degree of adaptiveness, but it is more even for different source-destination pairs. APSRA outperforms the other algorithms for small NoC sizes, but its performance decreases very fast as NoC size increases and communication density increases. In fact, for a given network size, increasing the communication density is the same as moving from the application-specific domain to the general-purpose domain, where in the extreme case, each node communicates with each other node of the network and where the turn model gives an upper bound on the maximum reachable degree of adaptiveness [9].

At any rate, the uniform random traffic scenario is not very representative for an NoC system [49]. Usually, in fact, the cores that communicate most are mapped close to each other [36], [37], [38]. The second traffic scenario models this behavior. Fig. 5 shows results obtained for $ohp = 0.4$. In this case, APSRA outperforms the other algorithms in terms of both adaptiveness (Fig. 5a) and standard deviation (Fig. 5b). Quantitatively, APSRA provides a very high level of adaptivity on the average, over 10 percent and 18 percent for the turn-model-based algorithms and Odd-Even for $\rho = 2$. And it provides 7 percent and 15 percent higher adaptivity over turn-model-based algorithms and Odd-Even for $\rho = 4$. Moreover, the degree of routing adaptiveness provided by APSRA is more even for different source-destination pairs.

Finally, we evaluate the performance improvement of APSRA when some information about communication concurrency is available. To do this, we randomly generate a communication graph as described above and use a *concurrency probability factor* χ to model the average number

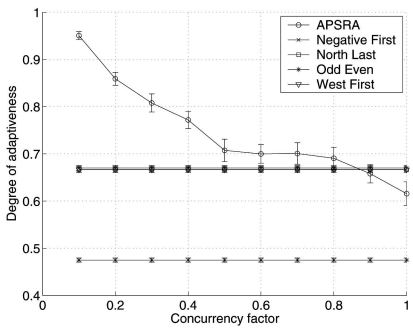


Fig. 6. Degree of adaptiveness versus communication concurrency probability for an 8 × 8 mesh and randomly generated communication graphs with $\rho = 2$.

of concurrent communications in an interval. That is, if $\chi = 1$, it means that all communications are concurrent, whereas $\chi = 0$ means that no communications overlap in time. Fig. 6 shows the degree of adaptiveness versus communication concurrency probability for an 8 × 8 NoC and randomly generated communication graphs with a communication density of $\rho = 2$. As expected, as the concurrency probability decreases, adaptiveness improves. For $\chi = 0.4$, for example, the degree of adaptiveness provided by APSRA is over 10 percent better than that by the turn models and over 30 percent better than that by Odd-Even. For higher values of χ, the exploitation of communication concurrency information does not result in any significant gain in adaptiveness over the turn-model-based routing algorithms.

5.2 Simulation-Based Evaluation

Now, we evaluate performance of APSRA using *noxim* [50], a flit-accurate simulator developed in SystemC. As performance metrics, we choose *throughput* and *delay*.

We compare APSRA with both a deterministic routing algorithm (XY) and an adaptive routing algorithm (Odd-Even). We choose Odd-Even because it has been proven to exhibit the best performance among different traffic scenarios [12]. It is also the most cited adaptive routing algorithms proposed for mesh networks without using VCs.

Comparison is carried out on both synthetic and real traffic scenarios. The evaluations were made on an 8 × 8 network using wormhole switching with a packet size randomly distributed between 2 and 16 flits and routers with input buffer size of 2 flits. The maximum bandwidth of each link is set to 1 flit per cycle. We use the source packet injection rate (*pir*) as the load parameter with Poisson packet injection distribution. For each load value, latency values are averaged over 60,000 packet arrivals after a warm-up session of 30,000 arrived packets. The 95 percent confidence intervals are mostly within 2 percent of the means. For adaptive routing algorithms, we use two different selection policies: *random* and *buffer level*. If multiple output ports are available for a header flit, a random selection policy selects the output randomly, whereas the buffer-level policy selects the output whose connected input port has the minimum buffer occupied.

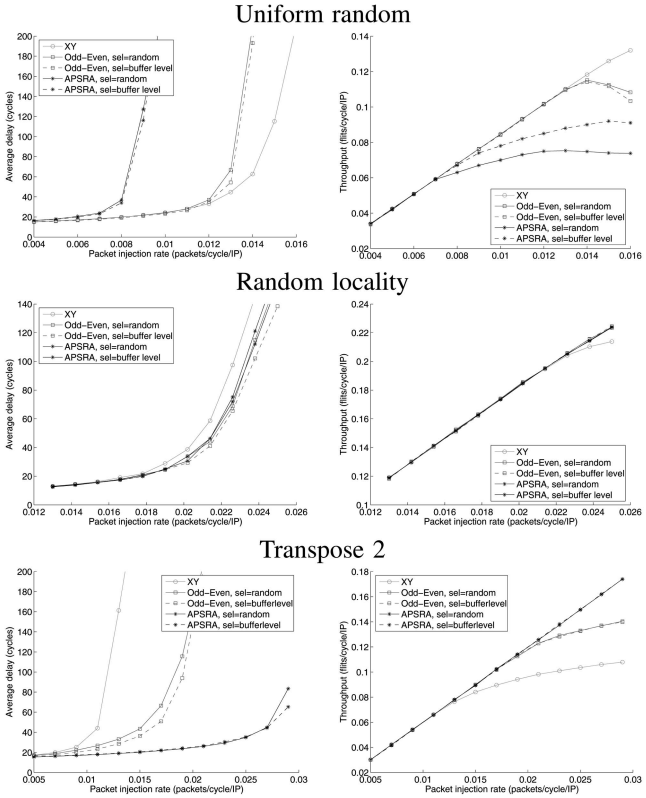


Fig. 7. Variation in (left) delay and (right) throughput under (top) uniform random traffic, (middle) random locality traffic with $\rho = 2$ and $ohp = 0.4$, and (bottom) *Transpose 2* traffic.

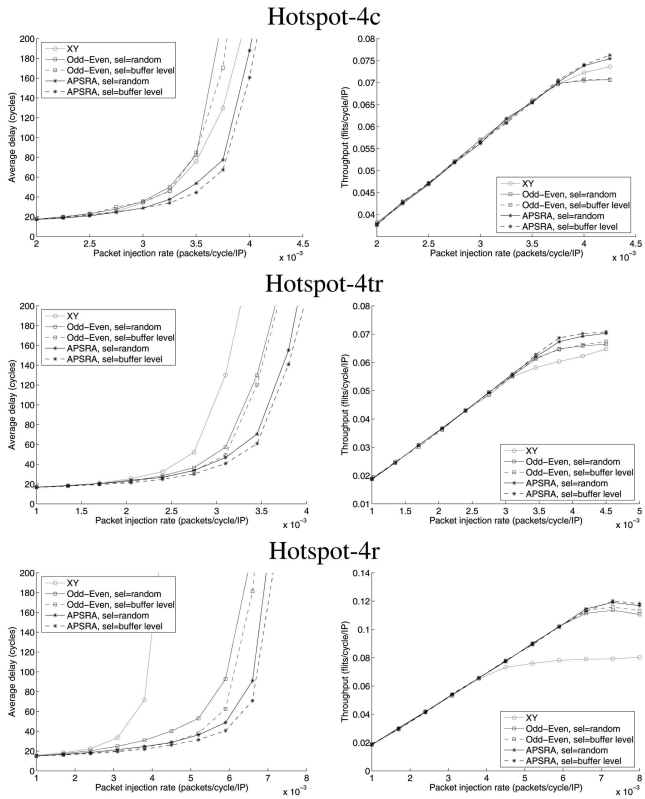


Fig. 8. Variation in (left) delay and (right) throughput under *hot-spot* traffic. Hot-spot nodes at the center of the network (Hotspot-4c), at the top right corner of the mesh (Hotspot-4tr), and at the right side of the mesh (Hotspot-4r).

5.2.1 Homogeneous 2D Mesh NoC

Fig. 7 (top) shows the results obtained under uniform traffic. We observe that nonadaptivity of the XY algorithm results in a higher saturation point as compared to other algorithms. The main reason for this is that the XY algorithm is based on long-term global information [9]. Routing packets along one dimension and then the other, the algorithm distributes the traffic in the most uniform manner possible in the long term. Adaptive algorithms, on the other hand, select the routing paths on the basis of short-term local information. Their way of operating tends to create "zigzag" paths that hinder the uniform distribution of traffic, causing a greater channel contention that deteriorates performance at higher *pir* rates.

Fig. 7 (middle) shows the average delay and throughput for locality traffic with $ohp = 0.4$ and $\rho = 2$. The high adaptiveness exhibited by APSRA for this traffic scenario (see Fig. 5) does not translate to an improvement in delay and throughput. This is due to the fact that the communicating pairs are very close to each other and the number of alternative paths is not so high compared to the single path required by a deterministic routing algorithm.

For the *Transpose 1* and *Transpose 2* traffic scenarios [9], APSRA outperforms the other routing strategies. Fig. 7 (bottom) shows the results for *Transpose 2* (similar results have been obtained for *Transpose 1*). The maximum load that the network is capable of handling using a deterministic routing is 0.012 packets/cycle/IP. This value increases to 0.017 packets/cycle/IP when Odd-Even is used and reaches 0.028 packets/cycle/IP with APSRA.

A more realistic traffic scenario is the *hot spot* [49]. In the hot-spot traffic scenario, some nodes are designated as the *hot-spot nodes*, which receive hot-spot traffic in addition to regular uniform traffic. Given a hot-spot percentage h, a newly generated packet is directed to each hot-spot node with an additional h percent probability. We consider three hot-spot traffic scenarios. In the first one, the hot-spot nodes are located at the center of the mesh [(3, 3), (4, 3), (3, 4), (4, 4)] with 20 percent hot-spot traffic. In the second scenario, the hot-spot nodes are located at the top-right corner of the mesh [(0, 6), (0, 7), (1, 6), (1, 7)] with 20 percent hot-spot traffic. In the third scenario, the hot-spot nodes are located at the right side of the mesh $[(i, 7), i = 0, 1, \ldots, 7]$ with 10 percent hot-spot traffic. Fig. 8 shows simulation results. We observe that in all the three scenarios, APSRA outperforms the other algorithms. In particular, the difference between deterministic and adaptive routing algorithms is more evident for the second and third scenario. In fact, an 8×8 NoC with a hot-spot node located at the center can be seen as a set of four small and isolated 4×4 NoCs (one NoC for each quadrant), each one stimulated with hot-spot traffic where the hot-spot node is located at one of the four corners. Since adaptiveness is really exploited in a large NoC, it justifies this behavior.

Finally, as a more realistic communication scenario, we consider a generic multimedia system (MMS), which includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [38],

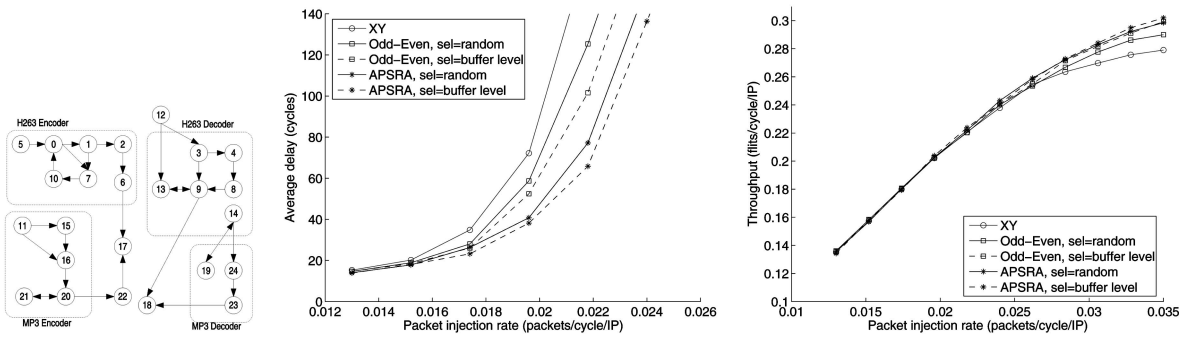


Fig. 9. Traffic generated by an MMS. (a) Communication graph. (b) Delay variation. (c) Throughput variation.

where the communication graph is shown in Fig. 9a. The application is partitioned into 40 distinct tasks, and then, these tasks were assigned and scheduled onto 25 selected IPs. The topological mapping of IPs into tiles of a 5×5 mesh-based NoC architecture has been obtained by using the approach presented in [36]. For this scenario, we consider self-similar packet injection distribution. Self-similar traffic has been observed in the bursty traffic between on-chip modules in typical MPEG-2 video applications [51] and networking applications [52]. Figs. 9b and 9c show the average delay and throughput variation for different injection loads. For an injection load of 0.018 packets/cycle/IP, that is, below the saturation for all the routing algorithms, the average delay is 36 cycles for XY, 32 cycles for Odd-Even, and 26 for APSRA when a random selection policy is used. By using a selection policy based on buffer levels, the average delay decreases to 30 cycles for Odd-Even and to 23 cycles for APSRA.

Tables 1 and 2 report a summary of the results both in terms of average delay and maximum injection rate sustainable by the network. With regard to the average delay, it has been measured at a packet injection rate where none of the algorithms are saturated.[2] The maximum packet injection rate sustainable by the network is the minimum *pir* that saturate the network. The saturating injection rate has been calculated as the *pir* at which the throughput drops by more than 5 percent from the earlier averaged slopes.

For each traffic scenario and for each routing algorithm, Table 1 reports the packet injection rate at the saturation point. It also shows the percentage improvement of APSRA over XY and Odd-Even. On the average, APSRA outperforms deterministic XY routing by 55 percent and adaptive Odd-Even routing by 27 percent.

Table 2 reports for each traffic scenario and for each routing algorithm the average delay measured at an injection load below saturation. It also shows the percentage improvement of APSRA over XY and Odd-Even. On the average, APSRA outperforms both deterministic XY and adaptive Odd-Even by almost 50 percent and 30 percent, respectively, in terms of average delay.

2. A network is said to start saturating when the increase in applied load does not result in a linear increase in throughput [7].

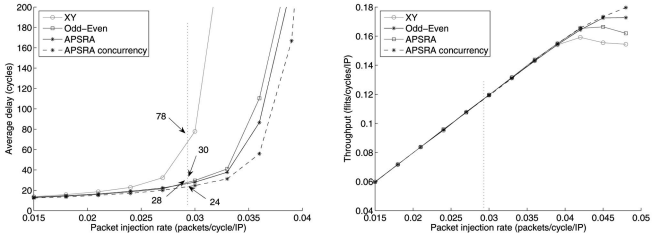
We conclude this section with the last experiment aiming to show how performance can be improved by exploiting concurrency information. Let us consider again the MMS, but this time, two communication scenarios are used. To the first one belong all the communications that refer to the encoding part of the codec. To the second one are those that refer to the decoding part of the codec. The 25 cores implementing the system have been randomly mapped on the NoC. The degree of adaptiveness is 0.62 and 0.79 for Odd-Even and APSRA, respectively. It reaches 0.98 for APSRA when the communication concurrency information is considered. The average delay and throughput variations for different injection loads are reported in Fig. 10. As can be observed, a great improvement in delay can be obtained by using adaptive routing algorithms. For an injection load of 0.03 (below saturation for all the algorithms, as can be

TABLE 1
Improvement in Saturation Point of APSRA as Compared to XY and Odd-Even for Different Traffic Scenarios

Traffic scenario	Max. *pir* (packets/cycle/IP) XY	OE	APSRA	APSRA improvement vs. XY	vs. OE
Random	.0120	.0105	.0080	-33%	-23%
Locality	.0190	.0200	.0210	10.5%	5.0%
Transpose 1	.0110	.0150	.0270	145.5%	80.0%
Transpose 2	.0110	.0160	.0270	145.5%	68.8%
Hotspot-4c	.0033	.0035	.0038	13.6%	7.1%
Hotspot-4tr	.0027	.0031	.0035	29.6%	12.9%
Hotspot-8r	.0039	.0059	.0067	71.8%	13.6%
MMS	.0174	.0174	.0196	12.6%	12.6%
Average improvement				54.6%	26.8%

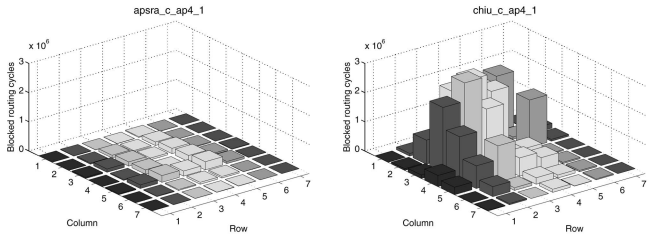
TABLE 2
Improvement in Average Delay of APSRA as Compared to XY and Odd-Even for Different Traffic Scenarios

Traffic scenario	*pir* (pkts/cyc/IP)	Avg. delay (cycles) XY	OE	APSRA	APSRA impr. vs. XY	vs. OE
Random	0.007	18	18	23	-27%	-27%
Locality	0.020	39	34	29	24.2%	13.2%
Transpose 1	0.011	91	39	19	79.2%	51.1%
Transpose 2	0.011	82	31	19	76.6%	38.4%
Hotspot-4c	0.003	46	50	34	26.7%	32.1%
Hotspot-4tr	0.003	52	37	30	42.2%	17.5%
Hotspot-8rs	0.003	34	25	20	41.8%	21.5%
MMS	0.018	36	30	23	36.1%	23.3%
Average improvement					47.1%	28.7%



(a) (b)

Fig. 10. Variation in (a) delay and (b) throughput for a multimedia application and two communication scenarios.



(a) (b)

Fig. 12. Blocked routing cycles/router with (a) APSRA and (b) Chen and Chiu's algorithm.

seen from the throughput curve), APSRA performs better than Odd-Even by almost 7 percent (28 cycles versus 30 cycles). Using the concurrency version of APSRA, the improvement grows to 20 percent (24 cycles).

5.2.2 Nonhomogeneous 2D Mesh NoC with Regions

In this section, we analyze the performance of APSRA for a 2D mesh NoC with regions [2]. We compare APSRA with a fixed version of the Chen and Chiu's algorithm [29], which is adopted from the area of fault-tolerant routing.

Experiments are carried out on a 7×7 mesh topology NoC with regions. The destinations for generated packets are randomly selected with a hot-spot probability of 60 percent for region access points. We compare APSRA and Chen and Chiu's algorithm with a region of size 2×2, either in the bottom-left corner with 3 access points (bl_ap3) or in center of the network with 4 access points (c_ap4).

We classify communication traffic into three types, namely, as communication traffic to a region, as other traffic, where a resource other than the region is a destination, and as all communications, which is the aggregate of the first two types of traffic.

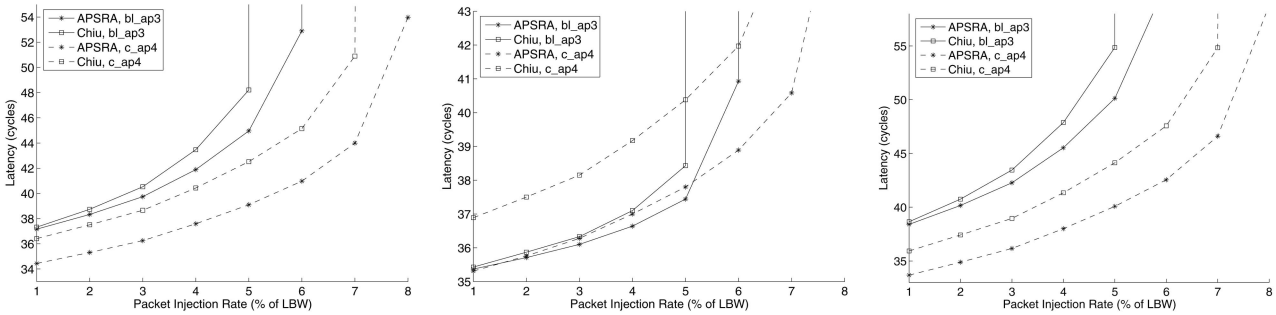
The first result shows the average latency for all communications in the network, as depicted in Fig. 11a. The lowest latency values are obtained for APSRA with a central region (apsra_c_ap4). The second lowest latency values are obtained with Chen and Chiu's algorithm with a central region (chiu_c_ap4). After this is APSRA with a region in the bottom-left corner (apsra_bl_ap).

The worst performance is shown by Chen and Chiu's algorithm with a region in the bottom-left corner (chiu_bl_ap3). In Fig. 11b, we give the average latency for traffic with destinations other than the region. The worst position from the latency point of view, up to an injection rate of 5 percent, is with Chen and Chiu's algorithm with a region in the center (chiu_c_ap4). In this case, all the other combinations provide similar latency values in this range. However, when the injection rate is increased to above 5 percent, Chen and Chiu's algorithm with a region in the corner position (chiu_bl_ap3) rapidly saturates. Next to saturate is APSRA with a region in the corner (apsra_bl_ap3). The best result from the saturation point of view is when using APSRA with a region in the center (apsra_c_ap4), although it has slightly higher latency at lower injection rates.

Fig. 11c shows results for traffic destined only to a region. In this case also, APSRA with a central region shows the best performance results in terms of low latency. Still, Chen and Chiu's algorithm with a central region clearly gives better results than both algorithms with a region at the bottom-left position. The worst performance is also in this case shown by Chen and Chiu's algorithm with a region in the bottom-left corner.

Fig. 12 gives more details about what causes the difference in latency values. The diagrams present values on how many routing cycles the packets were blocked in different routers (results are from simulations with $pir = 0.1$). Note that the scale of blocked routing cycles is not the same in the two diagrams.

Figs. 12a and 12b reveal that APSRA does not cause as much blockage as Chen and Chiu's algorithm does. Note that Chen and Chiu's algorithm result in more blockages



(a) (b) (c)

Fig. 11. Average latency versus packet injection rate in percentage of the link bandwidth with regions in the bottom left and the center. (a) For all communications. (b) For communications destined outside the region. (c) For communications destined to a region.

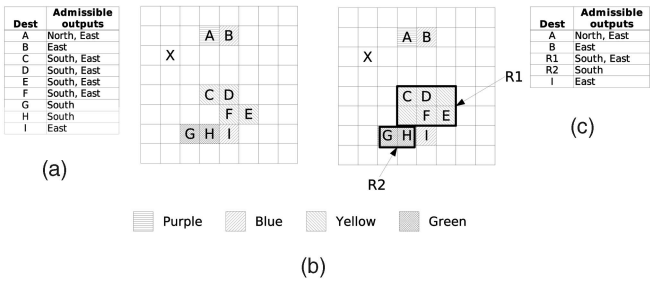


Fig. 13. The basic idea of the routing compression method. (a) Routing table before compression. (b) Color-based clustering. (c) Compressed routing table.

close to the north and west border of the region. The reason is that this path is highly utilized by the algorithm in the procedures of routing around the region border. APSRA, on the other hand, is not biased toward specific routes and thus spreads the traffic more evenly around the border. As APSRA in many situations have several paths to select from, it is also possible to avoid congested routes, which further decreases the blockage.

6 Architectural Implications of APSRA

One way to implement the routing function is to design it in hardware logic (also known as the algorithmic approach). In this case, there can, e.g., be a state machine that takes the current node address, the destination address, and some status information stored in the router as inputs and outputs the admissible out ports.

Another way to implement the routing function is to use a routing table [53]. The index to the table, where the admissible outputs are stored, is the destination address or a function of the destination address. The values of the table are dependent on which router it is implemented.

The algorithmic approach comes at the cost of a loss of generality with respect to the table-based approach. In fact, the algorithm is only specific to one topology and to one routing strategy on that topology [54]. On the other hand, this mechanism turns out to be more efficient (areawise and performancewise) than table-based routing. However, routing tables introduce an additional degree of freedom in routing algorithms as they can be reconfigured at runtime [41], [42] to adapt to the current traffic condition or offline to manage network topology variation due to manufacturing defects [33]. A disadvantage is that a table can take a large space if many destination addresses should be stored. That is, in such an implementation, the cost (silicon area) of the router will be proportional to the size of the routing table.

6.1 Routing Table Compression

In [55], we proposed a method to compress the routing table to reduce its size such that the resulting routing algorithm remains deadlock free and has high adaptivity. The basic idea of the compressing method is shown in Fig. 13. Let us suppose that the routing table associated to the west input port of node X is that shown in Fig. 13a. Since we are considering minimal routing, the destination node for a packet received from the west port will be in the right part of the NoC. Now, let us suppose to associate a color for each

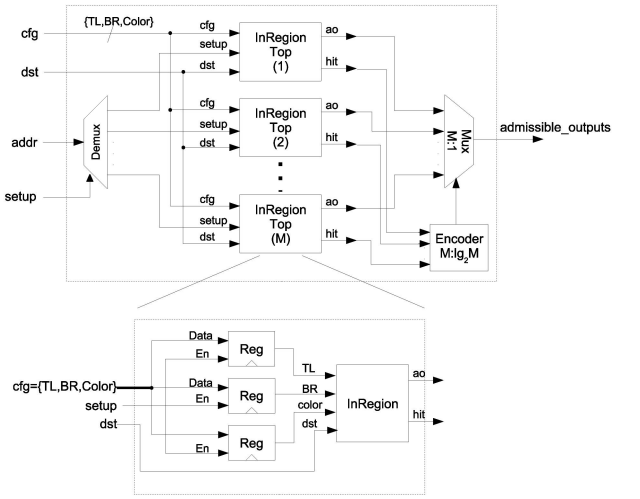


Fig. 14. Block diagram of the architecture implementing the routing function that uses the compressed routing table.

one of the five possible sets of admissible output directions. For instance, we mark with color Red (Green/Blue) all the destination nodes that, to be safely reached, needs that a packet in the west input port must be forwarded through the North (South/East) output port. Similarly, we mark with color Purple (Yellow) all the destination nodes that, to be safely reached, needs that a packet in the west input port must be forwarded through the North or East (South or East) output channels. The left side of Fig. 13b shows the set of destinations colored on the basis of the information stored in the routing table. The next step is to perform a color-based clustering of the destination nodes by means of rectangular regions. The right side of Fig. 13b shows an example of such a clustering. The basic idea of the compression technique is that it is no longer necessary to store the set of all the destinations, but only the set of clusters, as shown in Fig. 13c.

If a small loss in adaptivity is allowed, a cluster merging procedure can be used to increase the compression ratio. For instance, in Fig. 13c, clusters $R1$ (yellow) and $R2$ (green) can be merged to form a single cluster $R3$ (yellow) whose destinations can be reached by using only the east output port. As reported in [55], in practical cases, using from four to eight clusters is enough to cover all practical traffic scenarios without any loss in adaptivity. Using from two to three clusters gives a small reduction in adaptivity, which translates to less than 3 percent performance degradation as compared to uncompressed routing tables.

From the architectural viewpoint, the logic to manage the compressed routing table is very simple. Fig. 14 shows the block diagram of the architecture implementing the routing function that uses the compressed routing table. For a given destination *dst*, the block extracts from the compressed routing table the set of admissible outputs. The input port *setup* allows us to configure the router. When the *setup* is asserted, the *addr* input is used to select the region to be configured. Configuring a region means storing attributes characterizing the region (top right, bottom left, and color) into the compressed routing table. The block *InRegion* checks if a destination *dst* belongs to a region identified by

TABLE 3
Silicon Area Occupancy in Square Micrometers for Various Blocks of a Router

	XY	Odd-Even	RT4	RT8
Routing function	803.52	1365.12	17051.9	36384.77
Input FIFO x 5	132891.84	132891.84	132891.84	132891.84
Crossbar	14041.73	14041.73	14041.73	14041.73
Arbiter	2007.94	2008.94	2009.94	2010.94
Total	149745.02	150307.62	165995.4	185329.27

its top-left corner (TL) and its bottom-right corner (BR). If this condition is satisfied, the output *ao* assumes the value of the *color* input, and output *hit* is set.

6.2 Table-Based Implementation: Area Overhead Analysis

To evaluate the overhead in silicon area due to the use of routing table with respect to routing logic, we designed in VHDL and synthesized by using Synopsys Design Compiler for a UMC 0.13-μm technology library the following blocks:

- *Routing function.* It is the block that gives the set of admissible outputs for the current node and a given destination. We designed three different blocks that implement the XY routing function, the Odd-Even routing function, and the routing-table-based routing function. The first two are pure combinatorial logic. The latter one, whose block diagram is depicted in Fig. 14, contains memory elements also. With regard to the routing-table-based routing block, we considered two different instances able to manage four regions (RT4) and eight regions (RT8), respectively.
- *Input FIFO* $\times$ 5. They are the FIFO buffers at the input of each router. For a mesh-based network topology, there are five FIFO buffers in total. We considered four entry FIFO buffers with an entry size of 64 bits (flit size).
- *Crossbar.* It is a general 5 $\times$ 5 crossbar block that allows us to simultaneously route nonconflicting packets.
- *Arbiter.* It is a general arbiter that manages the situation where several packets simultaneously want to use the same output. In this case, arbitration between these packets has to be performed. We used a round-robin policy.

Table 3 reports the silicon area for each of the main blocks of a router for three different routers implementing XY routing, Odd-Even routing, and table-based routing, respectively. With regard to the latter, two different implementations able to manage compressed routing tables with a budget of four (RT4) and eight (RT8) table entries have been considered. The cost overhead of a routing table implementation based on the proposed compression technique and architecture represents only a small fraction of the overall router cost. The overhead over an XY router is 10.9 percent and 23.8 percent for RT4 and RT8, respectively. The overhead over an Odd-Even router is 10.4 percent and 23.3 percent for RT4 and RT8, respectively.

7 Conclusions

In this paper, we have proposed a topology-agnostic methodology to develop efficient APSRAs for NoCs. Our methodology not only uses the information about the topology of communicating cores but also exploits information about the concurrency of communication transactions. Results based on analysis and simulation-based evaluation demonstrate that routing algorithms developed using our approach significantly outperform general-purpose routing algorithms like XY and Odd-Even for a mesh topology NoC. We also show that routing algorithms generated by the APSRA methodology has even higher performance and adaptivity advantage over other deadlock-free routing algorithm for nonhomogeneous mesh NoCs. The APSRA methodology uses a heuristic to remove the minimum number of edges in the channel dependency graph to ensure deadlock-free routing. The heuristic does not guarantee an optimal routing algorithm and may not be able to find a routing algorithm that can route all messages (even if it exists). There is a scope of improving this heuristic to get higher adaptivity of the resultant routing algorithm. The APSRA methodology assumes that communication concurrency information can be organized as communication scenarios and is available after the task mapping and scheduling step of a system-level design. Developing an interval-based task mapping and scheduling methodology is an interesting open research problem. Our approach also assumes configurable SRAM-based tables in every router of the network. A straightforward table implementation may not be practically useful, since the table size grows with the number of nodes in the network. Therefore, we have also proposed a technique to compress tables such that the effect on the original routing algorithm is minimal.

Acknowledgments

This work was partially supported by the European Commission in the context of the HiPEAC network of excellence (project 004408 and 217068), under the research cluster on High Performance Interconnection Networks for Embedded Applications.

References

[1] A. Ivanov and G.D. Micheli, "The Network-on-Chip Paradigm in Practice and Research," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 399-403, Sept.-Oct. 2005.
[2] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A Network on Chip Architecture and Design Methodology," *Proc. IEEE CS Ann. Symp. VLSI*, p. 117, 2002.
[3] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. 38th Design Automation Conf. (DAC '01)*, pp. 684-689, 2001.
[4] F. Karim, A. Nguyen, and S. Dey, "An Interconnect Architecture for Networking Systems on Chips," *IEEE Micro*, vol. 22, no. 5, pp. 36-45, Sept.-Oct. 2002.
[5] P.P. Pande, C. Grecu, A. Ivanov, and R. Saleh, "Design of a Switch for Network on Chip Applications," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS '03)*, vol. 5, pp. 217-220, May 2003.
[6] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 1-51, 2006.

[7] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Computers,* vol. 54, no. 8, pp. 1025-1040, Aug. 2005.

[8] D. Linder and J. Harden, "An Adaptive and Fault-Tolerant Wormhole Routing Strategy for k-Ary n-Cubes," *IEEE Trans. Computers,* vol. 40, no. 1, pp. 2-12, Jan. 1991.

[9] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *J. Assoc. for Computing Machinery,* vol. 41, no. 5, pp. 874-902, Sept. 1994.

[10] A.A. Chien and J.H. Kim, "Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors," *J. ACM,* vol. 42, no. 1, pp. 91-123, Jan. 1995.

[11] J. Upadhyay, V. Varavithya, and P. Mohapatra, "A Traffic-Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes," *IEEE Trans. Computers,* vol. 46, no. 2, pp. 190-197, Feb. 1997.

[12] G.-M. Chiu, "The Odd-Even Turn Model for Adaptive Routing," *IEEE Trans. Parallel and Distributed Systems,* vol. 11, no. 7, pp. 729-738, July 2000.

[13] J. Hu and R. Marculescu, "DyAD—Smart Routing for Networks-on-Chip," *Proc. 41st Design Automation Conf. (DAC '04),* pp. 260-263, June 2004.

[14] A. Hansson, K. Goossens, and A. Rădulescu, "A Unified Approach to Mapping and Routing on a Network-on-Chip for Both Best-Effort and Guaranteed Service Traffic," *VLSI Design,* vol. 2007, 2007.

[15] T. Skeie, O. Lysne, J. Flich, P. Lépez, A. Robles, and J. Duato, "LASH-TOR: A Generic Transition-Oriented Routing Algorithm," *Proc. 10th Int'l Conf. Parallel and Distributed Systems (ICPADS '04),* pp. 595-604, 2004.

[16] S. Kaist, *Bone: Network on Chip, Real Chip Implementation,* http://ssl.kaist.ac.kr/ocn/, 2008.

[17] F. Angiolini, P. Meloni, S.M. Carta, L. Raffo, and L. Benini, "A Layout-Aware Analysis of Networks-on-Chip and Traditional Interconnects for MPSoCs," *IEEE Trans. Computer Aided Design of Integrated Circuits and Systems,* vol. 26, no. 3, pp. 421-434, Mar. 2007.

[18] T. Ahonen, D.A. Sigüenza-Tortosa, H. Bin, and J. Nurmi, "Topology Optimization for Application-Specific Networks-on-Chip," *Proc. Sixth Int'l Workshop System-Level Interconnect Prediction (SLIP '04),* pp. 53-60, 2004.

[19] A. Pullini, F. Angiolini, P. Meloni, D. Atienza, S. Murali, L. Raffo, G.D. Micheli, and L. Benini, "NoC Design and Implementation in 65 nm Technology," *Proc. First Int'l Symp. Networks-on-Chip (NOCS '07),* pp. 273-282, 2007.

[20] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "× Pipes: A Latency Insensitive Parameterized Network-on-Chip Architecture for Multi-Processor SoCs," *Proc. 21st Int'l Conf. Computer Design (ICCD '03),* pp. 536-541, 2003.

[21] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G.D. Micheli, "× Pipes Lite: A Synthesis Oriented Design Library for Networks on Chips," *Proc. Conf. Design, Automation and Test in Europe (DATE '05),* vol. 2, pp. 1188-1193, 2005.

[22] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC," *Proc. 18th Symp. Integrated Circuits and System Design,* pp. 178-183, 2005.

[23] J.-P. Soininen and H. Heusala, "Networks on Chip," *A Design Methodology for NoC-Based Systems,* chapter 2, pp. 19-38, Kluwer Academic Publishers, 2004.

[24] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[25] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Routing in Wormhole Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 6, no. 10, pp. 1055-1067, Oct. 1995.

[26] T.T. Ye, L. Benini, and G.D. Micheli, "Packetization and Routing Analysis of On-Chip Multiprocessor Networks," *J. System Architectures,* vol. 50, no. 2-3, pp. 81-104, 2004.

[27] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load Distribution with the Proximity Congestion Awareness in a Network on Chip," *Proc. Conf. Design, Automation and Test in Europe (DATE '03),* pp. 1126-1127, 2003.

[28] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip," *IEEE Trans. Computers,* vol. 57, no. 6, pp. 809-820, June 2008.

[29] R. Holsmark and S. Kumar, "Design Issues and Performance Evaluation of Mesh NoC with Regions," *Proc. 23rd IEEE Norchip Conf.,* pp. 40-43, Nov. 2005.

[30] A. Jouraku, M. Koibuchi, and H. Amano, "L-Turn Routing: An Adaptive Routing in Irregular Networks," Technical Report 59, IEICE, Apr. 2001.

[31] L. Cherkasova, V. Kotov, and T. Rokicki, "Fibre Channel Fabrics: Evaluation and Design," *Proc. 29th Hawaii Int'l Conf. System Sciences (HICSS '96),* pp. 53-58, 1996.

[32] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Scheme for Networks of Workstations," *Proc. Third Int'l Symp. High Performance Computing (ISHPC '00),* pp. 260-267, 2000.

[33] A. Mejia, J. Flich, J. Duato, S.-A. Reinemo, and T. Skeie, "Segment-Based Routing: An Efficient Fault-Tolerant Routing Algorithm for Meshes and Tori," *Proc. 20th Int'l Parallel and Distributed Processing Symp. (IPDPS '06),* Apr. 2006.

[34] W.J. Dally and C. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers,* vol. C, no. 36, pp. 547-553, 1987.

[35] J.-M. Chang and M. Pedram, "Codex-Dp: Co-Design of Communicating Systems Using Dynamic Programming," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 19, no. 7, pp. 732-744, July 2002.

[36] G. Ascia, V. Catania, and M. Palesi, "Multi-Objective Mapping for Mesh-Based NoC Architectures," *Proc. Second IEEE/ACM/IFIP Int'l Conf. Hardware/Software Codesign and System Synthesis,* pp. 182-187, Sept. 2004.

[37] S. Murali and G.D. Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," *Proc. Conf. Design, Automation, and Test in Europe (DATE '04),* pp. 896-901, Feb. 2004.

[38] J. Hu and R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 4, pp. 551-562, Apr. 2005.

[39] U.Y. Ogras and R. Marculescu, "It's a Small World After All: NoC Performance Optimization via Long-Range Link Insertion," *IEEE Trans. Very Large Scale Integration Systems,* vol. 14, no. 7, pp. 693-706, July 2006.

[40] S. Murali, D. Atienza, L. Benini, and G.D. Micheli, "A Multi-Path Routing Strategy with Guaranteed In-Order Packet Delivery and Fault-Tolerance for Networks on Chip," *Proc. 43rd Design Automation Conf. (DAC '06),* pp. 845-848, July 2006.

[41] J. Duato, O. Lysne, R. Pang, and T.M. Pinkston, "Part I: A Theory for Deadlock-Free Dynamic Network Reconfiguration," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 5, pp. 412-427, May 2005.

[42] O. Lysne, T.M. Pinkston, and J. Duato, "Part II: A Methodology for Developing Deadlock-Free Dynamic Network Reconfiguration Processes," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 5, pp. 428-443, May 2005.

[43] M. Palesi, S. Kumar, R. Holsmark, and V. Catania, "Exploiting Communication Concurrency for Efficient Deadlock Free Routing in Reconfigurable NoC Platforms," *Proc. 21st Int'l Parallel and Distributed Processing Symp. (IPDPS '07),* pp. 1-8, Mar. 2007.

[44] T. Skeie, O. Lysne, and H. Theiss, "Layered Shortest Path (LASH) Routing in Irregular System Area Networks," *Proc. 16th Int'l Parallel and Distributed Processing Symp. (IPDPS '02),* pp. 162-169, 2002.

[45] K. Aoyama and A.A. Chien, "The Cost of Adaptivity and Virtual Lanes in a Wormhole Router," *J. VLSI Design,* vol. 2, no. 4, pp. 315-333, 1995.

[46] A.S. Vaidya, A. Sivasubramaniam, and C.R. Das, "Impact of Virtual Channels and Adaptive Routing on Application Performance," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 2, pp. 223-237, Feb. 2001.

[47] T. Marescaux and H. Corporaal, "Introducing the SuperGT Network-on-Chip; SuperGT QoS: More Than Just GT," *Proc. 44th Design Automation Conf. (DAC '07),* pp. 116-121, 2007.

[48] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Design and Test of Computers,* vol. 22, no. 5, pp. 21-31, Sept.-Oct. 2005.

[49] R.V. Boppana and S. Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms," *Proc. 20th Ann. Int'l Symp. Computer Architecture (ISCA '93),* pp. 351-360, May 1993.

[50] F. Fazzino, M. Palesi, and D. Patti, *Noxim: Network-on-Chip Simulator,* http://noxim.sourceforge.net, 2008.

[51] G. Varatkar and R. Marculescu, "Traffic Analysis for On-Chip Networks Design of Multimedia Applications," *Proc. 39th Design Automation Conf. (DAC '02)*, pp. 510-517, June 2002.
[52] D.R. Avresky, V. Shubranov, R. Horst, and P. Mehra, "Performance Evaluation of the ServerNetR SAN under Self-Similar Traffic," *Proc. 13th Int'l Parallel Processing Symp./10th Symp. Parallel and Distributed Processing (IPPS/SPDP '99)*, pp. 143-149, Apr. 1999.
[53] M. Monchiero, G. Palermo, C. Silvano, and O. Villa, "Exploration of Distributed Shared Memory Architectures for NoC-Based Multiprocessors," *J. Systems Architecture*, vol. 53, no. 10, pp. 719-732, 2007.
[54] L. Benini and D. Bertozzi, "System-on-Chip: Next Generation Electronics," *IEE Circuits, Devices and System Series*, chapter 17, Network-on-Chip Architectures and Design Methods, pp. 589-624, 2006.
[55] M. Palesi, S. Kumar, and R. Holsmark, "A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architectures," *Proc. Sixth Int'l Workshop Systems, Architectures, Modeling, and Simulation (SAMOS '06)*, pp. 373-384, July 2006.

Maurizio Palesi received the Laurea degree and the PhD degree in computer engineering from the Università di Catania, Catania, Italy, in 1999 and 2003, respectively. Since December 2003, he has held a research contract as an assistant professor in the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facoltà di Ingegneria, Università di Catania. His research focuses on platform-based system design, design space exploration, low-power techniques for embedded systems, and network-on-chip architectures. Since May 2007, he has been on the editorial board of *VLSI Design* as an associate editor. He is a member of the IEEE.

Rickard Holsmark received the BS degree in electronics, with specialization in microcontroller systems, and the MS degree in electronics, with specialization in embedded systems, from Jönköping University, Jönköping, Sweden, in 2001 and 2003, respectively. He is currently a PhD student with the Embedded System Group, Department of Electronics and Computer Engineering, School of Engineering, Jönköping University. His research is focused toward specialized architectures and routing algorithms for networks on chip. His other interests include system-level design and processor architectures for embedded systems.

Shashi Kumar received BTech, MTech, and PhD degrees from the Indian Institute of Technology, Delhi, in 1974, 1976, and 1985, respectively. He is a professor of embedded systems in the Department of Electronics and Computer Engineering, School of Engineering, Jönköping University, Jönköping, Sweden. His research interests include system-level modeling and synthesis, parallel architectures and algorithms, reconfigurable computing, and heuristic search algorithms. He was a member of the team that was the first to propose the idea of packet switched networks for on-chip communication and coined the term network on chip (NoC) in 2000. He has interest in various aspects of NoC design, including NoC topologies, QoS issues in NoC communication, NoC architectural modeling and evaluation, application-specific NoC architecture design, mapping applications to NoC platforms, and testing of NoC. He is a senior member of the IEEE.

Vincenzo Catania received the Laurea degree in electrical engineering from the Università di Catania, Catania, Italy, in 1982. Until 1984, he was responsible for testing microprocessor system at STMicroelectronics, Catania. Since 1985, he has cooperated in research on computer network with the Istituto di Informatica e Telecomunicazioni, Università di Catania, where he is a full professor of computer science. His research interests include performance and reliability assessment in parallel and distributed system, VLSI design, low-power design, and fuzzy logic.