Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip

Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, *Member*, *IEEE Computer Society*, and Davide Patti, *Member*, *IEEE Computer Society*

Abstract—Efficient and deadlock-free routing is critical to the performance of networks-on-chip. The effectiveness of any adaptive routing algorithm strongly depends on the underlying selection strategy. When the routing function returns a set of admissible output channels with a cardinality greater than one, a selection function is used to select the output channel to which the packet will be forwarded. In this paper, we present a novel selection strategy that can be coupled with any adaptive routing algorithm. The proposed selection strategy is based on the concept of *Neighbors-on-Path*, the aim of which is to exploit the situations of indecision occurring when the routing function returns several admissible output channels. The overall objective is 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. Performance evaluation is carried out by using a flit-accurate simulator under traffic scenarios generated by both synthetic and real applications. Results obtained show how the proposed selection strategy applied to the Odd-Even routing algorithm yields an improvement in both average delay and saturation point up to 20 percent and 30 percent on average, respectively, with minimal overhead in terms of area occupation. In addition, as a consequence of the improved router efficiency, a positive effect on total energy consumption is also observed under near-congestion packet injection rates.

Index Terms—Networks-on-chip, adaptive routing, deadlock-free routing, selection strategy, performance analysis.

1 Introduction

The on-chip interconnection system represents one of the major elements which has to be optimized in designing a complex digital system [1]. The International Technology Roadmap for Semiconductors [2] foresees that it will represent the limiting factor for performance and power consumption in next generation systems-on-chip (SoCs). The continuous reduction in the time-to-market required by the telecommunications, multimedia, and consumer electronics market makes full-custom design of an interconnection system inappropriate and has led to the definition of design methodologies focusing on design reuse. This is confirmed by the great standardization effort made by the VSI Alliance [3] and the development, by the major EDA and semiconductor companies, of on-chip interconnection systems that are easy to integrate and scale [4], [5], [6], [7], [8]. However, although they are good solutions for simple SoCs integrating up to 10 bus masters, their use in next-generation systems, which are likely to integrate hundreds of modules, seems hardly feasible. The limiting factor is mainly the topological organization of the interconnection between the various units, which will remain substantially bus-based. Bus-based synchronous communication architectures in large SoCs operating at several hundred megahertz have tight timing constraints, slow wires, and require tight clock-skew control. This creates timing closure issues that can be tackled by decoupling communication from computation and using a globally asynchronous locally synchronous (GALS) design paradigm [9].

A type of architecture that emphasizes modularity and is intrinsically oriented toward supporting heterogeneous implementations is represented by network-on-chip (NoC) architectures [10]. These architectures loosen the bottleneck due to delays in signal propagation in deep-submicron technologies and provide a natural solution to the problem of core reuse by standardizing on-chip communications.

The overall performance of an NoC depends on several network properties, such as topology, routing algorithm, flow control, and switching technique. There has been significant work published on efficient routing schemes in the parallel and distributed computing areas. The important concerns of published research include the design of low-cost, high-performance, reliable, and flexible on-chip router architectures, the development of deadlock-free and livelock-free adaptive routing algorithms, and the design of contention and congestion-aware selection schemes. Selection schemes (also known as selection policies) strongly affect the overall performance of any adaptive routing algorithm [11], [12], [13] and represent the main topic of this paper.

1.1 Motivations

Wormhole switching [14] has emerged as the most widely adopted switching technique for NoC routers for two main reasons. First, it requires smaller buffers as compared to the

• *The authors are with the Dipartimento di Ingengneria Informatica e delle Telecomunicazioni, Università di Catania, V.le Andrea Doria, 6, 95125 Catania, Italy. E-mail: {gascia, vcatania, mpalesi, dpatti}@diit.unict.it.*

Manuscript received 2 Feb. 2007; revised 14 Sept. 2007; accepted 6 Feb. 2008; published online 21 Feb. 2008.
Recommended for acceptance by F. Lombardi.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0048-0207.
Digital Object Identifier no. 10.1109/TC.2007.38.

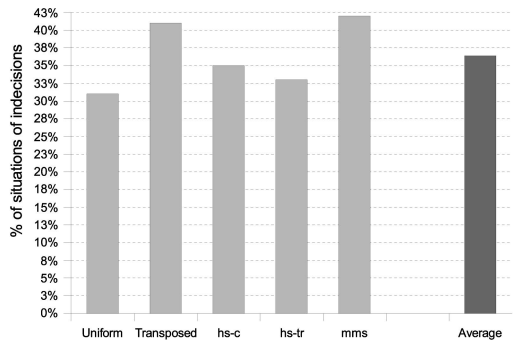


Fig. 1. Percentage of situations of indecision for different traffic scenarios.

store-and-forward switching scheme. Second, network latency becomes relatively insensitive to path length. In wormhole switching, each packet is serialized into a sequence of flow control units (flits). When the header flit of a packet arrives at a node, the routing function establishes the set of output channels it can be routed on. The body flits will then follow the reserved channel and the tail will later release the channel reservation. Unfortunately, blockage of the header flit of a packet blocks all of the remaining flits of the packet along the established path. The blocked header flit will have to wait for all of the flits of the blocking packet to pass. Routing a header flit along a path leading to a congested router is thus undesirable. In adaptive routing, if the set of output channels established by the router contains at least two *nonreserved*[1] output channels (i.e., the cardinality of the set of admissible output channels is greater than one), the router has to choose one of them. The further phase required to solve these *situations of indecision* is usually referred to as *selection policy* [15].

For different traffic scenarios, which will be described in Section 4, Fig. 1 shows the percentage of situations of indecision for an 8×8 mesh-based NoC with routers implementing the Odd-Even (OE) routing algorithm [16] and four flit input buffers, under nonsaturated traffic conditions. In particular, with the term "situations of indecision," we hereby mean the relationship, expressed as a percentage, between the number of times a router is able to route a packet toward alternative channels (i.e., more than one possible and available output channel not reserved for other packets) and the total number of packets. As can be observed, on average, the percentage of situations of indecision is over 36 percent, which represents an important optimization opportunity. Similar results can be found with different network configurations (e.g., buffer size, NoC size, packet length, etc.).

1.2 Contribution

The main focus of this work is to develop a selection strategy with the aim of allocating the channels that will allow the packets to be routed to their destination along a path that is as free as possible of congested nodes. In contrast to classic computer networks, where internode information can only be exchanged through packets, on-chip networks can take advantage of dedicated control wires to transmit data between routers. This makes it easier to exchange useful information about congestion-related aspects such as the buffer status of specific nodes without adding further traffic overhead. In this paper, we want to exploit such NoC-specific capability in order to acquire the knowledge of buffer availability in nodes that reside beyond the boundaries of adjacent neighbors. In particular, we introduce the notion of *Neighbor-on-Path* (NoP), a set of nodes that can be computed for a given node and a specific header flit.

1. An output channel is said to be *nonreserved* if a header flit belonging to another packet has not reserved it to transmit the flit making up the packet.

1.3 Paper Organization

The remainder of this paper is organized as follows: In Section 2, we summarize some major contributions regarding the improvement of adaptive routing in NoCs. Section 3 illustrates the idea behind the proposed approach together with a formal description of the algorithm and the extra signals required. In Section 4, we apply the proposed selection strategy to a well-known adaptive routing function, giving a performance comparison of various deterministic and adaptive routers. In Section 5, we address some design issues, reporting a block scheme of an NoP-based router implementation and an area/energy overhead analysis. Finally, Section 6 concludes this paper and draws some directions for future developments.

2 Background

Routing schemes have been classified in several ways in the literature. In a scheme called *source routing*, the source node selects the entire path before sending the packet. The major drawback of this approach is that each packet must carry this routing information, increasing the packet size. In addition, the path cannot be changed after the packet has left the source. A more common solution is the use of *distributed routing*. Here, a router, upon receiving a packet, decides whether it should be delivered to the local resource or forwarded to a neighboring router. In the latter case, a routing algorithm is invoked (or a routing table is accessed) to determine to which neighbor the packet should be sent.

In terms of the way in which to choose a path among the set of possible paths from source to destination, routing algorithms can also be classified as *deterministic*, *oblivious*, and *adaptive*. In deterministic routing algorithms, the path from the source to the destination is completely determined by the source and the destination addresses. In oblivious routing, multiple paths from a source node to a destination node are available, but the algorithm always chooses a route without knowing about the state of the networks. Adaptive routing algorithms use information about the state of the network to make routing decisions. First, a *routing function* computes the set of admissible output channels toward which the packet can be forwarded to reach the destination. Then, a *selection function* is used to select one output channel from the set of admissible output channels depending on dynamic network conditions, such as the presence of faulty or congested channels (see Fig. 2).

Many routing algorithms for wormhole-switched networks have been proposed in the literature [16], [17], [18], [19], [20]. Glass and Ni [18] propose a turn model for designing wormhole routing algorithms for mesh and hypercube topology networks that are deadlock and livelock free. The model is based on analyzing the directions in which packets can turn in a network and the

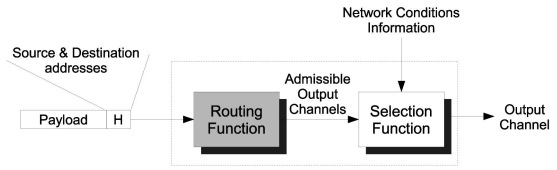


Fig. 2. Routing and selection blocks of a router.

cycles that the turns can form. 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 was later utilized by Chiu [16] to develop the *OE* adaptive routing algorithm for meshes without virtual channels. The model restricts the locations at which 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 more even for different source-destination pairs. Bolotin et al. [21] have proposed hard coded paths for deadlock safe routing for an application. In their approach, the possibility of deadlock for the application communication scenario is analyzed and solved offline. Any change in traffic patterns results in a complete reanalysis of deadlock freeness and may result in changes to be made to affected paths. A nonminimal deadlock-free routing algorithm is described for an irregular mesh topology NoC with regions in [22]. This algorithm is biased in favor of some area of the network as compared to the other area. Another adaptive routing algorithm, named DyXY, along with an analytical model based on queuing theory for a 2D mesh has been proposed by Li et al. [23]. Although the authors claim that DyXY ensures deadlock-free and livelock-free routing, they do not provide any formal proof. In the same paper, the authors present an analytical model based on queuing theory for a 2D mesh showing that analytical results match very well with the simulation results. An application specific routing algorithm named APSRA has been proposed by Palesi et al. [24]. APSRA exploits communication information to maximize the adaptivity while ensuring deadlock-free routing for an application. Unfortunately, in almost all of the papers reviewed above, there are no indications about the selection policy used when the routing function returns more than one admissible output ports. In [18], three selection policies have been used: zigzag, XY, and no turn. The results reported in [16] have been computed assuming a selection policy that always selects dimension 1 when multiple outputs are possible. In [24], the authors considered two different selection policies based on random and buffer occupancy.

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 deterministic routing, no virtual channels, and that can be adapted to be used in the NoC domain are up*/down* [25], lturn [26], smart-routing [27], FX [28], and Segment-based routing [29].

To the best of our knowledge, only a few contributions have been devoted to the definition of new selection strategies. In [11], Schwiebert and Bell present a detailed simulation study of various selection functions for several fully adaptive wormhole routing algorithms for 2D meshes. The obtained results show that the choice of selection function has a significant effect on the average message latency and saturation behavior. Similarly, Feng and Shin [12], in order to study the dependencies between routing algorithms and communication workloads, have performed a set of multifactor experiments showing that, in addition to adaptivity, the selection functions greatly affect network performance under various traffic patterns. An analysis of several selection functions in order to evaluate their influence on network performance has been carried out by Martínez et al. [13]. Increases in network throughput (up to 10 percent) and in latency when the network is close to saturation (up to 40 percent) have been observed. Hu and Marculescu [30] 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 [31], Ye et al. present a contention-look-ahead on-chip routing scheme that is similar to [32]. It is a 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). The proposed approach has not been proven to be deadlock free. Differently from the other approaches that focus on output selection, in [33], the authors investigate the impact of input selection and present a contention-aware input selection technique that improves the routing efficiency.

3 Neighbors-on-Path Congestion-Aware Selection

From now on, we consider a wormhole-based switching technique on a mesh topology. Let N be the set of nodes in the network and C the set of communication channels. An adaptive routing function $R : N \times N \rightarrow \wp(C)$, where $\wp(C)$ is the power set of C, supplies a set of alternative output channels toward which all of the flits of the packet should be routed. More precisely, the adaptive routing function can return the following:

- A single output channel toward which to route a packet. In this case, the router has no alternative: It has to send the flits to this channel or wait for it to be released if it has been reserved by another header flit.
- Several output channels to send a packet to, but only one of them is not reserved. In this case, the router could wait for the release of a channel considered to be "better," but generally, to reduce latency in packet delivery due to a long wait for the channel to be released, the packet is routed toward the only channel available.
- Several output channels to send a packet to, but all of them are reserved. In this case, the router has to wait for a channel to be released.
- Several output channels to send a packet to and at least two of them are not reserved. This situation is one of *indecision* for adaptive algorithms.

The availability of alternatives would be an advantage if the choices were made in such a way as to select the channel

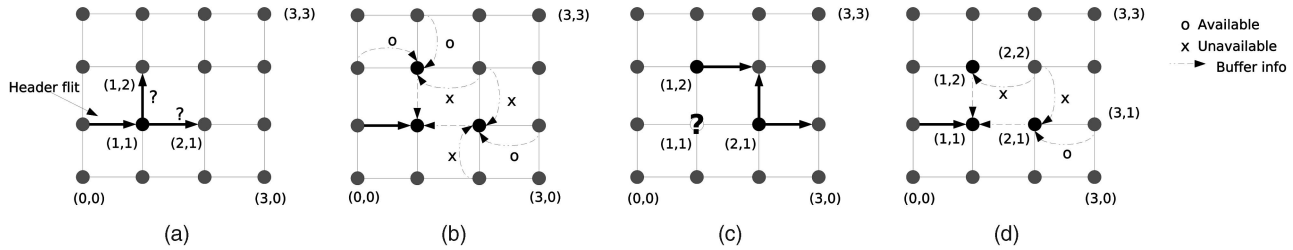


Fig. 3. A visual sketch of some concepts behind the NoP selection. (a) Situation of indecision. (b) Input buffer availability of nonadjacent nodes. (c) Computation of the neighbors that fall on the path. (d) Exploitation of the input buffer status of NoP to make the selection.

that allows the packet to reach its destination more quickly. By means of an example in Section 3.1, we show how our approach, based on the NoP computation, works.

3.1 NoP Algorithm Sketch

Let us first have a quick glance of some of the ideas behind the proposed approach. A formal description of the algorithm performed at each node is given in Section 3.2.

Although the approach is topology-agnostic, for the sake of example, let us consider a mesh topology. Suppose that, moving toward the destination node (3, 3), the header flit of a packet arrived at the west input queue of node (1, 1) and the adaptive routing function returned nodes (2, 1) and (1, 2) as possible output directions, as shown in Fig. 3a. Node (1, 1) has to choose whether to send the header flit (and subsequent data flits) to node (2, 1) or node (1, 2).

The idea is that node (1, 1) would make a better choice if only it had some hints about the input buffer status of nodes that resides beyond nodes (2, 1) and (1, 2), as shown in Fig. 3b. Note that, depending on the final destination node specified in the header, not all of the information represented in the figure is really useful to node (1, 1). In fact, a further step that node (1, 1) needs is to figure out which nodes can really be reached by the header flit once it has been routed toward node (2, 1) or node (1, 2). We thus introduce the concept of NoP: Node (1, 1) computes the routing function considering nodes (2, 1) and (1, 2) as starting nodes, to determine the links toward which they could route the header to reach the destination (3, 3). As result, node (1, 1) learns that nodes (2, 2) and (3, 1) are on a routing path leading to the destination (Fig. 3c), so it will base its choice of the next destination on the buffer availability in these nodes (Fig. 3d). In this way, to decide the next destination for the flit, the router will exclude all buffer availability information from nodes that are not on a possible routing path leading from the current node to the destination. For example, even if it is true that node (1, 2) has two adjacent nodes with available input buffers (nodes (1, 3) and (0, 2), as shown in Fig. 3b), these buffers could never be reached by the header flit considered since the routing function applied at node (1, 2) returned only node (2, 2) (see Fig. 3c). On the other hand, node (2, 1) has an adjacent and reachable node with available input buffer (node (3, 1)), thus resulting in a better choice.

It should be emphasized that this approach does not necessarily guarantee that the west input channel of node (3, 1) will still be available when the packet arrives at node (2, 1). However, an NoP selection strategy tends to prevent the congestion by making the most promising choice in prevision of successive routing paths. The positive effects on overall congestion and saturation point will be discussed in Section 4.

3.2 NoP Algorithm

Let us now give a formal description of the NoP algorithm performed at each node. To understand how it works, we introduce some ad hoc signals required between a node and its adjacent neighbors, as shown in Fig. 4. In particular, for each direction $d \in \{North, South, East, West\}$, we indicate

- $free\_slots\_in[d]$: number of free buffer slots available at the input buffer of the adjacent neighbor along the direction d;
- $free\_slots\_out[d]$: number of free buffer slots available at the input buffer of the current node along the direction d;
- $NoPData\_in[d]$: NoP-specific data computed by neighbor node along the direction d.

A further signal, named $NoPData\_out$, provides NoP-specific data computed by the current node by gathering the input status information from its adjacent neighbor nodes. Referring to Fig. 5, it is assembled by collecting, for each adjacent neighbor (line 3), a pair $(free\_slots, available)$. $NoPData\_out[d].free\_slots$ is the number of free buffer slots available at neighbor's input channel along the direction d and it is provided by the $free\_slots\_in[d]$ input signals (line 4). $NoPData\_out.available[d]$ is a Boolean value representing the reserved/not reserved status of the channel along the direction d (line 5). Such a value is already present in the local reservation table of the router.

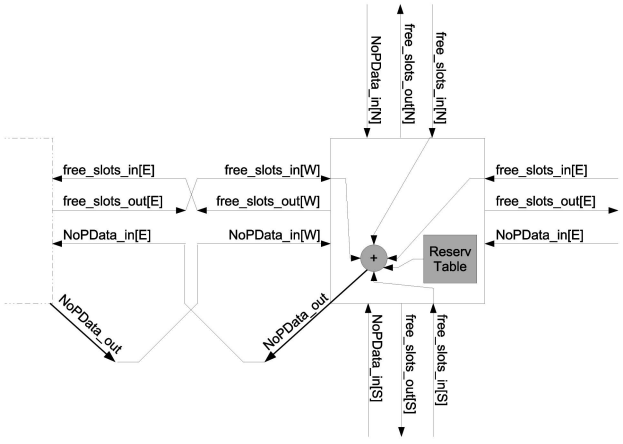


Fig. 4. Ad hoc signals required to implement NoP selection strategy.

```
1 ComputeNoPData(in : free_slots_in[],
2                out: NoPData_out) {
3   for d ∈ {North,South,East,West} {
4     NoPData_out[d].free_slots ← free_slots_in[d]
5     NoPData_out[d].available ← reservation_table[d]
6   }
7 }
```

Fig. 5. NoP data computed at each node.

Let us now analyze how NoP selection algorithm works. Fig. 6 shows the pseudocode of the NoP selection algorithm executed at node n_c. The input parameter is the set of admissible output channels, AOC, provided by the routing function R, i.e., $AOC = R(n_c, n_d)$, where n_d is the destination node of the header flit which has to be routed. The output parameter is the selected output channel $sc \in AOC$. For each candidate output channel (line 3), the current node, n_c, computes the set of NoP nodes (lines 4-5) to investigate the availability of their input buffers (we indicated with $dest(ch)$ the destination node connected to channel ch). Using a score mechanism, the score of a candidate destination is increased for each NoP with available space in a nonreserved input buffer (lines 7-8). Finally, the channel with the highest score is selected (line 11). If more than one channel have the same maximum score, a random choice is made.

As an important note, it should be pointed out that all of these data exchanges do not need to be globally synchronized. We expect that the NoP approach should work better when NoP related data are coherent between nodes and regularly updateed, but this is not a necessary condition for the router to work. If NoP data are never updated, NoP selection simply results in a static routing. On the other hand, using wrong/incoherent NoP data yields a random-like selection strategy. In both cases, the deadlock-free condition is preserved since it only depends on the adaptive routing function used, which we are assuming to be deadlock free.

4 Experiments

4.1 Simulation Environment

In order to perform a complete set of tests involving different traffic scenarios and routing strategies, we developed *Noxim* [34], an open source SystemC simulator of a mesh-based NoC. The whole source code of the environment and the data results presented here are publicly available for further tests and comparisons at the project's homepage [34].

4.2 Traffic Scenarios

We evaluate the NoP selection strategy by using both synthetic and real traffic scenarios. In *uniform* traffic, a node sends the packet to each other node with the same probability. In *transposed* traffic, a node (i, j) only sends packets to a node $(N-1-j, N-1-i)$, where N is the size of the mesh. In the *hotspot* traffic scenario, some nodes are designated as the *hotspot nodes*, which receive hotspot traffic in addition to regular uniform traffic. Given a hotspot percentage h, a newly generated packet is directed to each hotspot node with an additional h percent probability. Finally, a real communication traffic scenario generated by a MultiMedia System [35] (*mms*) is being considered.

```
1  NoP_Select(in : AOC, out: sc) {
2    scores[] ← 0
3    for ch1 ∈ AOC {
4      node1 ← dest(ch1)
5      AOC_neighbor ← R(node1, nd)
6      for ch2 ∈ AOC_neighbor {
7        if NoPData_in[ch1].available[ch2]
8          score[ch1] += NoPData_in[ch1].free_slots[ch2]
9      }
10   }
11   sc ← ch s.t. score[ch] = max(scores[])
12 }
```

Fig. 6. NoP selection algorithm performed at node n_c.

4.3 Evaluation Metrics

We indicate with *packet injection rate (pir)* $(0 < pir \leq 1)$ the rate at which packets are injected into the network. A *pir* of 0.1 (packets/cycle/node) means that each node sends 0.1 packets every clock cycle or that each node sends a packet every 10 clock cycles. The instant at which a packet is injected depends on the distribution of the interarrival times. This instant is chosen on the basis of a negative exponential distribution.

As performance metrics, we choose *throughput* and *delay*. Throughput can be defined in a variety of different ways depending on the specifics of the implementation. Similarly to [36], we define throughput as follows:

$$TP = \frac{Total\ received\ flits}{Number\ of\ nodes \times Total\ cycles},$$

where *Total received flits* refers to the number of whole flits that arrive at their destination nodes, *Number of nodes* is the number of network nodes, and *Total cycles* is the number of clock cycles elapsed between the occurrence of the first message generation and the last message reception. Thus, message throughput is measured as the fraction of the maximum load that the network is capable of physically handling. Delay is defined as the time (in clock cycles) that elapses between the occurrence of a header flit injection into the network at the source node and the occurrence of a tail flit reception at the destination node. We use the average delay, D, as a performance metric as follows:

$$D = \frac{1}{K}\sum_{i=1}^{K} D_i,$$

where K is the total number of messages reaching their destination nodes and D_i is the delay of message i.

The experiments carried out refer to an 8×8 sized NoC. Traffic sources generate 8-flit packets with an exponential distribution, the parameters of which depend on the packet injection rate. The FIFO buffers have a capacity of four flits. Each simulation was initially run for 1,000 cycles to allow transient effects to stabilize and, subsequently, it was executed for 20,000 cycles. To guarantee the accuracy of results, the simulation at each *pir* point has been repeated a number of times, sufficient to obtain an error within three percentage points with a 95 percent confidence interval.

4.4 Experimental Results

For each traffic scenario and algorithm, we will give the average packet delay and throughput with various *pir*. The

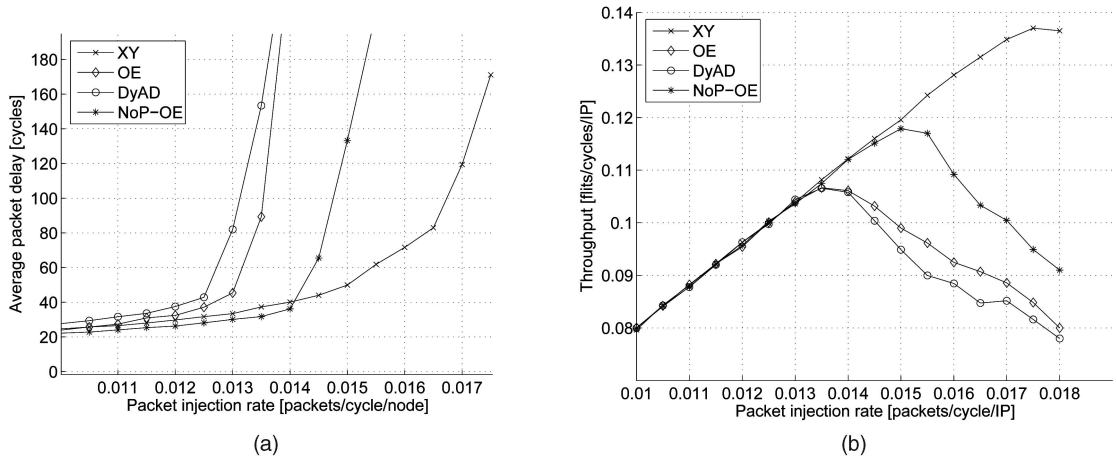


Fig. 7. (a) Delay variation and (b) throughput variation under a *uniform* traffic scenario.

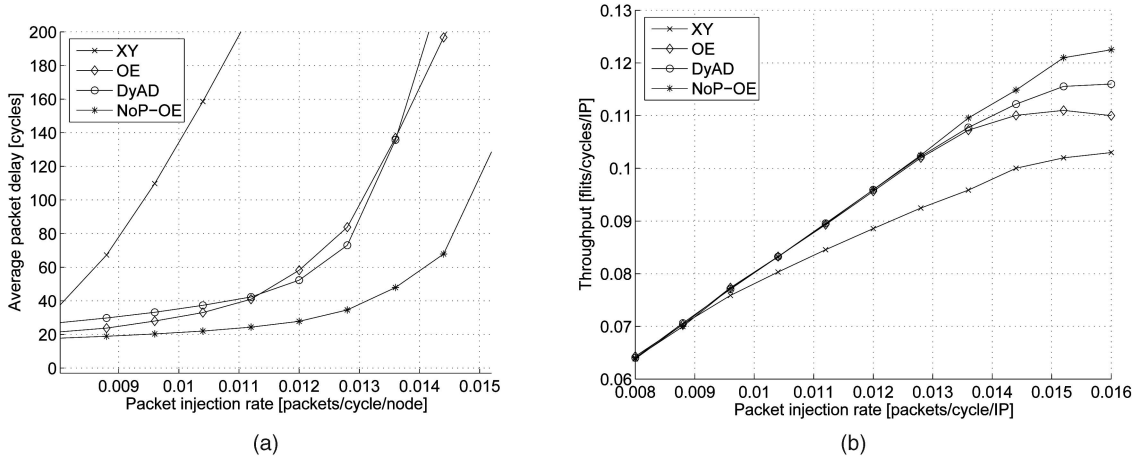


Fig. 8. (a) Delay variation and (b) throughput variation under the *transposed* traffic scenario.

routing algorithms compared are XY, DyAD [30], OE [16] using a selection policy that always selects dimension X when multiple outputs are possible (like in [16]), and OE using the NoP selection strategy (NoP-OE). We consider the choice of applying NoP selection to OE the most natural one since it has been proven to exhibit the best performance among different traffic scenarios [16] and is also at the base of the DyAD approach.

Fig. 7 shows the results obtained when the network has *uniform* traffic. This type of traffic is used in several simulation scenarios, but is not to be found in real applications. As can be seen, the nonadaptivity of the XY algorithm results in a higher saturation point. This result is consistent with other results reported in the literature (e.g., [16], [18], [30]). The main reason for this is that, although the XY algorithm is nonadaptive, it embodies global long-term information about the uniform traffic pattern [18]. By routing packets first along one dimension and then the other, it spreads uniform traffic as evenly as possible across the channels in the long term. The adaptive algorithms, on the other hand, select channels based on local short-term information. This type of decision benefits only the packets in the immediate future, which tend to interfere with other packets [30]. In fact, the selections often produce zigzag paths, which disturb the global long-term evenness of uniform traffic [18]. The result is increased contention and decreased performance at higher *pir* rates [16]. However, although the NoP-OE routing scheme does not perform as well as XY from the saturation viewpoint, it still yields the better average packet delay under non-saturated network conditions.

If we consider a *transposed* traffic scenario (Fig. 8), it is observed that a network adopting XY performs poorly due to its determinism in distributing packets. Under nonsaturated traffic conditions, the NoP-OE routing scheme gives an improvement ranging from 24 percent to 73 percent in average delay as compared to the other approaches.

We now consider two different traffic scenarios with hotspot nodes. Hotspot is considered to be a more realistic traffic model as, in most applications, processes communicate frequently with only a part of the total number of other processes (e.g., memory storage nodes, I/O resources). In the first scenario (*hs-c*), four hotspot nodes are located at the center of the mesh, that is nodes $[(3,3),(4,3),(3,4),(4,4)]$, with 20 percent hotspot traffic. Results thus obtained are shown in Fig. 9. When several traffic flows are directed toward a small subset of nodes, a router adopting a deterministic routing algorithm like XY will be forced to route them toward the same output channel, thus saturating the input queues. In addition, the blockage of a header blocks all of the data flits in the routers along the path. It is thus clear why the network based on XY, in contrast with the uniform

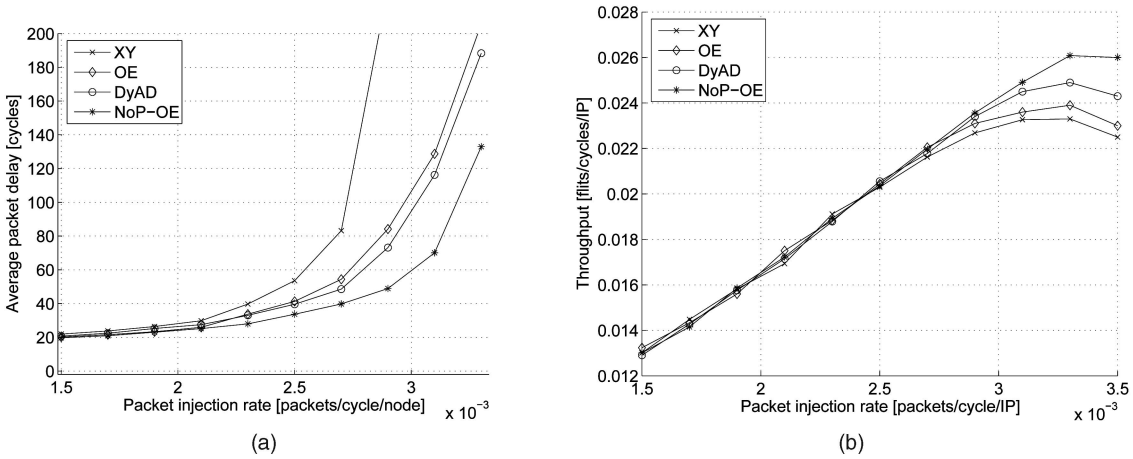


Fig. 9. (a) Delay variation and (b) throughput variation under the *hs-c* traffic scenario.

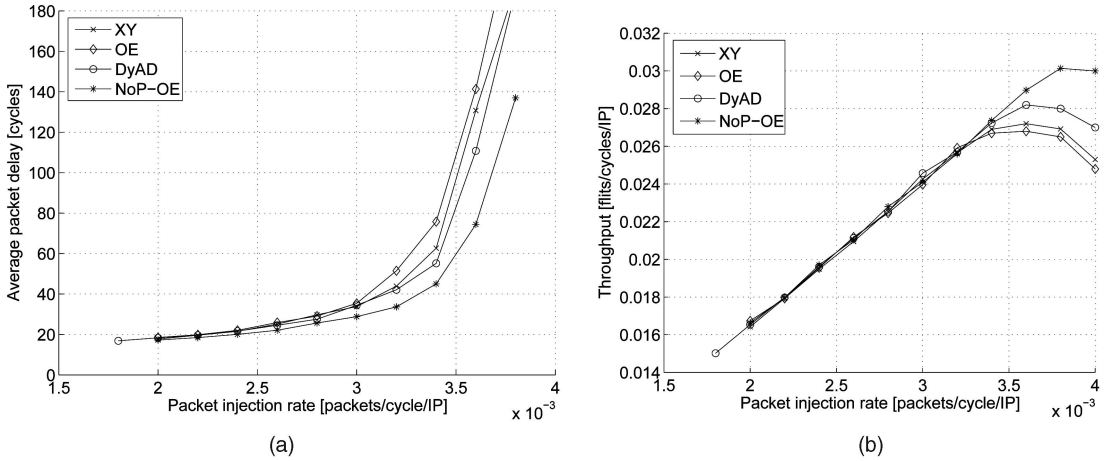


Fig. 10. (a) Delay variation and (b) throughput variation under the *hs-tr* traffic scenario.

traffic scenario, has a lower saturation point than the three adaptive algorithms that can cope with congestion better. Various packets directed toward the same destination can in fact be sent on various alternative output lines. Furthermore, being able to route packets on the basis of congestion information received from neighbors allows NoP to obtain a higher saturation point together with better performance under nonsaturated traffic.

In a second scenario (*hs-tr*), the hotspot nodes are located in the top-right corner of the mesh [nodes (0,6), (0,7), (1,6), (1,7)]. As shown in Fig. 10, the NoP-OE approach still results in an improvement in both the average delay and the saturation point.

Finally, as a more realistic communication scenario, we consider a generic *mms*, which includes an h263 video encoder, an h263 video decoder, an MP3 audio encoder, and an MP3 audio decoder [35]. 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 [37]. For this scenario, we consider self-similar packet injection distributions since it has been observed in the bursty traffic between on-chip modules in typical MPEG-2 video applications [38]. As we can observe from Fig. 11, once again NoP-OE outperforms the other routing algorithms. For a *pir* value of 0.018, where none of the algorithms are saturated, NoP-OE exhibits an average delay of 30 cycles versus 38 cycles of the other algorithms.

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 saturates the network. Saturating injection rate has been calculated as the *pir* at which throughput drops more than 5 percent from the earlier averaged slopes. For each traffic scenario and for each algorithm, Table 1 reports the packet injection rate at the saturation point. It also shows the percent improvement of NoP-OE over XY, OE, and DyAD. On average, NoP-OE outperforms deterministic XY routing by 29 percent, adaptive OE routing by 18 percent, and DyAD by 12 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 percent improvement of NoP-OE over XY, OE, and DyAD. On average, NoP-OE outperforms deterministic XY, adaptive

2. A network is said to start saturating when an increase in applied load does not result in a linear increase in throughput [36].

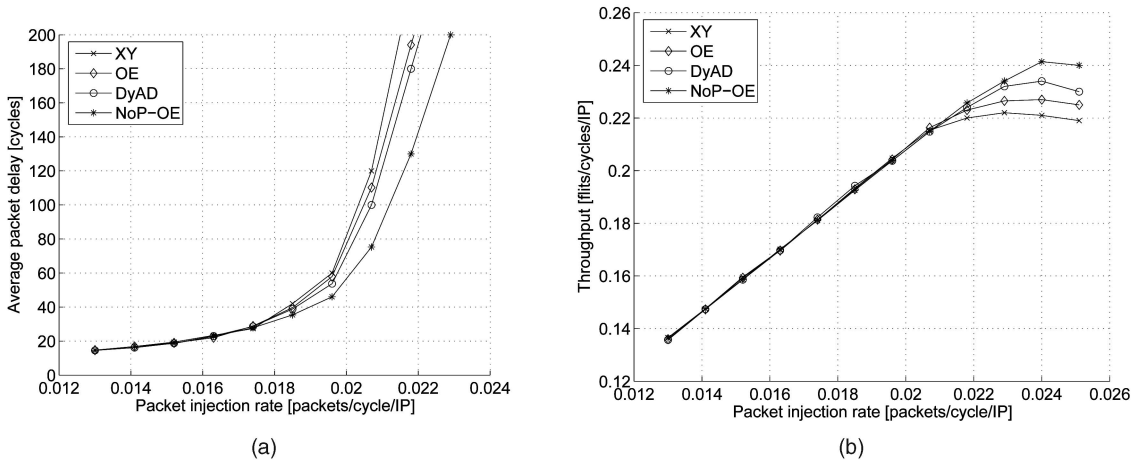


Fig. 11. (a) Delay variation and (b) throughput variation under the *mms* traffic scenario.

TABLE 1
Improvement in the Saturation Point of NoP-OE as Compared to XY, OE, and DyAD for Different Traffic Scenarios

Traffic scenario	Max. *pir* (packets/cycle/IP)				NoP-OE improvement		
	XY	OE	DyAD	NoP-OE	vs. XY	vs. OE	vs. DyAD
Uniform	0.0175	0.0140	0.0140	0.0155	-11%	11%	11%
Transposed	0.0085	0.0130	0.0140	0.0160	88%	23%	14%
hs-c	0.0033	0.0032	0.0035	0.0040	21%	25%	14%
hs-tr	0.0025	0.0027	0.0028	0.0033	32%	22%	18%
Mms	0.0207	0.0224	0.0232	0.0240	16%	7%	3%
Average improvement					29%	18%	12%

TABLE 2
Improvement in Average Delay of NoP-OE as Compared to XY, OE, and DyAD for Different Traffic Scenarios

Traffic scenario	*pir* (packets/cycle/node)	Avg. delay (cycles)				NoP-OE improvement		
		XY	OE	DyAD	NoP-OE	vs. XY	vs. OE	vs. DyAD
Uniform	0.0130	33	45	82	30	10%	34%	63%
Transposed	0.0080	67	24	30	18	73%	24%	40%
hs-c	0.0030	36	35	33	28	22%	20%	15%
hs-tr	0.0025	54	42	41	34	37%	20%	18%
Mms	0.0196	59	58	54	46	22%	21%	14%
Average improvement						33%	24%	30%

OE, and DyAD by 33 percent, 24 percent, and 30 percent, respectively, in terms of average delay.

5 Router Implementation Issues

Fig. 12 shows the schematic of a router for a mesh-based network topology implementing a selection strategy based on NoP. As shown, the top level scheme of the router consists of a set of input buffers, one per direction, plus a further input buffer for the traffic locally generated by the processing/storage element associated to the node. Each input buffer is connected to a block implementing the routing algorithm. A routing algorithm block is used when a routing decision has to be made, that is, when a header flit reaches the input buffer. The destination node (*dst*) provided by the header flit, together with the buffer status information (*NoPDATA_in[]*) coming from adjacent neighbors is used to make the whole routing decision. The output of the routing algorithm block, that is, the selected output channel, is then used by the arbiter in order to set up the crossbar interconnections that bound input traffic flows to the proper output directions.

The content of a routing algorithm block is more accurately shown in the central part of Fig. 12. Each routing algorithm block consists of two components: a first block (*Routing Function*) implementing the adaptive routing function that provides the set of candidate channels (*AOC*) and a second block (*NoP Selection Function*) implementing the NoP selection in order to choose the selected channel (*sc*). As can be seen, the NoP selection block exploits the *NoPData_in[]* provided by adjacent neighbors.

Let us now look inside the *NoP Selection Function* block. A formal description of the NoP selection algorithm has already been given in Fig. 6; thus, we will focus on the architectural elements to be implemented.

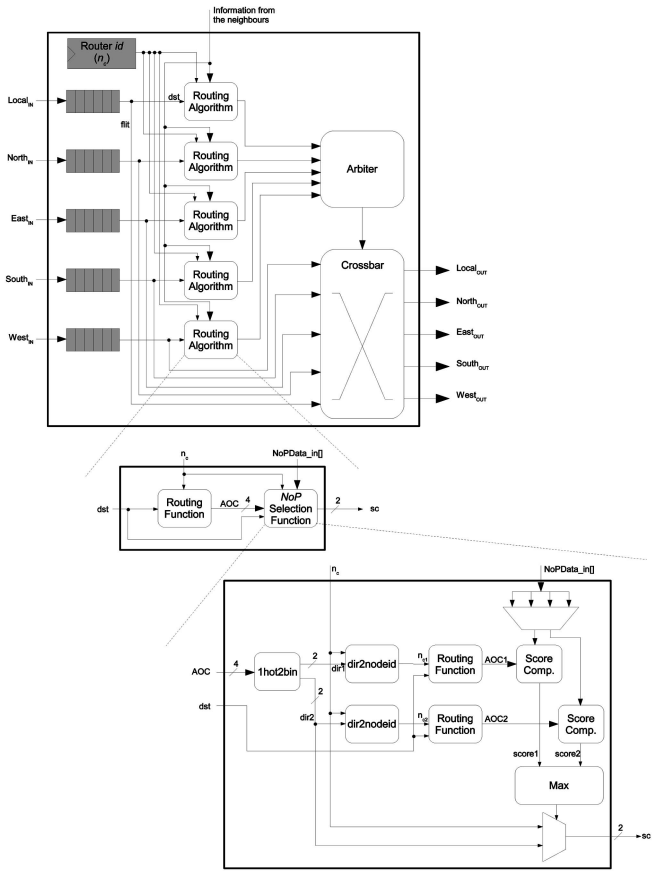


Fig. 12. Schematic of the router. (Top) Top-level view. (Middle) Block implementing the routing algorithm. (Bottom) Block implementing the NoP selection strategy.

First of all, the *1hot2bin* logic decodes the *AOC* set of candidate channels expressed as a one-hot encoding mask of 4 bits into a set of two signals representing the candidate directions. Note that, assuming a 2D mesh topology and a minimal routing function, there can be a maximum of two candidate channels for each routing function block. The *dir2nodeid* logic is necessary to convert a candidate output direction with respect to the current node to the corresponding neighbor node ID. The most relevant part of the *NoP Selection Function* is the presence of two additional *Routing Function* blocks. These blocks are identical to the *Routing Function* block previously described, except that they use the candidate nodes n_{c1} and n_{c2} as current nodes. As seen in Fig. 6, this is necessary to compute the *AOC_neighbor* representing the set of NoP nodes. Finally, each set of *AOC_neighbor* is used to select the proper subset of *NoPData_in* that must be accounted for when computing the score of each candidate channel. The channel associated to the highest score is the final output of the whole *NoP Selection Function* block, that is, the *sc*.

It is evident from the implementation presented in Fig. 12 that an NoP router requires some additional blocks in order to perform the selection phase. In particular, two *Routing Function* blocks in cascade are present in the critical path since a first routing function is computed for the current node n_c and a second routing computation must be performed assuming n_{c1} and n_{c2} as current nodes. However, the architecture introduced in Fig. 12 is only one of the possible router implementations. Different optimizations could be applied, depending on the particular case considered. For instance, when considering a mesh-based topology, the cascade of the two *Routing Function* blocks could be avoided. This can be accomplished by performing the first routing phase, with current node n_c, in parallel with the routing function of the selection stage. In this case, instead of waiting for the computation of candidate nodes n_{c1} and n_{c2}, three *Routing Function* blocks could speculatively compute the routing assuming as current nodes all three possible output directions. Note that a further *Routing Function* block would be required, thus leading to additional area/power increase.

A further point of interest could be the generalization of the approach in order to look at neighbors beyond the neighbors of adjacent nodes. However, for the traffic patterns used and the network topology we are investigating, we found that this results in more outdated buffer status information without any gain in performance. In other words, there is no trade-off since such NoP-related data increase the probability of being unreliable while, at the same time, the complexity of the router increases nonlinearly. For example, even considering the area optimized version of the router as shown in Fig. 12, considering N hops ahead would require $\sum_{i=1}^{N} 2^i$ extra routing blocks.

5.1 Area Overhead

In an NoC-based SoC, the requirement is that routers should not consume a large fraction of silicon area compared to the IP blocks. We have designed in VHDL four routers based on XY, OE, DyAD, and NoP-OE, respectively. We have synthesized the designs using Synopsys Design Compiler and mapping them onto a 0.13 μm library from MOSAID. We found that, within the router, the buffer area significantly dominates the logic [39]. The buffer area, in turn, largely depends on the flit size. For a flit size of 64 bits and FIFO buffers with a capacity of four flits, we found that the area overhead due to the NoP is less than 8 percent, 6 percent, and 5 percent as compared to XY, OE, and DyAD, respectively.

5.2 Extra Wiring Cost

The implementation of the NoP selection strategy requires some extra control wires, as shown in Fig. 4. Let S be the FIFO input buffer size. The `free_slots_in[i]` and `free_slots_out[d]` ports ($d \in \{North, South, East, West\}$) are $\lceil \lg_2 S \rceil$ bits wide each. The `NoPData_in[d]` input ports ($d \in \{North, South, East, West\}$) and `NoPData_out` are $4 \times (1 + \lceil \lg_2 S \rceil)$ bits wide each. The factor 4 takes account of all four output directions. For each direction d, the quantity inside the brackets counts the number of bits necessary to store the status information of the output channel along direction d. Such information concerns the availability of the output channel d and the number of free slots in the input buffer connected to d. Overall, the total number of wires required to support the NoP selection strategy is

$$NoP_{wires} = 8 \times \lceil \lg_2 S \rceil + 4 \times 4 \times (1 + \lceil \lg_2 S \rceil) + 4 \times (1 + \lceil \lg_2 S \rceil). \quad (1)$$

Considering the data used in the experiments (FIFO buffers with a capacity of four flits and flit size of 64 bits), we have

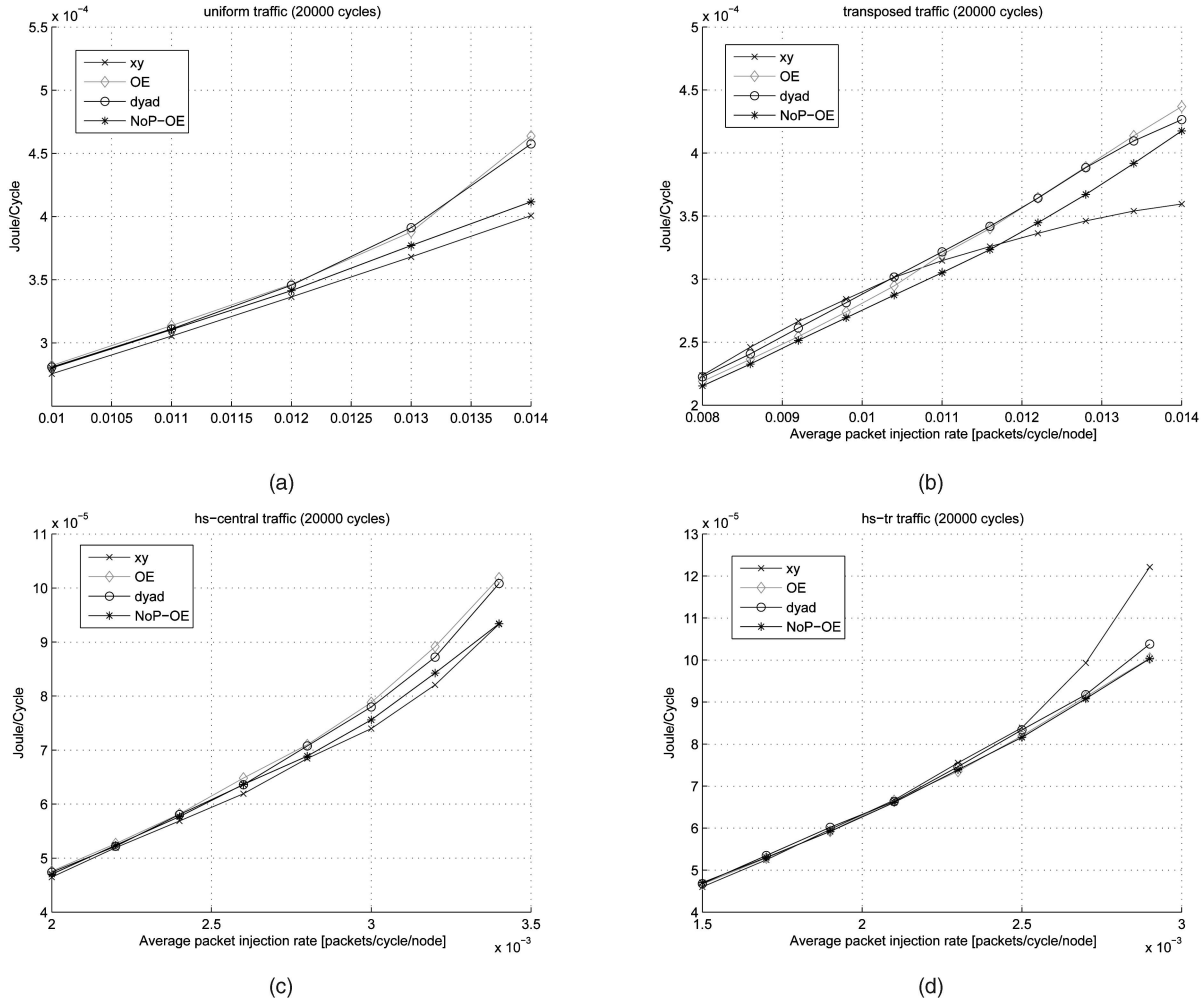


Fig. 13. Average power dissipation under different traffic scenarios.

$NoP_{wires} = 76$ wires, which is less than 11 percent of the total wiring cost.

5.3 Scalability and Generality Issues

The NoP selection strategy is scalable and general with respect to the network size and the network topology, respectively. As regards the scalability property, it should be pointed out that the router architecture described so far is network size-agnostic. In fact, both the number and the size of the input/output ports along with the number and the complexity of the basic blocks forming the router architecture sketched in Fig. 12 do not depend on the size of the network. From the side of the generality, the proposed selection strategy can be applied to any network topology without any significant variation of the structure shown in Fig. 12. However, in this case, the cost of the block implementing the NoP selection strategy is a linear function of the number of output ports. In fact, as discussed above, the block implementing the routing function must be instantiated a number of times equal to the size of the maximum admissible output set allowed by the underlying routing function. The extra wiring cost is also a function of the number of output ports. In this case, term 4 that appears in (1) should be replaced with the number of output ports.

5.4 Energy/Power Analysis

In order to perform an energy analysis of the designs, we can observe that energy dissipation of NoC fabrics arises from two different sources: 1) the router blocks, which include the buffers, and 2) interrouter wire segments. We determine the energy dissipated in a router by running Synopsys Design Power on the gate-level netlist of the router (including the FIFO buffers) when it is stimulated by different random input data streams. The average energy dissipated by a flit for a hop switch was estimated as being 0.151, 0.178, 0.182, and 0.189 nJ for XY, OE, DyAD, and NoP-OE, respectively. We assumed the tile size to be $2\ \mathrm{mm} \times 2\ \mathrm{mm}$ and that the tiles were arranged in a regular fashion on the floorplan. The load wire capacitance was set to 0.50 *fF* per micron, so, considering an average of 25 percent switching activity, the amount of energy consumed by a flit for a hop interconnect is 0.384 nJ. We used these values to backannotate our functional simulator in such a way as to obtain an estimation of the total energy consumption.

The energy comparison was performed from two different points of view: the average power dissipation and the total energy required to drain a fixed amount of traffic. It should be pointed out that both aspects are essential for an energy-aware design of an NoC router.

TABLE 3
Energy Consumption to Drain 10 Mbytes of Data for Different Traffic Scenarios and Different *pirs*

Scenario	*pir*	Energy (mJ) XY	OE	DyAD	NoP-OE
Uniform	.010	1.68	1.87	1.86	1.98
	.013	1.77	2.24	3.72	2.08
	.014	1.85	3.92	7.43	2.08
	.015	2.02	-	-	3.57
	.016	3.37	-	-	-
Transpose	.008	1.48	1.77	1.81	1.88
	.009	3.71	1.85	1.9	1.9
	.012	-	4.42	3.62	2.26
	.013	-	7.07	5.43	2.82
	.014	-	-	-	5.65
hs-c	.0015	0.79	0.93	0.95	0.99
	.003	1.43	1.87	1.53	1.39
	.0032	1.58	2.05	1.62	1.59
	.0033	3.17	3.92	3.82	1.98
	.0034	5.54	-	-	3.96
hs-tr	.0015	1.98	2.33	2.39	2.38
	.0023	5.94	3.27	4.77	3.33
	.0025	-	4.67	9.54	4.04
	.0028	-	5.13	11.93	4.75
	.0031	-	11.67	14.31	9.51
mms	.010	0.69	0.71	0.73	0.79
	.016	1.04	0.99	0.94	0.87
	.018	1.25	1.06	1.02	0.95
	.019	2.08	1.42	1.38	1.27
	.020	2.78	2.13	2.1	1.98

Power dissipation, which is linked to the amount of heat the system is subjected to, is a fundamental element for aspects such as packaging, which directly affect the final cost of implementing the system. Total energy consumption, on the other hand, could prove to be a decisive factor in battery-powered mobile devices.

For what concerns the first aspect, that is, power dissipation, in Fig. 13, we report the average Joule/cycle consumed for each traffic scenario at different *pir* rates. As can be expected, as the injection rate increases, the average power dissipated increases. We do not show values beyond the saturation point since a further increase in *pir* would not result in more traffic injected in the network and packets simply remain unqueued at the network interface of the source node. We can observe that the NoP average power dissipation trend does not significantly deviate from other router implementations. Even if, in some cases, it results in a minor increase as compared to static XY, it still remains slightly below the other adaptive solutions for all ofthe traffic scenarios. This is not in contrast with the higher flit hop switch energy previously reported for NoP simply because the flit switch is not the only source of power dissipation in NoCs. For example, even if invoking the NoP selection function implies additional energy consumption, this may be balanced by a minor usage of FIFO buffers due to better congestion avoidance.

This is even more evident if we analyze the second perspective of our analysis, that is, the total energy needed to drain a fixed volume of traffic. Table 3 summarizes, for each traffic scenario, the overall energy required to consume a fixed workload of 10 Mbytes at five representative packet injection rates. The first *pir* value represents a very low traffic load where none of the algorithms are saturated. The subsequent four *pir* values have been chosen in order to emphasize the effects of congestion on the overall energy consumption. A missing value in Table 3 simply reflects a condition of full saturation. In these cases, the fixed workload of 10 Mbytes cannot be processed in a reasonable amount of time and the comparison of energy values would make no sense.

At very low *pir* rates, that is, the first row of each traffic scenario, NoP-OE results in a negligible energy overhead as compared to the other adaptive algorithms, whereas the overhead with respect to XY is more evident, ranging from 18 percent to 27 percent. This simply means that, at low traffic rates, the power dissipation overhead of adaptive routers is not balanced by any performance improvement. On the other hand, when higher values of *pir* in Table 3 are being considered, the NoP-OE router may result in better energy usage since it is less affected by congestion-related issues. In particular, in most of the cases, under heavy traffic workloads, the better performance of NoP-OE may balance its power dissipation overhead, thus resulting in an overall saving of energy consumption. Once again, this excludes the uniform traffic scenario as a consequence of the better performance of the static XY routing for this type of traffic, as already discussed in Section 4.4.

6 Conclusions

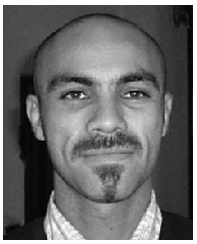
In this paper, we have proposed a selection strategy that introduces the concept of NoP to improve the performance of an NoC. The aim of our algorithm is to exploit the situations of indecision that can occur in adaptive wormhole routing. The approach, which is general in nature, has been applied to the OE and compared to both deterministic and adaptive routing algorithms. The simulations performed showed, in most of the cases, an improvement in average delay and energy consumption, especially under heavy traffic workloads.

The comparison of routing and selection strategies strictly depends on the traffic scenario that populates the NoC. Future developments include different interesting issues, such as the analysis of different topologies and traffic patterns. A further and essential step will be the mapping of real applications on NoCs to assess the validity of the proposed selection strategy for applications whose performance at higher packet injection rates is critical.

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] "International Technology Roadmap for Semiconductors—Interconnect," Semiconductor Industry Assoc., 2006.
[3] "On-Chip Bus Attributes Specification Version 1," VSI Alliance, http://www.vsi.org/, Sept. 2001.
[4] "The CoreConnect Bus Architecture," IBM, http://www.ibm.com/, 2008.
[5] "AMBA Specification," ARM, http://www.arm.com/, May 1999.
[6] "SoC Bus Architecture," Palmchip, http://www.palmchip.com/, 2008.
[7] "WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores," Silicore, http://www.silicore.net/, Sept. 2002.
[8] "SiliconBackplane III MicroNetwork IP," Sonics, http://www.sonicsinc.com/, 2008.
[9] *Networks on Chip*, A. Jantsch and H. Tenhunen, eds., ch. 1. Kluwer Academic, 2003.

[10] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. ACM/IEEE Design Automation Conf.*, pp. 684-689, 2001.
[11] L. Schwiebert and R. Bell, "Performance Tuning of Adaptive Wormhole Routing through Selection Function Choice," *J. Parallel and Distributed Computing*, vol. 62, no. 7, pp. 1121-1141, July 2002.
[12] W.C. Feng and K.G. Shin, "Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks," *Proc. 11th Int'l Conf. Supercomputing*, pp. 132-139, 1997.
[13] J.C. Martínez, F. Silla, P. López, and J. Duato, "On the Influence of the Selection Function on the Performance of Networks of Workstations," *Proc. Int'l Symp. High Performance Computing*, pp. 292-299, 2000.
[14] P. Mohapatra, "Wormhole Routing Techniques for Directly Connected Multicomputer Systems," *ACM Computing Surveys*, vol. 30, no. 8, pp. 374-410, Sept. 1998.
[15] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann, 2002.
[16] 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.
[17] 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.
[18] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *J. ACM*, vol. 41, no. 5, pp. 874-902, Sept. 1994.
[19] 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.
[20] 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.
[21] E. Bolotin, A. Morgenshtein, I. Cidon, and A. Kolodny, "Automatic and Hardware-Efficient SoC Integration by QoS Network on Chip," *Proc. IEEE Int'l Conf. Electronics, Circuits and Systems*, Dec. 2004.
[22] R. Holsmark and S. Kumar, "Design Issues and Performance Evaluation of Mesh NoC with Regions," *Proc. IEEE Norchip*, pp. 40-43, Nov. 2005.
[23] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY—A Proximity Congestion-Aware Deadlock-Free Dynamic Routing Method for Networks on Chip," *Proc. ACM/IEEE Design Automation Conf.*, pp. 849-852, July 2006.
[24] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A Methodology for Design of Application Specific Deadlock-Free Routing Algorithms for NoC Systems," *Proc. Int'l Conf. Hardware-Software Codesign and System Synthesis*, pp. 142-147, Oct. 2006.
[25] M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, and C.P. Thacker, "Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links," Technical Report 59, Digital Equipment, Apr. 1990.
[26] A. Jouraku, M. Koibuchi, and H. Amano, "L-Turn Routing: An Adaptive Routing in Irregular Networks," Technical Report 59, IEICE, Apr. 2001.
[27] L. Cherkasova, V. Kotov, and T. Rokicki, "Fibre Channel Fabrics: Evaluation and Design," *Proc. Hawaii Int'l Conf. System Sciences*, pp. 53-58, 1996.
[28] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Scheme for Networks of Workstations," *Proc. Int'l Symp. High Performance Computing*, pp. 260-267, 2000.
[29] 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. Int'l Parallel and Distributed Processing Symp.*, Apr. 2006.
[30] J. Hu and R. Marculescu, "DyAD—Smart Routing for Networks-on-Chip," *Proc. ACM/IEEE Design Automation Conf.*, pp. 260-263, June 2004.
[31] T.T. Ye, L. Benini, and G.D. Micheli, "Packetization and Routing Analysis of On-Chip Multiprocessor Networks," *J. System Architectures*, vol. 50, nos. 2-3, pp. 81-104, 2004.
[32] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load Distribution with the Proximity Congestion Awareness in a Network on Chip," *Proc. Design, Automation and Test in Europe*, pp. 1126-1127, 2003.
[33] D. Wu, B.M. Al-Hashimi, and M.T. Schmitz, "Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection," *Proc. Asia and South Pacific Design Automation Conf.*, pp. 36-41, 2006.
[34] "Noxim: Network-on-Chip Simulator," http://sourceforge.net/projects/noxim, 2008.
[35] 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.
[36] 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.
[37] 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.
[38] G. Varatkar and R. Marculescu, "Traffic Analysis for On-Chip Networks Design of Multimedia Applications," *Proc. ACM/IEEE Design Automation Conf.*, pp. 510-517, June 2002.
[39] P.P. Pande, C. Grecu, and I. Ivanov, "High-Throughput Switch-Based Interconnect for Future SoCs," *Proc. IEEE Int'l Workshop System-on-Chip for Real-Time Applications*, pp. 304-310, 2003.



Giuseppe Ascia received the Laurea degree in electronic engineering and the PhD degree in computer science from the Università di Catania, Italy, in 1994 and 1998, respectively. In 1994, he joined the Institute of Computer Science and Telecommunications at the Università di Catania, where he is currently an associate professor. His research interests are soft computing, VLSI design, hardware architectures, and low-power design.

Vincenzo Catania received the Laurea degree in electrical engineering from the Università di Catania, Italy, in 1982. Until 1984, he was responsible for testing microprocessor system at STMicroelectronics, Catania. Since 1985, he has participated in research on computer networks at 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.

Maurizio Palesi received the Laurea and PhD degrees in computer engineering from the Università di 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. He is a member of the IEEE Computer Society.

Davide Patti received the Laurea and PhD degrees in computer engineering from the University of Catania, Italy, in 2003 and 2007, respectively. His research focuses on platform-based system design, design space exploration, low-power techniques for embedded systems, and Network-on-Chip architectures. He is a member of the IEEE Computer Society.