A New Selection Strategy for On-Chip Networks[*]

Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, Davide Patti
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
Università di Catania, Italy

September 20, 2006

Abstract

Efficient and deadlock-free routing is critical to the performance of networks-on-chip. In this paper we present an approach that can be coupled to any adaptive routing algorithm to improve the performance with a minimal overhead on area and energy consumption. The proposed approach introduces the concept of *Neighbors-on-Path* to exploit the situations of indecision occurring when the routing function returns several admissible output channels. A selection strategy is 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. Performance evaluation is carried out by using a flit-accurate simulator on traffic scenarios generated by both synthetic and real applications. Results obtained show how the proposed selection policy applied to the Odd-Even routing algorithm outperforms other deterministic and adaptive routing algorithms both in average delay and energy consumption.

1 Introduction

The International Technology Roadmap for Semiconductors [10] foresees that the on-chip interconnection system will represents the limiting factor for performance and power consumption in next generation systems-on-a-chip (SoCs). Network-on-chip (NoC) architectures [3] represent a fi rst answer to cope with the complexity and requirements of such a systems.

The NoC architectural topology most frequently referred to can be represented by an $m \times n$ mesh [7]. Each tile of the mesh contains a *resource* and a *router*. Each router is connected to a resource and the four adjacent routers.

[*]This document is available from the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni at the Università degli Studi di Catania, V.le Andrea Doria 6—I95125 Catania, Italy, as technical report DIIT-TR-01-060920, September 2006. Please, use the technical report number when you reference this document. Authors' addresses: G. Ascia, V. Catania, M. Palesi, and D. Patti Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, V.le Andrea Doria, 6, 95125 Catania, Italy, {gascia,vcatania,mpalesi,dpatti}@diit.unict.it.

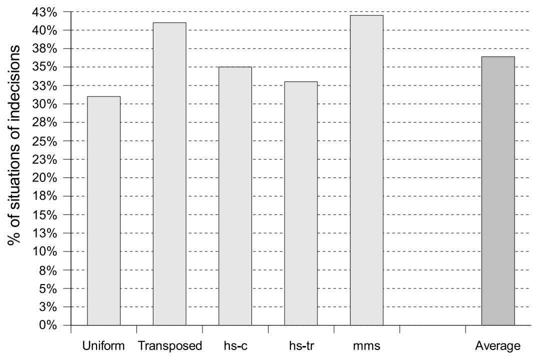

Figure 1: Percentage of situations of indecision for different traffi c scenarios.

Routers can be classifi ed as *deterministic* and *adaptive*. Given a source and a destination, in deterministic routing the path is completely determined. In adaptive routing, the path taken by a particular packet depends on dynamic network conditions such as the presence of faulty or congested channels. The main advantage of an adaptive routing algorithm is the possibility of routing packets along alternative paths in order to avoid congestion areas.

Wormhole switching [8] has emerged as the most widely adopted switching technique for NoC routers. 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. Unfortunately, blockage of the header flit of a packet blocks all the remaining flits of the packet along the established path. The blocked header flit will have to wait for all 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 channel established by the router contains at least two *non reserved*[1] output channels, the router has to choose one of them. This phase is usually referred as *selection policy*. For different traffi c scenarios, which will be described in Section 4, Figure 1 shows the percentage of *situations of indecision*, for a 8×8 NoC, with routers implementing the Odd-Even routing algorithm [2] and eight flits input buffers. With the term "situations of indecision" we indicate the relationship, expressed as a percentage, between the number of times a router is able to route a packet towards alternative channels (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% which represents an important optimization opportunity.

The reason for the development of a selection strategy is to solve situations of indecision. The main aim is to allocate 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 inter-node information can

[1]An output channel is said to be *non reserved* if a header fli t belonging to another packet has not reserved it to transmit the fli t making up the packet.

only be exchanged through packets, on-chip networks can take advantage of dedicated control wires to transmit data between routers. This makes easier to collect useful informations about congestion-related aspects such as buffer status of specific nodes. The focus of this paper is 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*, a set of nodes that can be computed for a given node and a specific header flit. In the following sections we show how these nodes are computed and how associated buffer status can be used to prevent congestion.

2 Related Work

Several efforts have been done attempting to improve the performance of routing strategies in Network-on-Chip. In [4] Glass and Ni present a model for designing wormhole routing algorithms. It is based on analysis of the directions in which packets can turn in a network and the cycles that the turn can form. The idea is to prohibit a subset of all the possible turns so as to avoid deadlock. The main problem with this approach is unfairness in the degree of adaptivity: Only one subset of source-destination pairs enjoys total adaptivity, whereas the others will route packets over a single minimum path. The routing algorithm known as *Odd-Even* proposed by Chiu in [2] considerably attenuates these problems, distributing the degree of adaptivity in a more uniform way. Deadlock is avoided by restricting the locations where certain types of turn can occur rather than by prohibiting turns. The observation that deterministic routing is more efficient than adaptive routing with low workloads was exploited by Hu and Marculescu in [5] to define a general routing methodology known as DyAD. This algorithm is the combination of a deterministic routing algorithm and an adaptive routing algorithm. The router can switch between these two routing modes based on the network's congestion conditions. Another selection strategy for NoCs, called *look-ahead*, was proposed by Ye et al. [12]. Although interesting, it does not guarantee the deadlock-free condition which we consider as key issue in Network-on-Chip routing.

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 processing nodes in the network and C the set of communication channels. An adaptive routing function $R : N \times N \to \wp(C)$, where $\wp(C)$ is the power set of C, supplies a set of alternative output channels toward which all the flits of the packet should be routed on. More precisely, the adaptive routing function can return:

- A single output channel to route a packet towards. 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.

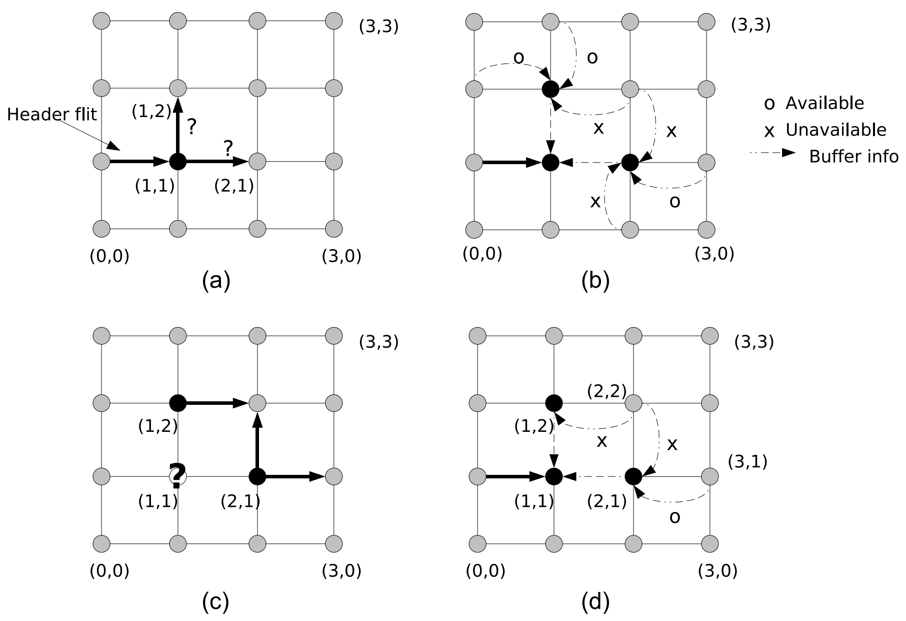

Figure 2: Neighbors-on-Path congestion-aware routing algorithm explaination. (a) Node $(1,1)$ has to choose between two possible candidates. (b) Nodes $(2,1)$ and $(1,2)$ receive information about buffer availability from their neighbors. (c) Node $(1,1)$ fi gure out which output channels would return routing function applied at nodes $(1,2)$ and $(2,1)$. (d) Node $(1,1)$ exploit buffer availability of its Neighbors-on-Path.

- 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 towards 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 choice were made in such a way as to select the channel which allows the packet to reach its destination more quickly. By means of an example, in the following subsection we show how our approach, based on the *Neighbors-on-Path* (NoP) computation, works.

3.1 NoP Algorithm Sketch

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

Suppose that, moving towards its destination, the header flit of a packet arrived at the 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 Figure 2(a). 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 Figure 2(b). Note that, depending on the final destination node specified in the header, not all the information represented in the figure is really useful to the node $(1,1)$. In fact, a further step that node $(1,1)$ needs is to figure out which nodes can be really reached by the header once it has been routed towards 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 towards which they could route the header flit. As result, node $(1,1)$ learns that nodes $(2,2)$ and $(3,1)$ are on a routing path leading to the destination [Figure 2(c)], so it will base its choice of the next destination on buffer availability in these nodes [Figure 2(d)]. 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. It is true that node $(1,2)$ has two adjacent nodes with available input buffers, but these buffers could never be reached by the header flit considered. On the other hand, node $(2,1)$ has an adjacent and reachable node with available input buffer [node $(3,1)$], thus resulting a better choice.

It should be emphasized that this approach does not necessarily guarantee that input channel of node $(3,1)$ will still be available when the packet arrives at the node $(2,1)$. However, a 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 Figure 3. 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;

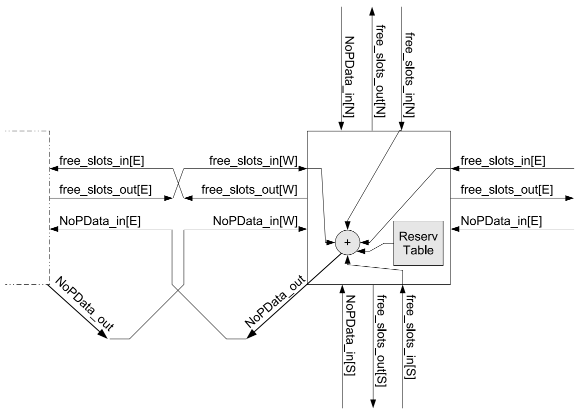

Figure 3: Ad-hoc signals required to implement NoP selection strategy.

```
1 ComputeNoPData(in : $free\_slots\_in[]$,
2                out : $NoPData\_out$) {
3   for $d \in \{North, South, East, West\}$ {
4     $NoPData\_out[d].free\_slots \leftarrow free\_slots\_in[d]$
5     $NoPData\_out[d].available \leftarrow reservation\_table[d]$
6   }
7 }
```

Figure 4: NoP data computed at each node.

- $free\_slots\_out[d]$: 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 Figure 4, 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.

Let us now analyse how NoP selection algorithm works. Figure 5 shows the pseudo-code of the NoP selection algorithm executed at the 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

```
1  NoP_Select(in : $AOC$ , out: $sc$) {
2    $scores[] \leftarrow 0$
3    for $ch_1 \in AOC$ {
4      $node_1 \leftarrow dest(ch_1)$
5      $AOC\_neighbor \leftarrow R(node_1, n_d)$
6      for $ch_2 \in AOC\_neighbor$ {
7        if $NoPData\_in[ch_1].available[ch_2]$
8          $score[ch_1] += NoPData\_in[ch_1].free\_slots[ch_2]$
9      }
10   }
11   $sc \leftarrow ch$ $st$ $score[ch] = max(scores[])$
12 }
```

Figure 5: NoP selection algorithm performed at node n_c.

$sc \in AOC$. For each candidate output channel (line 3), the current node n_c computes the set of neighbors-on-path nodes (line 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 neighbor-on-path with available space in a not reserved input buffer (lines 7-8). Finally, the channel with the higher score is selected (line 11).

As a further note it should be pointed out that all these data exchanges do not need to be globally synchronized. We expect that NoP approach should work better when NoP related data are coherent between nodes and regularly up-to-date, 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.

4 Experiments

4.1 Traffic Scenarios

We evaluate the NoP selection strategy by using both synthetic and real traffi c scenarios. In the *Uniform* traffi c a node sends the packet to each other node with the same probability. In the *Transposed* traffi c, 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 *Hot-spot* traffi c scenario some nodes are designated as the *hot spot nodes*, which receive hot spot traffi c in addition to regular uniform traffi c. Given a hot spot percentage h, a newly generated packet is directed to each hot spot node with an additional h percent probability. Finally, as a real traffi c scenario we consider the communication traffi c generated by a MultiMedia System [6] (*mms*).

4.2 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 processing element (PE) sends 0.1 packets every clock cycle, or that each PE sends a packet every 10 clock cycles. The instant at which a packet is injected depends on the distribution of the interarrival times.

A metric commonly used to evaluate the performance of a network is the *average packet delay*. In a wormhole network packet delay covers the time interval between the instant at which the header flit is sent by the source and the instant at which the last flit is received at the destination node, including queuing times at the source. Another index of the quality of a routing algorithm is the *saturation point* which is the injection rate at which increase in applied load does not result in linear increase in throughput. If the injection rate is close to or above saturation point, the performance of the system deteriorates rapidly. Routing algorithms are required to be characterized by a high saturation point.

Experiments were carried out on a NoC simulation platform developed in SystemC. The size of the NoC was 8×8. Traffi c sources generate 8-flits packets with an exponential distribution, the parameters of which depend on the packet injection rate. The FIFO buffers have a capacity of 4 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 suffi cient to obtain an error within three percentage points with a 95% confi dence interval.

4.3 Results

For each traffi c scenario and algorithm, we will give the average packet delay (expressed in clock cycles) with various *pir*. The routing algorithms compared are XY, DyAD, Odd-Even (OE) and NoP applied on Odd-Even (NoP-OE) routing. We consider the choice of applying NoP selection to Odd-Even the most natural one, since it has been proved to exhibit the best performance among different traffi c scenarios [2] and is also at the base of the DyAD approach.

Figure 6 shows the results obtained when the network has *Uniform* traffi c. This type of traffi c is used in several simulation scenarios but is not to be found in real applications. As can be seen, the non adaptivity of the XY algorithm results in a higher saturation point. The main reason for this is that the XY algorithm is based on long-term global information [4]. Routing packets along one dimension and then the other, the algorithm distributes the traffi c 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 which hinder the uniform distribution of traffi c causing a greater channel contention that deteriorates performance at higher pir rates. However, although the NoP-OE routing scheme does not perform as well as XY

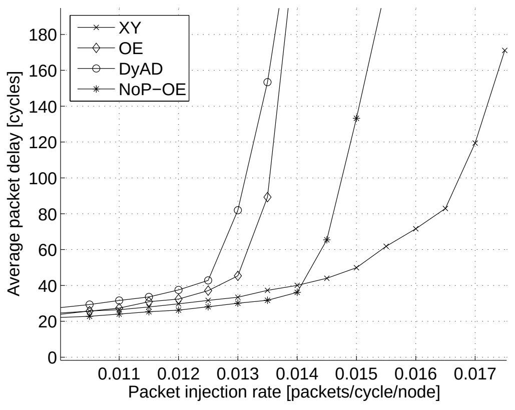

Figure 6: Delay variation under *Uniform* traffi c scenario.

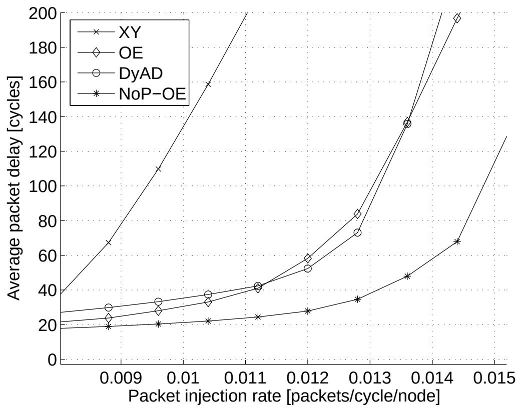

Figure 7: Delay variation under *Transposed* traffi c scenario.

from the saturation view-point, it still yelds the better average packet delay under non-saturated network conditions.

If we consider *Transposed* traffi c scenario (Figure 7), it is observed that a network adopting XY performs poorly due to its determinism in distributing packets. Under non-saturated traffi c conditions the NoP-OE routing scheme gives about a 50% improvement in average delay as compared to the other adaptive approaches.

We now consider two different traffi c scenarios with hotspot nodes. Hotspot is considered to be a more realistic traffi c 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 fi rst scenario (*hs-c*), four hot spot nodes are located at the center of the mesh, that is nodes [$(3,3)$, $(4,3)$, $(3,4)$, $(4,4)$)], with 20% hot spot traffi c. Results thus obtained are shown in Figure 8. When several traffi c flows are directed towards a small subset of nodes, a router

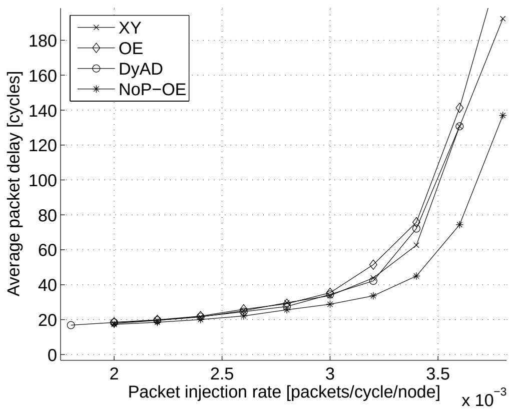

Figure 8: Delay variation under *hs-c* traffi c scenario.

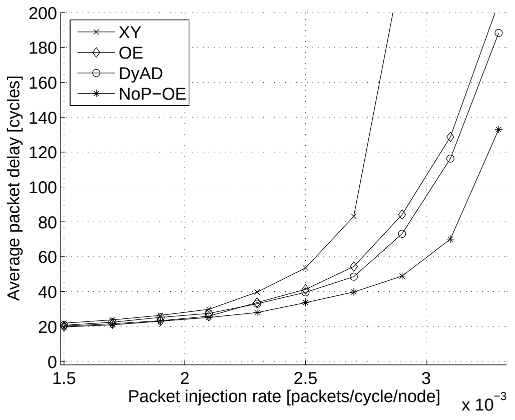

Figure 9: Delay variation under *hs-tr* traffi c scenario.

adopting a deterministic routing algorithm like XY will be forced to route them towards the same output channel, thus saturating the input queues. In addition, the blockage of a header blocks all the data flits in the routers along the path. It is thus clear why the network based on XY, in contrast with uniform traffi c scenario, has a lower saturation point than the three adaptive algorithms which can cope with congestion better. Various packets directed towards 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 an higher saturation point together with a better performance under non-saturated traffi c.

In a second scenario, (*hs-tr*), the hot spot nodes are located at the top-right corner of the mesh [nodes $(0,6)$, $(0,7)$, $(1,6)$, $(1,7)$]. As shown in Figure 9, although less evident than the previous hot-spot scenario, NoP-OE approach still

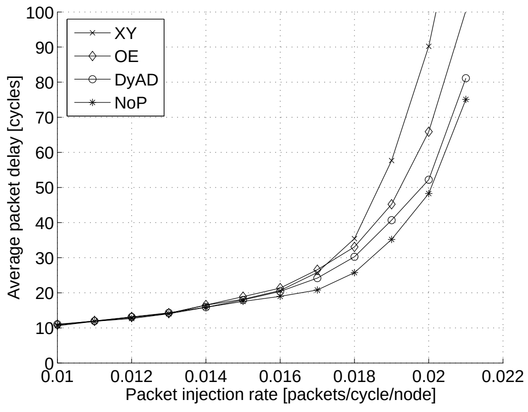

Figure 10: Delay variation under *mms* traffi c scenario.

results in a improvement of both average delay and saturation point.

Finally, as a more realistic communication scenario we consider a generic MultiMedia System which includes an h263 video encoder, an h263 video decoder, an mp3 audio encoder and an mp3 audio decoder [6]. 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 [1]. For this scenario we consider self-similar packet injection distributions since it has been observed in the bursty traffi c between on-chip modules in typical MPEG-2 video applications [11]. As we can observe from Figure 10, once again NoP-OE outperforms the other routing algorithms. For a pir value of 0.016, where none of the algorithms are saturated, NoP-OE exhibits an average delay of 17 cycles vs. 21 cycles of the other algorithms.

5 Implementation Issues

In a SoC environment, 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 synthesised the designs using Synopsys Design Compiler and mapping them onto a $0.13\mu m$ library from Virtual Silicon. We found that, within the router, the buffer area signifi cantly dominates the logic [9]. The buffer area, in turn, largely depends on the flit size. For a flit size of 64 bit and FIFO buffers with a capacity of 4 flits we found that the area overhead due to the NoP is less than 8%, 6%, 5% as compared to XY, Odd-Even, and DyAD respectively.

We also performed energy analysis of the designs. The energy dissipation of NoC fabrics arise from two different sources: 1) the router blocks, which include the buffers, and 2) inter-router wire segments. We determine the energy dissipated

Table 1: Energy consumption to drain 10MB of data for different traffi c scenarios and different pirs.

		Energy (mJ)			
Scenario	pir	XY	OE	DyAD	NoP
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

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 $0.151nJ$, $0.178nJ$, $0.182nJ$ and $0.189nJ$ for XY, Odd-Even, DyAD, and NoP-OE respectively. We assumed the tile size to be $2mm \times 2mm$ and that the tiles were arranged in a regular fashion on the floorplan. The load wire capacitance was set to $0.50fF$ per micron, so considering an average of 25% switching activity the amount of energy consumed by a flit for a hop interconnect is $0.384nJ$. We used these values to backannotate our functional simulator in such a way to obtain an estimation of the total energy consumption.

Table 1 summarizes, for each traffi c scenario, the overall energy required to consume a fi xed workload of 10 MB at fi ve representative packet injection rates.

The fi rst pir value represents a very low traffi c 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 the Table 1 simply reflects a condition of full saturation. In these cases the fi xed workload of 10 MB 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 fi rst row of each traffi c scenario, NoP-OE results in a negligible energy overhead as compared to the other adaptive algorithms, whereas the overhead respect to XY is more evident, ranging for 18% to 27%. This simply means that, at low traffi c 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 1 are being considered, NoP-OE router may result in a better energy usage since it is less affected by congestion related issues. In particular, in most of the cases, under heavy traffi c workloads the better performance of NoP-OE may balance its power dissipation overhead thus resulting in an overall saving of energy consumption. Once again makes exception the uniform traffi c scenario, as a consequence of the better performance of the static XY routing for this type of traffi c, as already discussed in Subsection 4.3.

6 Conclusions

In this paper we have proposed a selection strategy that introduces the concept of Neighbors-on-Path to improve the performance of a Network-on-Chip. The aim of our algorithm is to exploit the situations of indecision that can occur in an adaptive wormhole routing. The approach, that is general in nature, has been applied to the Odd-Even 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 traffi c workloads.

The comparison of routing and selections strategies strictly depends on the traffi c scenario that populates the NoC. Future developments include a further mapping of real application tasks on NoCs to test the validity of the proposed selection strategy for applications whose performance at higher packet injection rates is critical.

References

[1] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 182–187, Stockholm, Sweden, Sept. 8–10 2004.

[2] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel Distribuited Systems*, 11(7):729–738, 2000.

[3] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, Las Vegas, Nevada, USA, 2001.

[4] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of the Association for Computing Machinery*, 41(5):874–902, Sept. 1994.

[5] J. Hu and R. Marculescu. DyAD - smart routing for networks-on-chip. In *ACM/IEEE Design Automation Conference*, pages 260–263, San Diego, CA, USA, June 7–11 2004.

[6] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551–562, Apr. 2005.

[7] 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. In *IEEE Computer Society Annual Symposium on VLSI*, page 117, 2002.

[8] P. Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Computing Surveys*, 30(8):374–410, Sept. 1998.

[9] P. P. Pande, C. Grecu, and I. Ivanov. High-throughput switch-based interconnect for future SoCs. In *IEEE International Workshop on System-on-Chip for Real-Time Applications*, pages 304–310, 2003.

[10] International thechnology roadmap for semiconductors. Semiconductor Industry Association, 2003.

[11] G. Varatkar and R. Marculescu. Traffi c analysis for on-chip networks design of multimedia applications. In *ACM/IEEE Design Automation Conference*, pages 510–517, June 2002.

[12] T. T. Ye, L. Benini, and G. D. Micheli. Packetization and routing analysis of on-chip multiprocessor networks. *Journal of System Architectures*, 50(2-3):81–104, 2004.