Proceedings of the 5th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Madrid, Spain, February 15-17, 2006 (pp94-99)

A New Selection Policy for Adaptive Routing in Network on Chip

Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, Davide Patti
University of Catania
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
V.le Andrea Doria, 6 — 95125 Catania, Italy

Abstract: Efficient and deadlock-free routing is critical to the performance of networks-on-chip. In this paper we present a new selection policy that can be coupled to any adaptive routing algorithm to improve the performance in terms of average delay. The proposed approach introduce 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. Experimental comparisons between the proposed approach against both the deterministic XY routing and current state-of-the-art of adaptive routing algorithm for different traffic scenarios show an improvement in terms of saturation point respectively of 36% and almost 10% in average.

Key–Words: Networks on Chip, adaptive routing, deadlock-free routing

1 Introduction

The International Technology Roadmap for Semiconductors [7] foresees on-chip interconnection system will represents the limiting factor for performance and power consumption in next generation systems-on-a-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. The limiting factor is mainly the topological organization of the interconnection between the various units, which will substantially remain bus-based.

A type of architecture which lays emphasis on modularity and is intrinsically oriented towards supporting heterogeneous implementations is represented by Network-on-Chip (NoC) architectures [2]. 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 standardising on-chip communications. The NoC architectural topology most frequently referred to can be represented by an $m \times n$ mesh [5]. Each tile of the mesh contains a *resource* and a *router*. Each router is connected to a resource and the four adjacent routers. A resource is any IP compatible with the NoC interface specifications.

The design of efficient, high performance, on-chip routers represents a critical issue for the success of the NoC approach. Routers can be classified into two types: *Deterministic* and *adaptive*. In deterministic routing the path is completely determined by the source and the destination address. A routing technique is adaptive if, given a source and a destination, 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 [6] 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 adaptive algorithm establishes the set of output channels it can be routed on. The body flits will then follow the reserved channel, the tail will later release the channel reservation. Unfortunately, blockage of the header flit of a packet blocks all the remaining flits of the packet along the established path, thus occupying space in the router buffers. 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 estabilished by the router contains at least two *non reserved*[1] output channels, the router has to choose one of them. This further phase required is usually

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

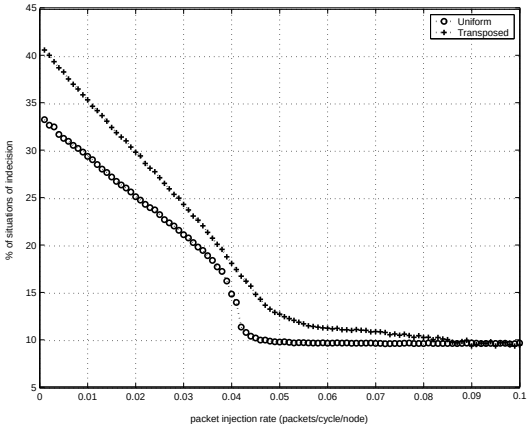

Figure 1: Percentage of situations of indecision with various packet injection rates and in different traffic scenarios.

referred as *selection*. Figure 1 gives the percentage of situations of indecision, i.e. the relationship, expressed as a percentage, between the number of times a router is able to route a packet towards alternative channels (provided they are not reserved for other packets) and the total number of packets. The condition is simulated with various *packet injection rates* (*pir*) and in two different traffic scenarios (which will be described in Section 4). As can be observed, with low *pir* values (below 0.04) the number of packets circulating in the network is low, so there is much more possibility of having free (non reserved) channels on which to route the packet. As *pir* increases, the number of situations of indecision is about 10% of the total.

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 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 [3] 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 [1] 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 [4] to define a general routing methodology known as DyAD (Dynamically switching between Adaptive and Deterministic modes). DyAD work combines the state-of-art in adaptive routing strategy (odd-even) with the efficiency of deterministic XY, proposing a selection strategy that chooses the direction in which the corresponding downstream router has more empty slots in its input FIFO. Another selection strategy for NoCs, called *look-ahead*, was proposed by Ye et al. [8]. Althought interesting, it doesn't guarantee the deadlock-free condition which we consider as key issue in Network-on-Chip routing.

The rest of the paper is organized as follows: In Section 3 we illustrate the idea behind the proposed approach, explaining how it can be applied to an adaptive routing function. In Section 4 we apply our selection strategy to the wormhole routing based on odd-even and compare it to DyAD which represents, at our knowledge, the best attempt to improve adaptive routing in NoCs. Finally, in Section 5 we outline directions for future development.

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 $R : N \times N \rightarrow \mathscr{P}(C)$, where $\mathscr{P}(C)$ 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.
- Several output channels to send a packet to, but only one of them is not reserved. 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.

Consider the situation shown in Figure 2(a) where the node $(1,1)$ has to choose between two possible candidates as the next destination for a header flit. Our problem is to choose the next node so that the path taken by the packet will be as free of congestion as possible. 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. 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

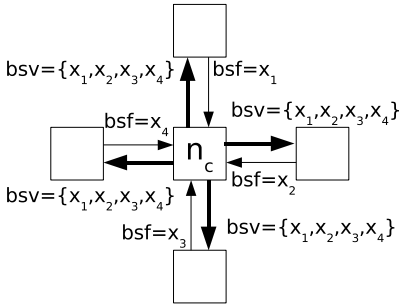

Figure 3: Data exchange for buffer space vector creation at node n_c

the concept of *Neighbors-On-Path*: node $(1,1)$ executes 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's 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 Neighbor-on-Path 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

To understand how neighbors-on-path approach works it must be remembered that each node can actually monitor its congestion status since it can monitor the status of each of its four input queues. In our scenario, depicted in figure 3, a generic node n_c receives a *buffer space flag* (bsf) from each of the four adjacent neighbors. Each flag consist of a single bit representing the available/unavailable status of the input buffer that connect the neighbor to n_c. The four bits received by n_c are used to create a *buffer space vector* (bsv), which is regularly (e.g. each clock cy-

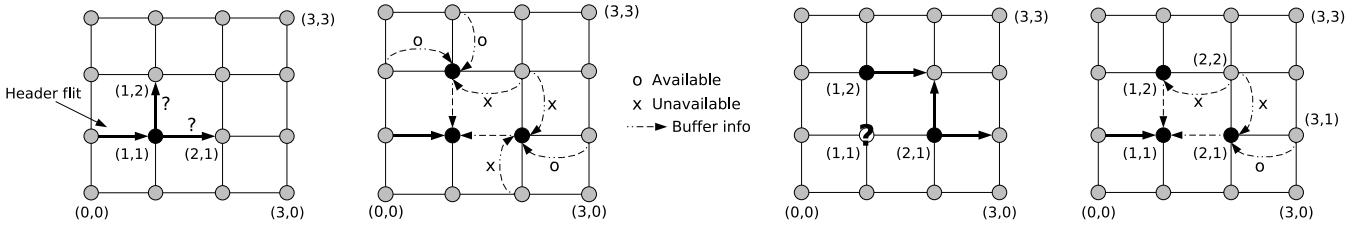

Figure 2: Neighbors-on-Path congestion-aware routing algorithm explaination. (a) Node $(1,1)$ has to choose between two possible candidates. (b) Nodes $(2,1)$ and $(1,2)$ receive information about buffer availability from their neighbors. (c) Node $(1,1)$ figure out which output channels would return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

cle) updated and sent to each adjacent neighbor node using ad-hoc point-to-point signals. Let n_c and n_d be the current node and the destination node respectively. We introduce some notations:

- *AdmissibleChannels*: the set of alternative channels provided by routing function $R(n_c, n_d)$ that are available.
- $dest(c)$: The destination node of a channel c. In other words, if $n = dest(c)$, n is the node for which the channel c is an input channel.
- $bsv[n1][n2]$: Boolean value (available/unavailable) representing the status of the channel from node $n1$ to node $n2$, where $n1$ is a neighbour of n_c and $n2$ is a neighbour of n_1.

Figure 4 gives the NoP algorithm executed at the node n_c in pseudo-code.

For each candidate output channel (line 4), the current node n_c computes the set of neighbors-on-path nodes (line 5-6) to investigate the availability of their input buffers. Using a credit mechanism, the credit of a candidate destination is increased for each neighbor-on-path with available space in the input queue (line 9). The node finally selected as the destination for the packet is the one with the most credit (line 13).

```
1   NoP_Select(input: AdmissibleChannels,
2                     output: SelectedChannel)
3   credits[] ← 0
4   for_each channel1 ∈ AdmissibleChannels {
5     node1 ← dest(channel1)
6     NoPChannels ← R(node1, nd)
7     for_each channel2 ∈ NoPChannels {
8       node2 ← dest(channel2)
9       if bsv[node1][node2] is available then
10        credits[channel1]++
11    }
12  }
13  SelectedChannel ← ch st credits[ch] = max(credits[])
```

Figure 4: NoP selection algorithm performed at node n_c.

4 Experiments

4.1 Traffic Scenarios

Traffic scenarios characterize the traffic on the network. Three types of traffic are considered in simulations:

1. *Uniform*. At times indicated as *random* traffic, this is the most simple scenario: A node sends the packet to each other node with the same probability.
2. *Transposed*. This scenario is only applicable if the two-dimensional mesh is square. A node (i, j) only sends packets to a node $(N-1-j, N-1-i)$, where N is the size of the mesh.
3. *Hot-spot*. According to the type of traffic, some nodes are more favored when the destination is chosen. Each node, that is, has the same probability of being a destination, with the exception of some that have a higher probability.

4.2 Evaluation Metrics

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.

It is also important to know how many packets are "injected" into the network, and when. We will assume that each node sends the same number of pack-

ets and indicate the number of packets a processing element (PE) sends at each clock cycle as the *packet injection rate* (*pir*) ($0 < pir \leq 1$). A *pir* of 0.1 [packets/cycle/node] means that each 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. This instant is chosen on the basis of a negative exponential distribution.

Evaluation of the quality of the various routing algorithms is based on the *saturation point* (*sp*), which is the injection rate below which the network is not saturated and above which it is. If the injection rate is close to or above *sp*, the performance of the system deteriorates rapidly. Routing algorithms are required to have a high saturation point.

Experiments were carried out on a NoC simulation platform developed in SystemC. The size of the NoC was 6×6. Traffic sources generate 16-byte packets with an exponential distribution, the parameters of which depend on the packet injection rate. A flit is 32 bits. The FIFO buffers have a capacity of 3 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% confidence interval.

4.3 Results

For each traffic scenario and algorithm, we will give the average packet delay (expressed in clock cycles) with various *pir* (expressed as the number of packets sent by each node in each clock cycle). The routing algorithms compared are XY, DyAD, and NoP applied on odd-even routing. We consider the choice of applying NoP selection to odd-even the most natural one, since odd-even has been proved to exhibit the best performance among different traffic scenarios [1] and is also at the base of the DyAD approach.

Figure 5 shows the results obtained when the network has *uniform* traffic. As can be seen, the non adaptivity of the XY algorithm gives the best performance when the workload is high (the saturation point is 0.048 as compared with 0.042 for NoP and 0.04 for OE). The reason for this is that the algorithm is based on long-term global information [3]. Routing packets along one dimension and then the other, the algorithm distributes the traffic in the most uniform manner possible in the long term.

Adaptive algorithms, on the other hand, select the routing paths on the basis of short-term local information. Their way of operating tends to create "zigzag"

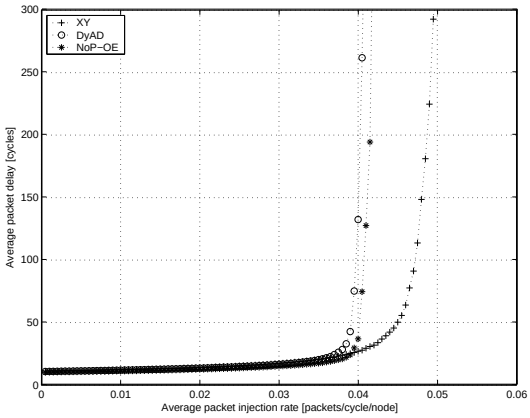

Figure 5: Average delay with various *pir* for the *uniform* traffic scenario.

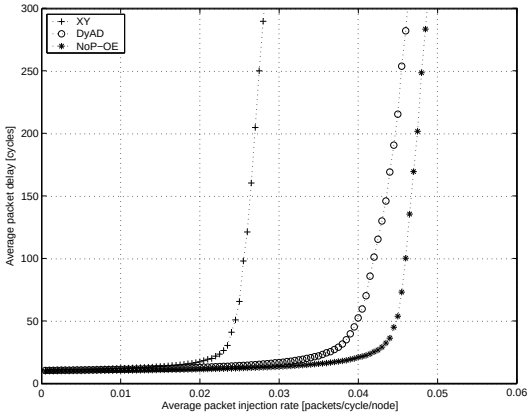

Figure 6: Average delay with various *pir* for the *transposed* traffic scenario.

paths which hinder the uniform distribution of traffic. This causes greater contention and deteriorates performance. This type of traffic is used in several simulation scenarios but is not to be found in real applications. However, although the NoP-OE routing scheme does not perform as well as XY, it results slightly better than DyAD.

If we consider *transposed* (Figure 6) traffic scenario, it is observed that a network adopting XY performs poorly due to its determinism in distributing packets. The network saturates at an *sp* lower than 0.03. The performance of a network based on the NoP-OE and DyAD routing schemes is different. In this case the *sp* is 0.048 and 0.045 respectively. In these traffic conditions, the NoP-OE scheme gives a 55% improvement in sustainable throughput as compared with XY and a 6-7% improvement on DyAD.

The next traffic scenario considered is *hot-spot*. It is considered to be a more realistic model as in most applications processes communicate frequently with only a part of the total number of other processes (e.g.

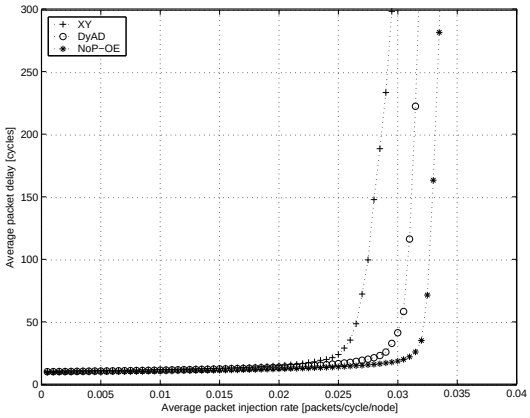

Figure 7: Average delay with various *pir* for the *hot-spot* traffic scenario with $hsp = 15\%$.

memory storage nodes, I/O resources).

In our case study, we simulated the behavior of a network in which there is only one node receiving a greater amount of traffic. Node $(3,3)$ was chosen as the hot-spot node, and the percentage of hot-spot traffic (*hsp*) was set to 15%. This means that the traffic directed towards node $(3,3)$ will be about 15 times greater than that directed towards the other nodes. Figure 7 shows the result of the simulation.

When several traffic flows are directed towards a single node, a router adopting a deterministic routing algorithm like XY will be forced to route them towards the same output line, thus saturating the input queues. It is thus clear why the network based on XY, which performed better with uniform traffic, has a lower saturation point (about 0.029) than the two 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. What is interesting to point out is the better performance obtained by NoP-OE. Being able to route packets on the basis of congestion information received from neighbors allows NoP to obtain a higher saturation point, with a consequent 22% improvement in sustainable throughput as compared with XY and 8% as compared with DyAD.

5 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 occour in an adaptive wormhole routing. The approach, that is general in nature, has been applied to the odd-even and compared to both static XY and DyAD routing schemes. Althought the performance obtained at lower packet injection rates resulted very similar to the other approaches, the simulations performed show an improvement of the saturation point.

The comparison of routing and selections strategies strictly depends on the traffic scenario that populates the NoC. Future developments include the 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.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel Distributed Systems*, 11(7):729–738, 2000.

[2] 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.

[3] 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.

[4] 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.

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

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

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

[8] 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.