PathAware: A Contention-aware Selection Function for Application-specific Network-on-Chips

Behrad Niazmand, Akram Reza, Midia Reshadi
Department of Computer Engineering
Science and Research Branch, Islamic Azad University
Tehran, Iran
{b.niazmand, reza_ak, reshadi}@srbiu.ac.ir

Abstract—Network-on-Chip (NoC) has been introduced as a novel solution to overcome the constraints met in on-chip interconnection networks. Performance of NoCs is one of the important concerns of researchers and designers. One of the factors that can affect performance in on-chip networks is the occurrence of congestion when routing packets. In this paper we introduce an output selection function, named “PathAware”, which can be exploited with any adaptive routing algorithm. The main purpose is to address the situations in which more than one output port are chosen as candidates and thus to reduce latency and balance traffic load by selecting the appropriate output port leading to a minimal path as soon as it becomes available. In order to avoid deadlock, we have exploited turn model adaptive routing algorithms. Simulation results demonstrate that when using “PathAware” selection function along with “West-First” and “Odd-Even” adaptive routing algorithms, it can outperform ”Random”, ”Buffer-Level” and ”Neighbor-on-Path” selection functions in terms of latency and an improvement of 41% can be achieved (in best case), while imposing a negligible overhead on energy consumption.

Keywords—Network-on-Chip (NoC), routing, selection function, contention, congestion, traffic distribution

I. INTRODUCTION

Network-on-Chip (NoC) is considered as a promising solution for providing scalable, reliable and modular communications in Multi-Processor System-on-Chips (MPSoCs). The concept of NoC has been introduced in order to address and overcome the scalability issues met in shared-bus networks [1] and make it possible to maintain over hundreds of processing cores on a single die. Yet, there are still many challenges in this field that has kept the designers concerned, such as reducing latency, providing energy-efficient communications and much more, on a limited die area. One of the important factors that can affect the performance of NoCs, specially latency, is routing and the way it is implemented. A routing algorithm can be implemented either using routing tables or in form of combinational circuits. One of the methods proposed is LBDR [2], a logic-based distributed mechanism that makes it possible to implement routing algorithms using sets of configurations bits (routing bits and connectivity bits) and removes the need of routing tables at every switch, leading to better results in terms of both area and latency. By using such a mechanism, increasing the number of on-chip cores would not be that troublesome as the performance of the network is not affected by the number of cores. In this paper we have also made use of this method for implementing routing algorithms.

One of the important matters that needs to be avoided when routing, is the occurrence of deadlock. In order to maintain this feature, we have used turn model adaptive routing algorithms [3] which try to avoid closed cycle scenarios in routing by imposing some restrictions on the turns allowed by packets to take. Some of the well-known introduced turn model routing algorithms are: West-First, North-Last, Negative-First and Odd-Even [4]. The first three algorithms act the same in all networks columns, invalidating two of the 8 possible turns. However, in Odd-Even routing, the restricted turns are different in odd and even columns, making it possible to achieve more flexibility when routing packets.

As mentioned in [5], a routing algorithm is modeled by means of two components: routing function and selection function (Fig. 1). The former applies the routing algorithm to the packet and provides a set of output ports as candidates, i.e. the output ports on which the packet can be sent. The latter selects the appropriate output port from the provided set, considering the status of the output channels, such as traffic or fault at the current node. The main duty of a selection function is to choose the free available output port (if any) by taking into account the distribution of traffic in network and avoiding congested areas, as much as possible. According to [5], it is the selection function that has the most impact on performance parameters, such as latency and

![Fig. 1. Overview of routing function and selection function [7]](image-url)
throughput, thus, our focus is also on this component.

Network contention has been classified into three types: source-based, destination-based and path-based contention [6]. The naming of each type is based on the location where the contention has occurred, i.e. at least two data flows having overlapping at starting nodes, ending nodes or intermediate nodes. It has also been mentioned in [6] that path-based contention has more influence on performance, specially latency. Thus, we also focus on this type of contention and try to mitigate it and balance traffic load in the network by proposing a contention-aware output selection function.

In this paper, we introduce a new output selection function, named "PathAware" and it is designed with the aim of distributing traffic and avoiding path-based contention in network as much as possible, thus leading to better latency results.

The rest of the paper is organized as follows: In section II a summary of the related work is provided, listing some of the selection functions introduced recently. Section III describes the proposed selection function by providing a motivational example and explaining the functionality of the function in detail. Section IV is dedicated to experimental results in which the proposed approach is compared with three other selection functions in terms of latency and energy consumption. Finally, concluding remarks and works that can be pursued in future are provided in Section V.

II. RELATED WORK

Before explaining the concept of the proposed selection function, we first provide a review of some of the related work and contributions in the domain of selection functions.

Ascia et al. have introduced a selection strategy for adaptive routing in Networks-on-Chip, known as Neighbor-based regional routing (NBAR) [7]. The concept of NoP relies on the aim of selecting the appropriate output port which leads to minimum congestion. It overcomes the situations of indecisions by collecting congestion status from routers beyond the neighbors of the current node. In [8] an adaptive routing algorithm has been proposed, in which a fuzzy controller is exploited to combine two metrics related to congestion, i.e. number of free buffers and number of crossbar requests. Authors of [9] have shown the possibility of designing high-performance application-specific routing algorithms, in order to distribute traffic uniformly in the network. This is done by means of the information related to communications in applications, including communication topology and communication bandwidth. In [10] a power-aware output selection function has been represented, considering power (both self and coupling switching activity of the selected output link) in addition to performance, as a metric when selecting the appropriate output port. In [11] an Adaptive Power-Aware Output Selection (APAS) has been proposed which exploits the regional table organization technique statically and dynamically, with the goal of providing a model for further load-balancing in the network. Authors of [12] have introduced an adaptive routing algorithm, Traffic- and Throttling-Awareness Routing (TTAR) that is able to deal with traffic congestion scenarios caused by throttling of transient-temperature control, making it possible to balance the network traffic and detour the throttled tiles. In [13] a lightweight technique, named Regional Congestion Awareness (RCA), has been proposed in order to improve global network balance when adaptive routing is used. In [14] Destination-Based Adaptive Routing (DBAR) has been represented in order to address issues related to workload consolidation and overcome issues associated with intra- and inter-application interference during output port selection for consolidated workloads. It is mentioned that DBAR offers effective adaptivity for congestion beyond neighboring nodes. In [15] a method for improving routing in network-on-chips has been proposed that utilizes a self-optimized strategy with the aim of avoiding hotspots when routing packets. Authors of [16], unlike the previous works that had focused on improving output selection functions, have introduced a mechanism in order to improve input selection in routers. In this paper, a technique, named Contention-Aware Input Selection (CAIS), is proposed with the aim of improving routing performance through selecting the contention-free input port based on the contention level of the upstream switches, which can in turn remove possible network congestion. In [17] a novel output selection method has been proposed, named Look-Ahead Traffic-aware Execution (LATEX), which can be used with any adaptive routing algorithm and can be exploited for specific applications. The main goal is to provide an efficient load balance and traffic distribution, leading to better performance results.

III. PATHAWARE SELECTION FUNCTION

A. Motivational Example

As mentioned in section I, one of the factors that affects the performance of routing in on-chip networks, is the robustness of the selection function in dealing with scenarios imposing congestion. So, it is important that the selection

![Image](image_url)

Fig. 2. The impact of contention-awareness on latency
strategy would avoid congested areas as much as possible. One type of congestion that has the most impact on latency is path-based contention [6]. As it can be seen in Fig. 2, two different routing algorithms have been considered: the one in case (a) is non-contention-aware and the other in case (b) is contention-aware. It can be observed that the employment of a contention-aware routing algorithm not only avoids situations leading to path-based contention as much as possible, but it also improves performance in terms of latency. Thus, we also focus on this concept and introduce a selection function that is contention-aware and intends to choose the output port with the least potential of congestion occurrence.

B. PathAware Selection Function Description

In the following paragraphs the functionality of the proposed selection function, "PathAware", will be explained in detail. It is worth mentioning that similar to the routing algorithm, the selection function is run at every node, having a distributed manner. Due to its simplicity and low cost, we have implemented the routing algorithms using LBDR, a logic-based routing implementation mechanism that describes the network topology and routing algorithm using two sets of configuration bits (routing bits and connectivity bits). The proposed selection function makes use of the signals generated by LBDR mechanism when choosing the appropriate output port.

One of the features of PathAware selection function is that it can easily be adapted to any adaptive routing algorithm and be customized for its needs. For this purpose, only the routing bits need to be re-initialized, as different routing algorithms may impose different routing restrictions. In the following paragraphs, we have chosen West-First as the predominant routing algorithm, thus we describe the proposed selection function customized for this routing algorithm. However, as mentioned, the selection function can easily be adapted to other routing algorithms, such as Odd-Even is as well.

As mentioned in Section I, in a routing algorithm, first the routing function provides a set of output ports as candidates, taking into account the routing restrictions imposed by the routing algorithm. Afterwards, it is the duty of the selection function to choose the appropriate output port, considering the status of candidate ports, trying to avoid congestion in the network and reduce path-based contention as much as possible. We describe the functionality of PathAware selection function, based on the number of ports chosen as candidates by the routing algorithm and leading to minimal paths:

When two output ports are chosen as candidates: When employing West-First routing algorithm, this case occurs when the destination node is located on the North-East or South-East quadrant relative to the current node. Since the routing algorithm does not impose any restrictions in these quadrants, both output ports can be chosen as candidates. We consider the North-East quadrant for now. Similar deduction can be inferred for the South-East quadrant. As it can be seen in Fig. 3.a, first the appropriate output ports are determined by the routing logic (in this case: N and E ports). Afterwards the PathAware logic comes into action and chooses the suitable output port based on the availability of N and E ports. The Comparator unit (Fig. 3.b) is responsible for determining whether the location of the current node is the source node or not. If both of the output ports are available, two conditions can occur: If the current node is the source node of the packet, then the selection is done by means of NoP selection function (described in [7]). Otherwise, if the current node is an intermediate node on the way to destination, then the selection function immediately forwards the packet to the output port opposite to the incoming one, e.g. for NE quadrant, packet coming from S (with binary value '11') and W (with binary value '10') ports are forwarded to N ('00' in binary) and E ('01' in binary) output ports, respectively. This functionality can easily be modeled by means of an inverter which complements the binary value of the incoming port (Fig. 3.b). If only one of the output ports is available, then it is chosen immediately by the selection function without waiting on the other one to become available. Finally, If both of the output ports are busy, then the packet has no choice but to wait until at least one output port becomes available.

When only one output port is chosen as a candidate: In West-First routing algorithm, this situation occurs when the destination node is located on the NW or SW quadrant relative to the current node, since the routing algorithm imposes North-to-West and South-to-West restrictions in these quadrants, respectively. Thus, no selection has to be performed, because only one output port is chosen as a candidate. It is worth mentioning that this case also happens when the destination node is on the same row or column as the current node, i.e. when located on the East, West, North or South side. In both cases, the selection function has no
impact on the path the packet takes, since there is only one possible minimal path and thus the function will not be invoked.

**When no output port is chosen as a candidate:** This situation can occur because of two reasons. It can either be because of the packet reaching its destination, or it can be because of the packet not being able to be forwarded to any output ports due to the presence of fault in the intermediate links or routers. Again, In both cases the selection function is not involved and thus it has no effect on the path the packet takes.

### IV. EXPERIMENTAL RESULTS

#### A. Proposed Cross Traffic Generator Tool

In this paper we have also developed a traffic generator tool that is used in our simulations and is capable of generating cross table-based traffic patterns. We have named the proposed tool "CrossTrafficGenerator" and it is written using C++ programming language. The aim of the proposed tool is to consider regions in the network and generate cross traffic patterns, meaning that the nodes located at the quadrant's boundaries send data flow to each other diagonally. In fact, the purpose is to create a traffic scenario with the worst condition of having path-based contention on most of the links inside the quadrant.

#### B. Simulation Environment

In this paper, we have used Noxim [18] as our simulator, which is a flit-level cycle-accurate Network-on-Chip simulator. We have considered "Random" and "Cross Table-based" (generated by the proposed tool) traffic patterns in our simulations and evaluation of the proposed selection function. It is worth mentioning that the proposed selection function (PathAware) is compared to three other existing selection functions, "Random", "Buffer-Level" and "Neighbor-on-Path" in terms of latency and energy consumption.

The metrics used while simulation and evaluating the results include: performance (focusing on latency), energy consumption and Output Waiting Array. In order to be able to evaluate the distribution of traffic in network when applying the proposed selection function and making comparisons, we have implemented an array, named
Waiting Array, per each output port in every router, storing flit information as its elements. If a header flit needs to be forwarded to one of the output ports which is already busy or occupied by another flit, it needs to wait until the channel becomes free. In such situations the flit information, such as the source ID, destination ID, number of hops, etc. are stored as an element in the waiting array. It is worth noting that the array is not a buffer and the flit itself is not stored in it, but it is just an array for storing flit information which will be used later in evaluation. By means of the waiting array in every router, it is possible to count the maximum number of elements and use it as a parameter, determining the maximum number of data flows waiting on a specific output port and its corresponding link to become free. Thus, if this parameter becomes less in value when using a selection function, it can be inferred that the scenario has led to a better distribution of traffic and path-based contention is avoided as much as possible. This leads us to the conclusion that the maximum number of elements in the waiting array is related to the level of traffic distribution.

In order to evaluate simulation results, we have considered three different scenarios. In all three scenarios, wormhole [19] has been used as the switching mechanism. Turn model adaptive routing algorithms are used in both scenarios, implemented using LBDR mechanism. The buffer size in each router is set to 4 flits. We have tried to provide a long range of packet injection rates when simulating the scenarios, thus we have considered packet injection rates ranging from 0.007 to 0.04 in one scenario and from 0.02 to 0.1 in the two other scenarios. In all scenarios, the proposed selection function, PathAware, All the communication schemes have been set to unicast. It is also worth mentioning that each simulation has been repeated 10 times per each packet injection rate in order to achieve more accurate results and the minimum value is chosen amongst the calculated values.

Now we try to explain each scenario in detail, with its specific characteristics and afterwards, the simulation results and their analysis will be demonstrated for each scenario.

C. Simulation Results for Random Traffic Pattern

In the first scenario, we have considered a 4x4 2D mesh as the topology and four selection functions, including Random, NoP, BufferLevel and PathAware are evaluated for West-First routing algorithm under Random traffic pattern, in which every node sends data to all other nodes with equal probability. The results related to packet latency are depicted in Fig. 4.a. As it can be seen, the proposed selection function has achieved better latency results in comparison to three other selection functions (Random, NoP and BufferLevel), specially when the packet injection rate exceeds 0.03 (after the saturation point), which could be due to having a better traffic distribution in network. Another reason for achieving better latency results can be that the selection of the output port is mostly done immediately without the need of the packet to wait for the NoP signals to provide status information from neighbor nodes. Regarding energy consumption, Fig. 4.b shows that there is a slight difference in energy consumption between all selection functions and thus the energy overhead of the proposed approach is acceptable and negligible.

D. Simulation Results for Cross Table-based Traffic Patterns

In the second scenario a 4x4 2D Mesh network has been considered and the switching mechanism and the routing algorithm are again considered worm-hole West-First, respectively. However the traffic pattern has been changed to cross table-based, which is generated by the proposed CrossTrafficGenerator tool, with the aim of causing the network to have path-based contention in different regions. Fig. 4.c demonstrates the latency results, comparing only three selection functions (excluding Random), and it can be seen that once again the proposed selection function outperforms the other two functions in terms of latency and in the worst case it is similar to Bufferlevel, but its saturation point is higher than NoP. The energy overhead is also acceptable and it even shows slight improvements when applying the proposed selection function (Fig. 4.d). The reason might be that less router ports are involved in the process of routing the packets when PathAware selection function is exploited.

The last scenario is almost the same as the second one, except that the mesh size has been set to 5x5 and the routing algorithm has been changed to Odd-Even, which is a more flexible routing algorithm. Fig. 4.e shows the simulations results for latency and it can be inferred that the proposed selection function can achieve better latency results in comparison with NoP and BufferLevel selection functions even when using Odd-Even routing algorithm. The other remarkable feature is that the saturation point has increased from 0.06 to 0.07 when using the proposed selection function, which shows a more distributed traffic and that the network is able to tolerate more traffic load. Also, as it can be seen in Fig. 4.f, energy overhead is acceptable and it is approximately the same value for all selection functions.

In addition, in order to have a better evaluation of latency results and show the effect of applying the proposed selection function on traffic distribution and alleviating path-based contention, we have considered a waiting array in every router (described in Section IV-B) and we have monitored the maximum value of its elements in the last two scenarios. Fig. 5 demonstrates the maximum value of this parameter for the last two scenarios. As it can be seen, when

Fig. 5. Evaluation of maximum number of flows on one link
using West-First as the routing algorithm (Fig. 5.a) and PathAware as the selection function, this value is decreased to 2, which is less than the value acquired for the other two selection functions (NoP and BufferLevel), thus showing a better traffic distribution and less packets waiting to acquire output ports. Fig. 5.b also shows that when deploying Odd-Even routing algorithm, even in the worst case, the value of this parameter is the same for all three selection functions, although there is still improvements in latency (according to Fig. 4.e).

V. CONCLUSION AND FUTURE WORK

One of the important concerns when improving routing in on-chip networks, is to avoid the scenarios of congestion. This can be done by designing a contention-aware output selection function that can balance traffic load and minimize path-based contention as much as possible, leading to better latency results. In this paper a new output selection function was proposed, named "PathAware", that could be used along with any adaptive routing algorithm. The main purpose is to overcome the situations of indecisions when the routing function provided more than one output port as candidates and also to reduce the latency caused by the packets waiting to acquire a busy output port. The ultimate aim was to both reduce path-based contention and distribute traffic load in the network as much as possible with having the least impact on energy consumption. Simulation results have shown that when customizing the proposed selection function for Odd-Even and West-First routing algorithms under random and cross table-based traffic patterns, it can outperform its rivals, i.e. Random, Buffer-Level and NoP selection functions in latency, and an improvement of 41% can be achieved in the best case, while imposing a negligible overhead on energy consumption. As a future work, non-minimal paths can also be considered in the selection function, leading to more flexible conditions when selecting output ports. In addition, since the routing implementation used in this paper is LBDR, only regular topologies (such as 2D mesh) and the irregular ones derived from the 2D mesh are supported. But, it can be expanded to support any irregular topology with an arbitrary structure to become more compatible with applications that impose irregularities on topology.

REFERENCES