A Path-Load Based Adaptive Routing Algorithm for Networks-on-Chip

Leonel Tedesco
FACIN-PUCRS
Av. Ipiranga, 6681 - 90619-900
Porto Alegre – BRASIL
leonel.tedesco@pucrs.br

Fabien Clermidy
CEA-LETI-MINATEC
38054 Cedex 9
GRENOBLE - FRANCE
fabien.clermidy@cea.fr

Fernando Moraes
FACIN-PUCRS
Av. Ipiranga, 6681 - 90619-900
Porto Alegre – BRASIL
fernando.moraes@pucrs.br

ABSTRACT

Applications executing in current MPSoCs present traffic behavior with different characteristics in terms of QoS requirements and traffic modeling. Another important MPSoC traffic feature is its unpredictability and dynamic nature. Networks-on-chip (NoCs) are communication structures being used due to higher degree of parallelism, fault tolerance, and scalability, when compared to busses. Even with increased bandwidth due to parallelism, some flows may compete for the same network resources, affecting the applications performance, and possibly violating QoS requirements. Adaptive routing algorithms may reduce such congestion, enabling dynamic path modification according to some congestion evaluation metric. State of the art approaches have a limited view of the congestion areas, since each router take routing decisions based on its neighbors congestion status. Such local decision may lead packets to another NoC congested region, therefore being inefficient. This paper proposes a new method, using the information of all routers in the source-target path. This method relies on a protocol for QoS session establishment, followed by distributed monitoring, and re-route to non-congested routers. The set of executed experiments presents results concerning latency and buffer utilization when the method proposed is applied.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – advanced technologies, algorithms implemented in hardware, VLSI (very large scale integration).

General Terms

Design, Experimentation, Measurement, Performance, Theory, Verification.

Keywords

Networks on Chip, Traffic Monitoring, Dynamic Routing, Quality of Service.

1. INTRODUCTION

Application traffic that occurs on typical MPSoCs environments includes data transfers between processors, memories, specific units, being characterized in terms of modeling by multimedia streams, control signals, and data blocks transfers [1]. This traffic also has flows with different QoS levels requirements, which is usually specified in terms of throughput, latency, and deadlines.

NoCs are generally built targeting a specific application, where data are generated at periodic time intervals and traffic behavior can be previewed. On the other side, the amount of computation and communication in emerging MPSoCs incurs in systems with a considerable variability and unpredictability on the overall communication throughput, bringing a dynamic nature to these. Changes on systems constraints and user operation are events that may appear at MPSoCs runtime. Fault tolerance aspects must also be considered [2].

Some approaches on the NoC literature target the treatment of unpredictable events and performance increasing on application traffic at run time. They have in common the utilization of monitoring units, which inform some entity that abnormality in the network traffic has occurred. Based on monitoring information, decisions are taken. The packets insertion control regulates the amount of data present on the NoC, to avoid congestion areas, being adopted by [3] and [4]. The attribution of dynamic priority for QoS packets based on router delivering rates is adopted by [5].

The focus of this work is the use of adaptive routing algorithms to treat network traffic at runtime. Based on some condition on network traffic, NoC routers are configured to change their routing decisions. The major part of state of the art proposals considers local traffic monitoring, that is, each router analyses the condition of its immediate neighbors to perform routing. The disadvantage of this approach is the absence of a global view of the current traffic, once a routing decision for a given flow may lead its traffic to other congested region. The present work proposes an algorithm that uses information collected for a given path to perform traffic routing. Monitoring data are transmitted on available bits present on application packets. Notifications of network conditions inform source traffic units that it is necessary the modification on the parameter that configures the path to the target router.

The remaining of the paper is organized as follows. Section 2 presents the state of art on adaptive routing algorithms. Section 3 presents the protocol adopted for session establishment and Section 4 focus on the proposed dynamic routing algorithm. Section 5 presents the experimental setup and results. Section 6 summarizes the contributions of this work.

2. RELATED WORK

Li et al. [6] present the architecture and modeling of the *DyXY* router. This router provides adaptive routing and assures deadlock-free and livelock-free guarantees, once each packet is transmitted along the shortest path between a given pair of source-target nodes. For each packet to be routed, is verified the alignment on X or Y axis with the target router. If there is no alignment, the packet is adaptatively routed according to stress values, which represent the congestion condition of the router. Buffer fillings on the neighboring routers configure stress values. A history buffer is used to help the input arbiter to select input requests for routing.

Hu et al. [7] present the *Dyad* routing algorithm, which combines the advantages of deterministic and minimal adaptive routing algorithms. Routing decisions are based on congestion values, which are represented by the input buffers occupation. Congestion flags information is exchanged between neighbor routers. This information is analyzed and, if the network is not congested, the DyAD router works on deterministic mode, else, the adaptive mode is activated.

Lofti-Kamran et al. [8] elaborate the *BARP* routing algorithm. Each router on the NoC considers its state of congestion, where two levels of threshold are monitored. Threshold levels correspond to the number of packets on the input buffer. According to the threshold reached, packets are sent to critical groups of routers, which have to change their packets output port, in order to avoid congested regions. It is a different approach compared to *DyAD* and *DyXY*, once the traffic is monitored in routers that are different to those which modifies their routing processing. Deadlock avoidance is guaranteed with the use of different virtual channels for each traffic direction. Many monitoring messages to groups of routers may be generated, which increases the overall traffic load on the NoC. In addition, there is a need at some level to control packets ordering at the target routers.

Al Faruque et al. [9] propose an adaptive on chip communication scheme, which dynamically allocates paths from a given flow to meet QoS requirements. In the presented method, the intermediate routers makes decisions in a local way depending on the available bandwidth in each direction to the neighboring routers and on the distance between current and target routers. This availability is evaluated according to an algorithm called *wXY*, which is based on the weight of a path between a given current and destination node. The obtained results increase the individual buffers utilization and decrease the total number of allocated buffers.

Gratz et al. [10] propose the *Routing Congestion Awareness Algorithm* technique (*RCA*), which informs congestion monitoring values in parts of the network beyond adjacent routers, bringing a global view of congestion. Congestion information is propagated across a specific network, separated to the one used for application data transmission. The number of active requesters for output ports was chosen as the preferential metric of congestion. Each router analyses the congestion information received to define which is the output port for its current flow. The efficiency of the approach is indicated by reduction on latency and increase on throughput compared to local congestion strategies for routing.

The analysis of congestion on neighbors routers presents a limited view of the overall network traffic condition by each router. This approach is suitable for traffic workloads with a high degree of locality, where nodes communicate with those that are closer to themselves. In traffic patterns with lower locality, the local monitoring may induce the routing of a given traffic to other congested areas. This work explores the use of an adaptive routing algorithm based on monitoring information on traffic paths.

3. PROTOCOL FOR MONITORING PROCESS

This Section presents the protocol used to transmit monitoring data information and notification of congestion occurrences. Two classes of packets are defined. DATA packets carry application data and traffic monitoring information (direction: source→ target). ALARM packets carry *congestion information*, used to change the path of the DATA packets (direction: target→source). This work adopts *source routing*. Figure 1 presents the packet structure. The header flit of DATA packets carries the path to the target, and *monitoring field*s. The path to the target router is specified in the *path to the target* field, which assumes for each router the values {N,S,E,W}. When the current direction is the opposite of the last direction taken, it means that the local port is reached. The *monitoring fields* contain: (i) `ident`, router address responsible to collect the congestion status; (ii) `ocup`, congestion status of the router specified in the field `ident`.

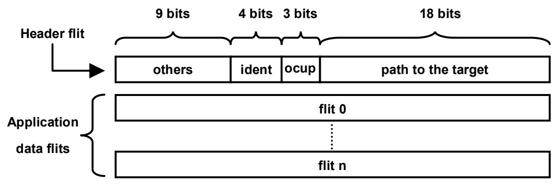


Figure 1 - DATA packet structure.

ALARM packets, presented in Figure 2, are back-propagated from the target to the source, containing one flit. The 18 less significant bits are used to indicate the path to target, as in the DATA packets. The field *congestion information* carries the path congestion status, where each bit indicates if a hop on the path is congested or not (limiting the maximum number of hops between source to target to 10). This packet is then used in the source PE to define a new path to the target. The computation of the new path to the target is explained in detail in Section 4.

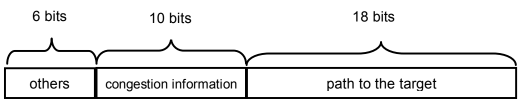


Figure 2 – ALARM packet structure.

The protocol for congestion registering works as follows. The traffic initiator opens a monitoring session with a first DATA packet. This packet configures each hop with its unique identification on the path in a structure named **RCT** (*Router Congestion Table*). At the target side, a **TCT** (*Target Congestion Table*) is initialized with size equal to the number of hops of the flow being initialized (function `init_table`). This table store traffic statistics for each hop of the flow. The subsequent DATA packets are sent with the `ident` field configured with the router address responsible to store its congestion level information.

Consider the source-target traffic pair 1-7 in Figure 3, starting the flow 'X'. The first DATA packet assigns to the `ident` field the value '1'. At each router in the path, this value is incremented, and stored in the field `Identification` of its **RCT** (function `ident_hop`). In the example, the **RCT** of router 2, is initialized with flow 'X' when the DATA packet with `ident`=2 arrives at router 2.

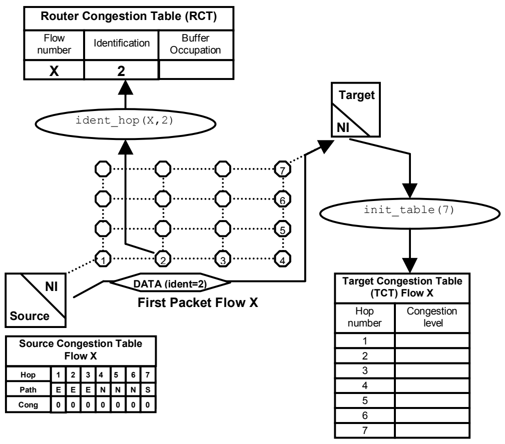


Figure 3 - Monitoring Session Establishment.

After path initialization, when an intermediate router receives a DATA packet, it verifies the `ident` field, and if its matches with the router address, the `ocup` field of this packet must be filled (function `monit`). The `ocup` is filled with the value of the adopted parameter for QoS evaluation, which can be the amount of used buffer slots, the output data rate or the average time spent by packets to traverse the router (function `register_cong`). If there is no match, the router updates its congestion table. The example of Figure 4 illustrates a request to capture the mean occupation of the congestion state of router 3 (`ident`=3). The mean buffer occupation (BO) of the input port receiving flow 'X' is 2, value forwarded to the target router by means of the `ocup` packet field.

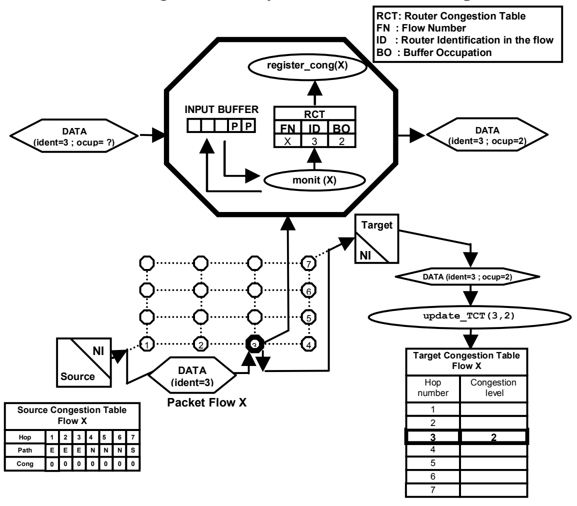


Figure 4 – Registering monitoring values.

When the DATA packet arrives at the target router, the content of the `ocup` field is inserted in the TCT (function `update_TCT`). The TCT, sized according to the number of hops in the path, is addressed by the `ident` field. As shown in Figure 3, the table has 7 entries, since there is 7 hops in the path of flow 'X'. Note in Figure 4 that the third entry on the table is updated. The target router, using the information stored at the TCT, has a global knowledge of the path being used by DATA packets. Such global knowledge enables the implementation of new routing heuristics. This is the main difference of our proposal to the major part of state of the art heuristics, which considers only neighbours routers.

4. CONGESTION DETECTION AND NEW PATH COMPUTATION

Periodically, the NI compares the TCT congestion levels to a given threshold, which is previously defined according to the QoS requirements of the application that generates the packet flow. If one or more values are greater than the threshold value, an ALARM packet containing the address of the congested routers is back propagated to the source router. When the source router receives the ALARM packet, a new path is computed and used for the next DATA packets.

As an example, consider the target NI at Figure 5. It is possible to observe that a congestion level equals to 5 is reached at routers 3 and 5. At this moment, a function `gen_alarm` is invoked. This function configures a new ALARM packet, with the address of the 2 congested routers.

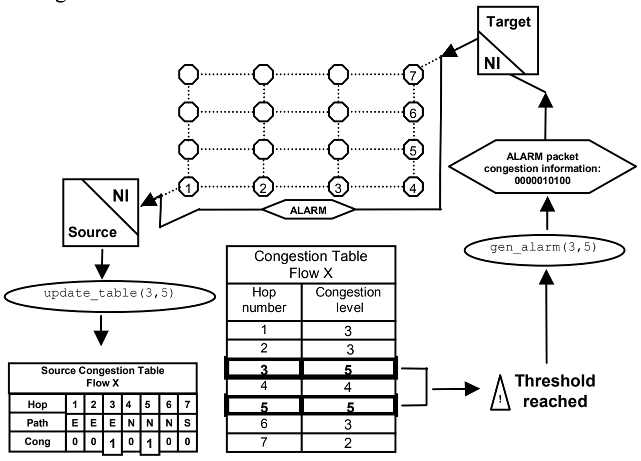


Figure 5 - Congestion Detection.

This ALARM packet is detailed in Figure 6. As routers 3 and 5 are congested, bits 3 and 5 of the *congestion information* field are asserted. The source NI receive this packet and update the `cong` field of *Source Congestion Table*. Note in Figure 6 values '1' in hops 3 and 5.

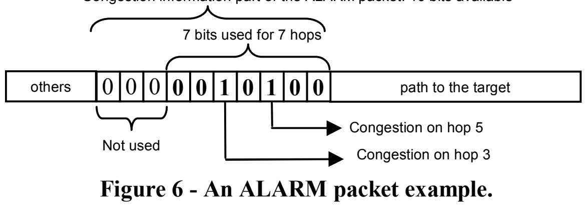


Figure 6 - An ALARM packet example.

The definition of the new path follows two assumptions: (*i*) minimal routing, and therefore all paths have the same number of hops to the target; (*ii*) the new path should use, if it is possible, routers near to the oldest path, to minimize the impact of the new flow in other existing flows.

The proposed heuristic to compute the new path to the target uses two vectors to store the congestion states. The X and Y direction vectors store the congestion state for each NoC column and line, respectively. Figure 7 shows the relationship between these vectors in a 5x5 mesh.

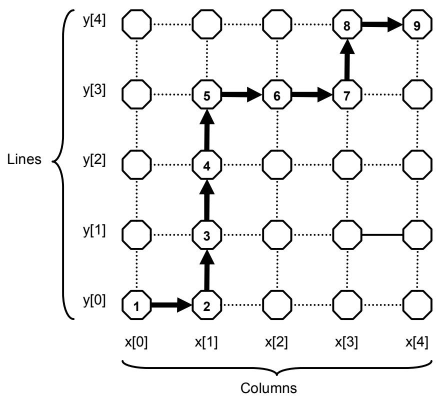


Figure 7 - Example of a path in a 5x5 mesh.

Table 1 presents the source congestion table, with hops '3', '6', 7' and '8' presenting congestion. Figure 8(a) presents the congested routers shadowed and the lines and columns that are congestion free. Vectors X and Y are filled by counting the number of congested routers at each column and line, respectively.

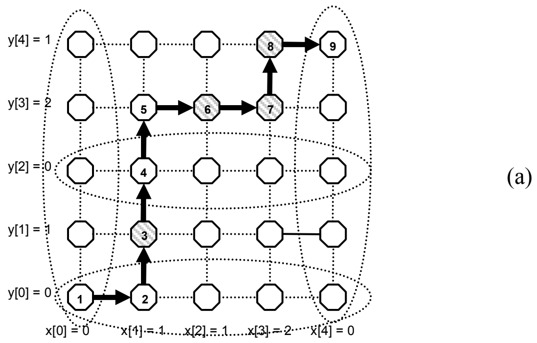
Table 1 - Example of Source Congestion Table, router 00.

Hop	1	2	**3**	4	5	**6**	**7**	**8**	9
Path	E	N	**N**	N	E	**E**	**N**	**E**	**W**
Cong	0	0	**1**	0	0	**1**	**1**	**1**	0

The resulting vectors of the example presented in Figure 8(a) are: X = [0 ; 1 ; 1 ; 2 ; 0] and Y = [0 ; 1 ; 0 ; 2 ; 1].

The proposed heuristic to define the new path to target seeks in an *alternatively* way in both X and Y vectors, the first index equal to '0'. When the first '0' is found, the source router is connected to router index having this value. If the first '0' is found in the X vector, a horizontal connection is implemented; otherwise a vertical connection is created when this '0' is found in the Y vector. The next step is to repeat this process for the next '0' value (which may belong to an X or Y direction vector), creating a new partial path between the previous router to present one.

Consider the vectors X and Y of the previous example. If `x[0]` and `y[0]` are different to zero, it means that the first hop of the routing path is already a congested one. In this case, the first direction of the new path is the opposed to the original one. That is, if the old first direction is in *x* axis, the new one will be on the *y* axis, and vice-versa. As shown in Figure 9, the first '0' is found at y[2]. Thus, it will be added on the new path 2 routings on the North direction. The next '0' found is in the position x[4]. In this way, it will be added 4 East directions on the path.



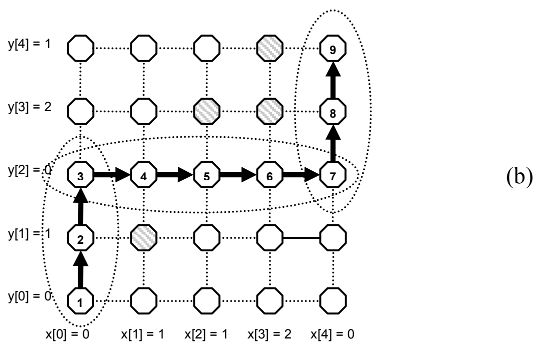
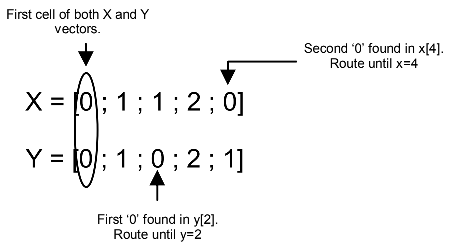


Figure 8 - (a) Congestion points, free lines and columns highlighted; (b) New path to the target computed.

At the end of the process, if the target router is not reached, the path must be completed until it. Note that the target router does not present congestion, since it is receiving the flow. In this way, that partial path always reaches the line or column of the target router, being the complete path obtained with a vertical or a horizontal segment. Therefore, the path will be completed with 2 routings on the North direction and 1 on the Local direction. This is coherent with the congestion-free path principle, once these two routing on North direction are made on a column without congestion points. Figure 8(b) illustrates the new path to the target computed.



First cell of both X and Y vectors.
Second '0' found in x[4]. Route until x=4
X = [0 ; 1 ; 1 ; 2 ; 0]
Y = [0 ; 1 ; 0 ; 2 ; 1]
First '0' found in y[2]. Route until y=2

Figure 9 - Illustration of new path computation taking as reference congestion vectors X and Y.

Opposed to distributed routing, where the path to the target is defined hop per hop, source routing define the complete path at once, avoiding deadlock, starvation and livelock phenomena. To prevent deadlock, the turns in the path obey the turns allowed in partial-adaptive routing algorithms, as west-first. To be completely adaptive, 2 virtual channels can be used, being the first one using west-first routing, and the second one a symmetric one, as east-first.

Before the adoption of the new path, a last DATA packet is sent in order to clean the congestion tables that belonging to each intermediate router of the old path. The new path is thus initialized with a DATA packet to open a new routing session, identifying the routers of this path. Considering that minimal routing is adopted, the TCT on the target remains with the same size.

5. RESULTS

The set of experiments uses the asynchronous NoC platform presented in [11]. Implementation of the methods presented was made in SystemC TLM. This communication structure has the main following main features: (i) source routing; (ii) wormhole switching; (iii) 32-bit fixed size bits; (iv) end-to-end credit based flow control. Each packet is composed of a header flit and, depending on the type of packet, a number of flits, which carry application and configuration data. The header flit brings information about routing and commands for the units and network interfaces.

The complement traffic pattern is the communication spatial model used in the experimental setup, where each node sends data to the one located at its opposed side. Figure 10 illustrates the NoC elements positioned on the 5x5 mesh. In the middle of the NoC, there is a node connected to a CPU controller, which activates each unit to start communication.

Two types of traffic generators and receptors are defined, as shown in Figure 10. *QoS traffic flows* are those which have performance requirements. The network interface for each unit that generates QoS traffic executes the method of dynamic routing presented in Section 4. These flows are generated by the nodes labeled as **Q**. *Noise traffic flows* are those used as background traffic, and do not make considerable perturbations on the QoS flows. These flows are generated by the nodes labeled as **N**. Each unit transmits 1600 5-flit packets, at a clock frequency of 50 MHz. Initially, all flows execute the XY routing algorithm.

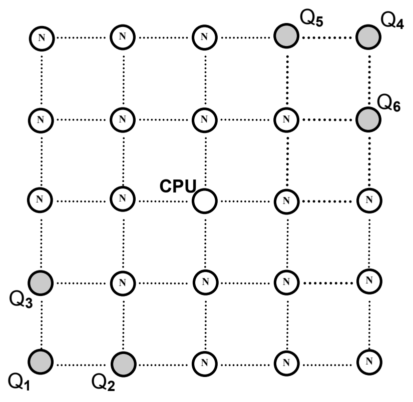


Figure 10 – Nodes connected to QoS units and to the CPU controller.

5.1 LATENCY EVALUATION

The first evaluated performance figure is the latency, which is obtained for each flit, being the time spent between the transmission and reception of a given flit at the target node. This analysis considered the latency only for the QoS flows, generated by the IPs highlighted in Figure 8 (a). Table 2 presents the average latency values obtained and their standard deviation, in ns.

Table 2 - Average and standard deviation latency values, in ns.

	without the proposed method		with the proposed method		Reduction (-) or increasing (+) on standard deviation?
	Average latency	Standard deviation	Average latency	Standard deviation	
$Q_1 \rightarrow Q_4$ (9 hops)	25.14	**8.81**	24.98	**8.29**	**- 5.9%**
$Q_4 \rightarrow Q_1$ (9 hops)	25.33	**9.19**	25.02	**8.50**	**- 7.5%**
$Q_2 \rightarrow Q_5$ (7 hops)	23.42	**11.29**	23.14	**10.75**	**- 4.8%**
$Q_5 \rightarrow Q_2$ (7 hops)	23.41	**11.06**	23.16	**10.68**	**- 3.4%**
$Q_3 \rightarrow Q_6$ (7 hops)	20.33	9.05	20.49	9.10	+ 0.5%
$Q_6 \rightarrow Q_3$ (7 hops)	20.43	**8.98**	20.48	**8.93**	**- 0.6%**

Even if the values for average latency in both approaches are similar, it is important to note the reduction on the standard deviation on the major part of the QoS flows. The packet latency standard deviation is an important metric for systems with QoS constraints. Packets originated from a given source must keep a regular interval between them to respect the injection rate. Variation in the latency incurs in jitter, which can lead to packet loss in applications with strict temporal deadlines like real-time applications.

Note also that the major reduction on the packet latency standard deviation occurs on the flows with the higher number of hops (9 hops), which shows the efficiency of the proposed method.

5.2 BUFFER OCCUPATION

This Section considers for buffer occupation analysis the different regions of traffic load. A router is attributed to a given region according to its distance to the CPU controller, which may be A, B or C. These three regions are shown on Figure 11.

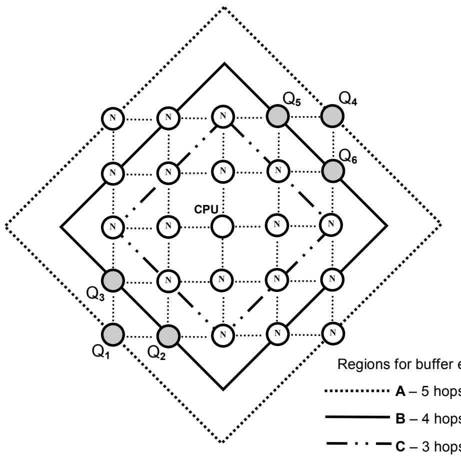


Figure 11 - Network regions for buffer occupation analysis, defined according to the distance to the CPU.

Table 3 presents results for buffer occupation. In this work, the number of buffered requests for routing in each network node is considered. Note that data concerning occupation after 15ms is shown, because this is the moment when starts the routing adaptability. The total simulation time is 25ms. In the same way of the latency analysis, the major contribution of the algorithm is in the standard deviation on buffer utilization, in particular for the buffers belonging to the region A (more distant from the CPU).

Table 3 - Results for buffer occupation, in number of routing request packets.

			without the method proposed	**with the method proposed**
Region A (5 hops to the CPU)	Total time	Average	1.085	1.088
		Std. dev.	0.057	0.053
	From 15 ms	Average	1.073	1.079
		Std. dev.	**0.030**	**0.015**
Region B (4 hops to the CPU)	Total	Average	1.289	1.286
		Std. dev.	0.234	0.229
	From 15 ms	Average	1.306	1.299
		Std. dev.	**0.253**	**0.246**
Region C (3 hops to the CPU)	Total	Average	1.357	1.352
		Std. dev.	0.248	0.244
	From 15 ms	Average	1.422	1.414
		Std. dev.	**0.241**	**0.237**

Figure 12 shows the buffer use for the duration of the simulation for the A region (5 hops). It can be observed the buffer use reduction after 15ms using the proposed routing method. The original values are displayed in black, while the results obtained with the proposed routing method in gray. This experiment demonstrates that the buffers utilization on these regions is more balanced, making a better use of the available resources.

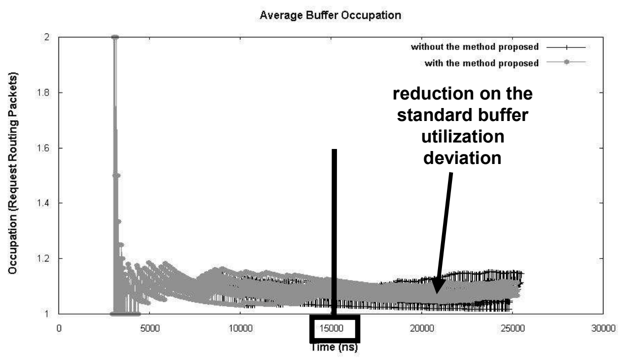


Figure 12 - Buffer utilization in different simulation periods.

6. CONCLUSIONS AND FUTURE WORK

The original contribution of this research work is a new method for adaptive dynamic routing to be used in networks on chip. This method takes routing decisions based on the congestion path of each QoS flow, bringing to the routing algorithm a global view of the path being routed, not only neighbors routers status, as the state of the art proposals. Results present reduction on the packet latency standard deviation, important parameter for QoS flows, and smaller buffer utilization. The major reductions occur with flows with a higher hop count, which highlights the benefits of the proposed method for long paths.

The proposed approach will be compared to other routing algorithms (which can use source or distributed routing) in a near future. Evaluation in terms of area and power are also part of future work. Other future works include: (1) analysis of the benefits when some routers of a path are selected to perform monitoring, instead of all routers, strategy that it is adopted today; (2) extension of the proposed routing algorithm, considering the weight of congestion (in the present implementation, a new path is considered free if its lines and columns have cong values equal to zero); (3) include new metrics for congestion beyond buffer occupation, e.g., throughput; (4) perform experiments using other traffic patterns and real traffic scenarios.

7. ACKNOWLEDGMENTS

This research was supported partially by CNPq (Brazilian Research Agency), project 300774/2006-0 and 471134/2007-4.

8. REFERENCES

[1] Tedesco, L. et al. "Application Driven Traffic Modeling for NoCs". In: SBCCI'06, pp. 62-67.

[2] Al Faruque, M.A. et al. "ROAdNoC: Runtime Observability for an Adaptive Network on Chip Architecture". In: ICCAD'08, pp. 543-548.

[3] Ogras, U. Y.; Marculescu, R. "Analysis and Optimization of Prediction-Based Flow Control in Networks-on-Chip". ACM Transaction on Design Automation of Electronic Systems, v.13(1), 2008, article 11, 28p.

[4] Manolache, S. et al. "Buffer Space Optimisation with Communication Synthesis and Traffic Shaping for NoCs". In: DATE'06, pp. 1-6.

[5] Mello, A. et al. "Rate-based Scheduling Policy for QoS Flows in Networks on Chip". In: VLSI-SOC 2007, pp. 140-145.

[6] Li, M. et al. "DyXY - A Proximity Congestion-aware Deadlock-free Dynamic Routing Method for Network on Chip". In: DAC'06, pp. 849-852.

[7] Hu, J.; Marculescu, R. "Dyad – Smart routing for networks on chip". In: DAC'04, pp. 260-263.

[8] Lotfi-Kamran, P. et al. "BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs". In: DATE'08, pp. 1408-1413.

[9] Al Faruque, M. A. et al. "Run-time Adaptive On-chip Communication Scheme". In: ICCAD'07, pp. 26-31.

[10] Gratz, P. et al. "Regional Congestion Awareness for Load Balance in Networks-on-Chip". In: HPCA'08, pp. 203-214.

[11] Lattard, D. et al. "A Reconfigurable Baseband Platform Based on an Asynchronous Network-on-Chip". IEEE Journal Of Solid State Circuits, v. 43(1), 2008, pp 223-235.