Efficient Traffic Balancing for NoC Routing Latency Minimization

João M. Ferreira, Jarbas Silveira, Jardel Silveira
LESC-DETI-UFC
Federal University of Ceará
Fortaleza, Brazil
jarbas@lesc.ufc.br

Rodrigo Cataldo, Thais Webber, Fernando G. Moraes, César Marcon
PPGCC-PUCRS
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Brazil
cesar.marcon@pucrs.br

***Abstract*— Modern technologies of integrated circuits allow billions of transistors arranged into a single chip, enabling to implement complex systems, which need a scalable and parallel communication architecture. Network-on-Chip (NoC) is a natural candidate to fulfill such communication requirements, providing high performance when the communication demands are balanced. This work proposes a new static balancing method that uses the application's traffic pattern for NoC latency reduction. This method allows the generation of a deterministic routing algorithm with simplistic implementation and low latency. Experimental results compare four balancing methods, showing the improvement of the proposed static balancing concerning the average NoC latency.**

Keywords— NoC; irregular topology; routing methods

1. Introduction

The evolution of VLSI semiconductor technology enables to integrate hundreds of cores into a single circuit. This massive integration allows implementing the entire functionality of a system into a single chip producing a System-on-Chip (SoC). The International Technology Roadmap for Semiconductors (ITRS) foresees hundreds of Processing Elements (PEs) integrated into a SoC by 2020 [1]. A Network-on-Chip (NoC) [2] plays a key role in the communication of these highly integrated SoCs, with the two-dimensional (2D) mesh as the most popular NoC topology, offering simple and regular structure for tile-based design [3].

The NoC performance can be maximized through the traffic balancing on links and routers. Adaptive routing algorithms can provide this balancing, which is adjusted dynamically to traffic variation. However, adaptive algorithms present several limitations for broad usage. For instance, the ones that only use local traffic information can characterize the overall NoC state incorrectly, which can lead to a new packet routing where the performance is even worse than the unbalanced version. Besides, adaptive algorithms that consult the global NoC state require a complex mechanism consuming much area and energy, and sometimes, when the state information reach its destination, a new traffic scenario may be stablished. Therefore, it is desirable to design a solution by traffic balancing without considerable overhead of area and energy, and with low delays when deciding new routes. Such solutions must require less runtime computations using global NoC state (i.e., ideally predicting traffic through all PEs).

Traffic patterns of several embedded applications executing on specific target architectures are known at design time. Therefore, it is possible to devise a routing algorithm where the routes are accomplished deterministically, having a simpler implementation where all computation is performed at design time. We propose the **Specific** balancing method that takes into account the sum of all bits that traverse links and routers in a giving interval. This method, which is a static traffic balancing on NoC resources, is compared with three others: (i) a static random selection of routes (**Random** balancing); (ii) a static **Uniform** balancing that takes into account the quantity of communication flows (i.e., communications between pairs of PEs); and (iii) a **Dynamic** balancing that adapts the path according to local traffic information.

The present work is divided as follows: Section 2 describes the basics of traffic balancing, which includes path selection and scenarios description. Section 3 presents the experimental setup and Section 4 discusses the experimental results. Section 5 draws the conclusions of this paper.

2. Basics of Traffic Balancing

The NoC traffic is imbalanced when some network resource is highly demanded whereas other is underused. This imbalance can produce extra packet latency, and overheating on links and routers, for instance. The traffic balancing in all NoC resources reduces the peak temperature and the temperature variation across the chip [4]. Moreover, links and routers located in the center of a mesh NoC are usually much more congested than others are, and, therefore, more likely can suffer temporary or permanent failure.

Efficient routing algorithms aim to balance the traffic between all the NoC links, minimizing competition for resources and hence the packet latency [5]. Thus, non-minima routing algorithms are often required to balance traffic because it increases the number of paths, increasing the freedom of the routing algorithm. Additionally, even with the augment on power consumption and complexity, the overall network's performance increases significantly [6].

The traffic balancing aims to equalize a weight criterion (e.g., quantity of bits) on the NoC links and routers. Methods that assume the amount of routing paths that pass through a link as a weight criterion are affordable for traffics with uniform pattern; otherwise, they may not result in the expected performance. This paper proposes the **Specific** traffic balancing metric that considers the weight criterion as the sum of all volumes of communications that pass through each NoC resource, along with a priority criterion to select the communicating pair of PEs that are candidates to use the referred resource. The volume of communication, which can be normalized for convenience, is the amount of bits that a PE sends to another PE in a given time interval. This approach uses the application traffic pattern as the main aspect to define the routing algorithm. This proposed metric is justifiable since pairs of PEs, with different communication volumes, influence differently the

978-1-4799-5341-7/16/$31.00 ©2016 IEEE

links and routers through which they pass, and this should be considered to balance the traffic effectively.

The routing mechanism defines how the routing information is computed and stored, which may encompass tables placed in each NoC router, storing the routing information. Routing tables support many topologies and are easy to implement. However, routing tables containing the addresses of all PEs do not scale with the NoC size. As proposed by Mejia et al. [7] in the Region Based Routing (RBR) approach, the scalability may be reached dividing the NoC into regions enclosing sets of PEs and mapping these regions in the routing table entries. RBR could be employed with a specific routing application, enabling the use of a small number of regions and assuring full network coverage.

This work employs RBR in our target NoC architecture since this mechanism enables to produce routings statically defined at the design time, but also it can be configured to adapt the routing path dynamically. This features are used to explore the four traffic balancing methods describe next.

Figure 1 shows that the application traffic (i.e., traffic pattern and traffic injection rate) and the NoC architecture are inputs of the three main steps employed on the explored balancing methods. Firstly, all methods execute the Segment Based Routing (SBR) algorithm [8], which is the step that generates restrictions of some paths to provide only deadlock free paths.

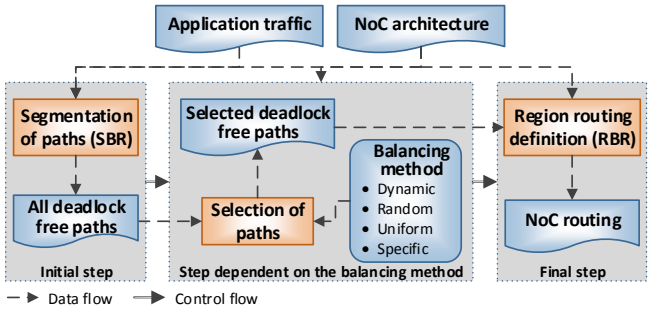

Figure 1 – Sequence of steps employed on four balancing methods.

The deadlock free paths are inputs for the algorithm of paths selection, which iterates over all communicating pairs of PEs generating the set of all minima routing paths. These sets are sorted in ascending order according to the number of paths have each communicating pair of PEs. Consequently, communicating pairs with lower diversity of paths come before the others, resulting in a priority for choosing the communication paths. The selection of paths depends on the traffic scenario, which differs regarding the way in which the traffic is balanced. The differences between each balancing method is described next:

- In **Dynamic** balancing, the routing mechanism of the router can explore all shortest paths between each communicating pair of PEs. Consequently, all paths are provided to RBR, which are selected at runtime by the routing mechanism according to the traffic load of each link that is perceived by the router (i.e., a local traffic vision);
- The **Random** balancing is a method with low computational cost that produces static random selection of paths and deterministic routing. Here, we generate five minima routing paths for each communicating pair of PEs. These routing paths are delivered to the RBR algorithm to observe the effect of random choices. Therefore, each random choice implies a new system simulation;
- The **Uniform** balancing performs a static and deterministic routing. The path with the smallest weight is selected providing a uniform distribution. The algorithm updates the links and routers' weight through which the selected path passes, increasing their weights unit by unit. Consequently, the RBR receives a reduced and optimized quantity of entries that are stored into the routers' routing table;
- The **Specific** balancing performs a static and deterministic routing. It is similar to the **Uniform** balancing; however, it captures the traffic pattern better, enabling to produce an optimized global routing. In the first iteration, for each communicating pair of PEs (starting from the nearest communicating pairs to the distant ones), the algorithm selects the path with lowest communication weight, updating the weight of all links and routers of the chosen path with the communication volume. This procedure can change the communication weight of the previous selected path, conducting to an inefficient choice. Aiming to avoid it, the algorithm makes iteratively the same procedure performed in the first iteration until the convergence of the standard deviation of the average of the NoC resources' weight. Note that this convergence value indicates the homogeneity of the global communication volume that are distributed through the NoC resources (routers and links). Therefore, when the standard deviation is minimum (ideally zero) the traffic is perfectly distributed into all NoC resources. The convergence of the algorithm produces a set of deadlock free paths that is delivered to the RBR step.

In the final step, the RBR algorithm receives a set of selected paths (according to each one of the four balancing methods) and generates the routing tables for each NoC router.

3. Experimental Setup

Figure 2 shows the experimental setup employed to evaluate and compare the performance of the four balancing methods applied to seven synthetic traffic patterns. In addition, we employed 37 traffic injection rates to explore details of low rates and the NoC saturation point. The experimental results encompasses 2072 simulations with two sizes of the NoC Phoenix [9], which is implemented in VHDL with the following configuration: regular mesh topology; 16-flits of buffering in each input port; wormhole switching; on-off control flow; round-robin arbitration and distributed tables filled with RBR approach (with 16 regions for each router) to implement the routing algorithm.

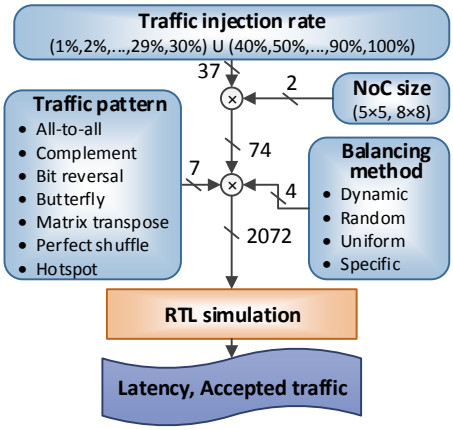

Figure 2 – Summary of the experimental setup.

The routing algorithms for each scenario were generated using

the RBR technique. The router performs the routing function by consulting the routing tables in the format described in [7], which contains the set of input ports, at least two vertices defining the region, and the set of output ports. Furthermore, the routing algorithm provides routing adaptability when there are multiple options: a selector (fixed priority) that selects one of the output ports are idle (returned as a table). If all source/destination pairs of PEs have only a single routing option, then the implemented algorithm is deterministic.

The spatial distribution of traffic was defined as all-to-all, complement, bit reversal, butterfly, matrix transpose, perfect shuffle, and hotspot. Aiming to vary the traffic injection rate at each simulation, we change the number of clock cycles between consecutive packets. In addition, warming up and cooling down technique was used. Therefore, each PE sends 20% more traffic at the beginning and the end of the simulation (10% each), which are completely overlooked in the evaluation phase.

4. Experimental Results

A. *Average Latency and Accepted Traffic with Random Balancing Method*

The first set of experiments explores the average latency and the accepted traffic according to several injection rates, whose traffic is balanced using the **Random** method. This experiment takes into account five random selections of routing paths and all-to-all traffic. Figure 3 shows some results: (i) the NoC start to saturate (not all injection rate are accepted) around of 20% of injection rate. However, the complete traffic saturation is reached near of 50% of injection rate; (ii) with small injection rate (i.e., less than 20%) the Random balancing attains low latencies, in average.

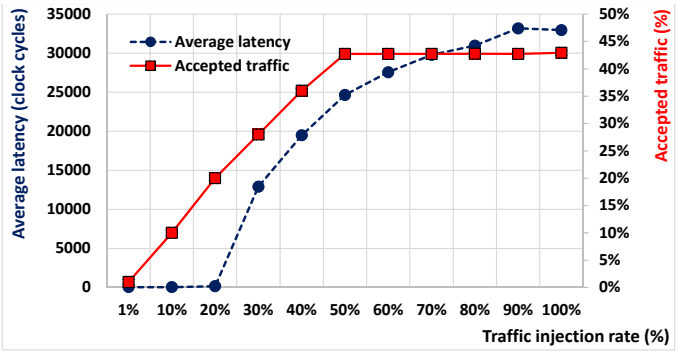

Figure 3 – Network latency and saturation point according to the traffic injection rate under a synthetic all-to-all traffic.

B. *Comparing Random, Dynamic and Uniform Balancing Methods According to the Traffic Latency*

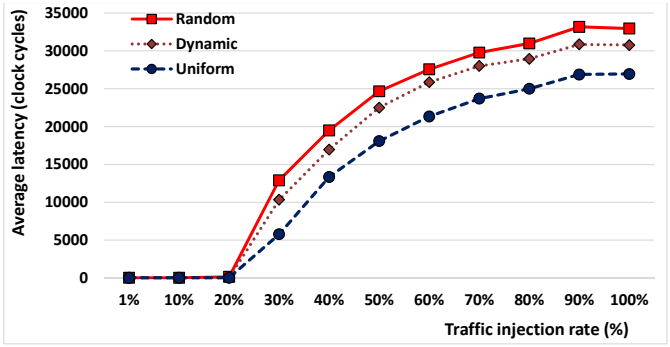

Figure 4 – Random, Uniform, and Dynamic balancing methods for several traffic injection rates and all-to-all traffic pattern.

The next experimental result compares **Random**, **Dynamic** and **Uniform** balancing methods. Figure 4 displays that (i) until reaching the saturation point, all evaluated methods have similar latencies, in average; and (ii) when the traffic injection rate exceeds the saturation point, the **Uniform** balancing is the most efficient method, followed by the **Dynamic** balancing.

C. *Influence of Traffic Pattern on Uniform Balancing*

The next set of experiments evaluates the performance of the **Uniform** balancing according to seven synthetic traffic patterns. Figure 5 shows that the traffic pattern affects the latency meaningfully when using **Uniform** balancing. To minimize this effect, this work proposes a more specialized balancing method for traffic patterns known in advance. The next experiments illustrate the performance of the proposed method.

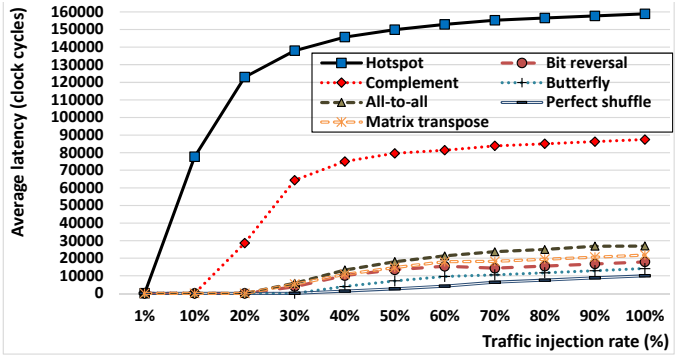

Figure 5 – Average latency for seven synthetic traffic and Uniform balancing.

D. *Comparing Uniform and Specific Balancing Methods According to the Traffic Latency*

Figure 6 illustrates an expressive latency reduction when using **Specific** balancing compared to the **Uniform** balancing method, showing the capacity of the **Specific** method in capture the traffic behavior. This experiment explores the average latency of synthetic applications with predefined traffic patterns according to the increase of traffic injection rate.

Aiming to detail the effect of each traffic pattern, Figure 7 shows the latency reduction achieved by the **Specific** balancing, when compared to **Uniform** method, for the seven traffic patterns with 1% and 100% of traffic injection rates.

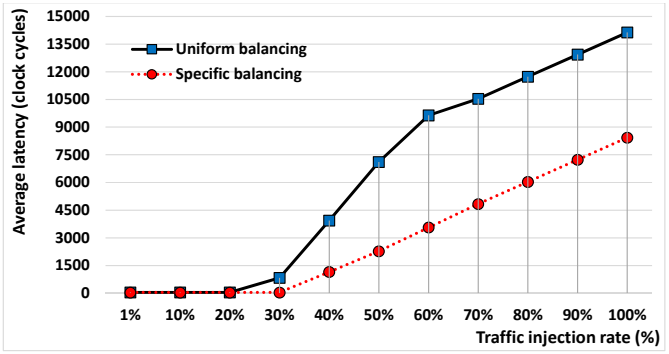

Figure 6 – Efficiency of Specific balancing compared with Uniform method for synthetic traffic patterns and several injection rates.

Figure 7 brings the following conclusions: (i) the latency gains are more expressive with the increase of traffic injection rate. The only exception is observed in the complement pattern; (ii) the latency gains are highly dependent on the traffic pattern. For

instance, the Specific balancing reaches 51% of latency reduction for butterfly pattern (100% of traffic injection rate), whereas the balancing methods produce the same latency for hotspot pattern, independent of the traffic injection rate.

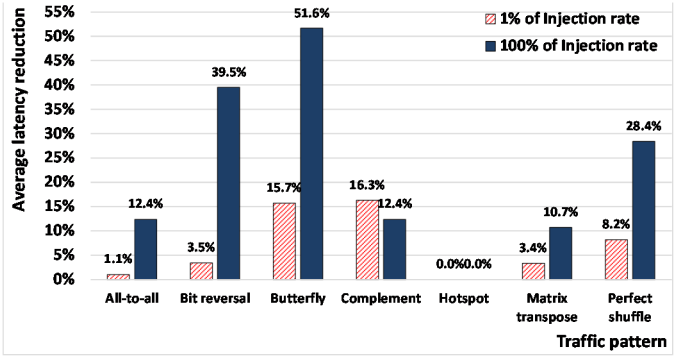

Figure 7 – Average latency reduction for Specific balancing with seven patterns of traffic and two injection rates.

E. Comparing Uniform and Specific Balancing Methods According to the Traffic Throughput

The following experiment compares the quality of the **Uniform** and **Specific** traffic balancing methods for the seven synthetic traffic patterns. The experiment aims to quantify how much the balancing method increases the accepted traffic during the full range of injection rates. Note that the accepted traffic is the throughput measurement of the network *input* traffic, and this measurement represents a sound estimation of the throughput average of the network *output* traffic, if the application executes during a long simulation period. Additionally, the values of the accepted traffic of each pattern are grouped and represented as an average.

Figure 8 shows that the two balancing methods attain similar accepted traffic rates to less than 20% of injection rate. This similarity is illustrated with two overlapping line of 45 degrees, meaning that all traffic injected is accepted. The **Specific** balancing preserves this same behavior until around of 30% of injection rate, meaning that the **Specify** balancing postpones the NoC saturation point. Additionally, for high injection rates (more than 70%) both methods attain the same accepted traffic.

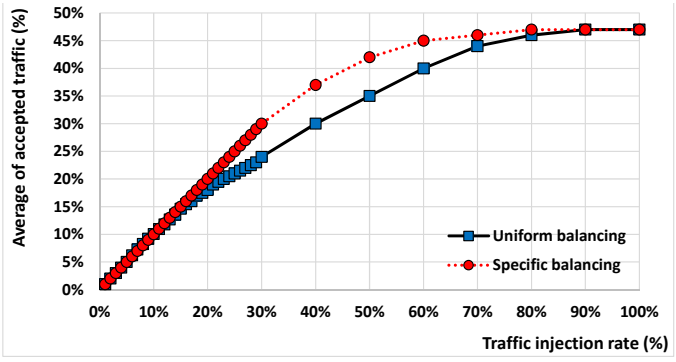

Figure 8 – Comparison of Uniform and Specific balancing methods according to the average of the accepted traffic.

5. CONCLUSIONS

This work analyzes the performance of four methods of NoC traffic balancing: (i) **Random** balancing with static path selection; (ii) **Dynamic** balancing performed by an adaptive routing algorithm that employ only local traffic information; (iii) **Uniform** static balancing, which equalizes the number of paths per link; and (iv) **Specific** static balancing, which equalizes the communication volume that passes through a link.

We proposed a **Specific** static balancing that combines the global network load information (i.e., communication volume and direction of all communicating pair) with the simplicity of implementation of a deterministic routing algorithm to distribute traffic between network resources efficiently. The experimental results showed that the **Random** balancing for generating a deterministic routing algorithm could severely degrade the overall NoC performance. Therefore, its low computational cost does not justify its adoption. Likewise, the static traffic balancing is more efficient than **Dynamic** balancing when employing global traffic information, in general. Besides, the results indicate that **Specific** balancing for known traffic pattern minimizes the NoC latency meaningfully.

Finally, the **Specific** and **Uniform** traffic balancing tend to reduce the average latency, both before and after the saturation point. Moreover, there is a tendency to increase the network throughput after NoC saturation, and a displacement of the saturation point toward a higher traffic injection rate.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS). 2013 Edition. Available in: www.itrs.net/reports.html.

[2] R. Marculescu et al. **Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, n. 1, pp. 3-21, Jan. 2009.

[3] S. Rodrigo et al. **Cost Efficient On-Chip Routing Implementations for CMP and MPSoC Systems**. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, n. 4, pp. 534-547, Apr. 2011.

[4] C. Nicopoulos. V. Narayanan. C. Das. **Network-on-Chip Architectures - A Holistic Design Exploration**. vol. 45, *Springer*, 2010, 209p.

[5] J. Flich et al. **A Survey and Evaluation of Topology-Agnostic Deterministic Routing Algorithms.** *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, n. 3, pp. 405-425, Mar. 2012.

[6] W. Dally, B. Towles. **Principles and Practices of Interconnection Networks**. *Morgan Kaufmann Publishers Inc.*, 2003.

[7] A. Mejia et al. **Region-Based Routing: A Mechanism to Support Efficient Routing Algorithms in NoCs**. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, n. 3, pp. 356-369, Mar. 2009.

[8] A. Mejia et al. **Segment-based routing: an efficient fault-tolerant routing algorithm for meshes and tori**. *International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 25-29, 2006

[9] C. Marcon et al. **Phoenix NoC: A distributed fault tolerant architecture**. *IEEE International Conference on Computer Design (ICCD)*, pp. 7-12, 2013.