Arbitration and Routing Impact on NoC Design

Edson I. Moreno, Cesar A. M. Marcon, Ney L. V. Calazans, Fernando G. Moraes
Faculty of Informatics, PUCRS, Porto Alegre, Brazil
{edson.moreno, cesar.marcon, ney.calazans, fernando.moraes}@pucrs.br

Abstract - **The increasing number of processing elements packed inside integrated circuits requires communication architectures such as a Networks-on-Chip (NoCs) to deal with scalability, bandwidth and energy consumption goals. Many different NoC architectures have been proposed, and several experiments reveal that routing and arbitration schemes are key design features for NoC performance. Therefore, this work proposes a routing scheme called planned source routing, which is implemented in a NoC architecture with distributed arbitration called Hermes-SR. The paper compares Hermes-SR to the Hermes NoC that employs distinct arbitration and routing mechanisms and algorithms. One set of experiments enables to confront design time planned source routing and runtime distributed routing. Additionally, the paper presents the advantages of using deadlock free adaptive routing algorithms as basis for balancing the overall communication load in both routing mechanisms. Another experiment reveals the tradeoffs between using centralized or distributed arbitration. A last evaluation exposes the performance advantages of combining distributed arbiters with planned source routing. Results enforce that design time planned source routing tends to avoid NoC congestion and contributes for average latency reduction, while distributed arbitration optimizes NoC saturation figures.**

I. Introduction

THE growing density of transistors per silicon unit area enabled the implementation of a complete system on a single die, the so called System-on-a-Chip (SoC). SoCs target high performance with small footprint and low energy consumption when compared to the same system implemented by an equivalent set of chips. A SoC is composed by a possibly large amount of processing elements (PEs), dozens or even hundreds interconnected by an on chip communication architecture. As the complexity of applications fitting inside a single SoC raises, scalability and flexibility are achieved through the use of multiprocessor systems on chip (MPSoCs), a special case of SoCs where most or all PEs are programmable processors [1], increasing the SoC architecture flexibility.

SoC and MPSoC designs rely on the massive reuse of predesigned PEs. The communication architecture, on the other hand is specifically built to fulfill the application requirements, making the design of these components communication centric. Traditional communication architectures such as shared busses and those based on dedicated point to point interconnections do not scale well with the ever-growing amount of parallel data transmission [2]. Dedicated point to point links lead to communication architectures that are difficult to reuse and enhance in subsequent design revisions. Busses may become bottlenecks, increasing latency and power dissipation. Despite the fact that hierarchical busses do support parallel communications, scalability suffers and contention increases when communication between PEs located at different sides of a bridge is needed. NoCs are currently considered as a better approach for enhancing scalability and power dissipation efficiency [3].

The design of NoC-based MPSoCs must take into account several communication architecture aspects, including topology, buffer dimensioning, output selection (i.e. routing algorithms) and input selection (i.e. arbitration algorithms). In this paper, a *NoC* is a communication architecture composed by a set of switching elements called *routers* that employ packet switching communication. Routers interconnected by links form the NoC topology. Some or all routers may also connect to PEs through a network interface, which is not itself considered part of the NoC. The basic function of routers is to monitor incoming packets from its input ports, select one output port and forward packets through some internal path. Arbitration and routing orchestrate router internal resources access/priority and directions decision.

The routing behavior may be either a deterministic or an adaptive function. Deterministic routing defines the output port that a packet will take based on its source and destination, irrespective of traffic characteristics. An example of a NoC applying runtime deterministic routing is Hermes [3]. Adaptive routing allows considering more than one candidate output port for a given input port at each router, which can be important for performance, congestion control and fault tolerance, as decisions may consider the instantaneous network status [4]. An example of a NoC architecture proposing the use of runtime adaptive routing is DyAD [5].

Since each router deals with several simultaneous requests to forward packets, an arbitration strategy is necessary. Two choices are to deal with one request at a time, called *centralized arbitration*, or to deal with a set of requests in parallel, called *distributed arbitration*. These choices lead to the generic router architectures depicted in Figure 1. In both arbitration strategies competition for router resources may occur.

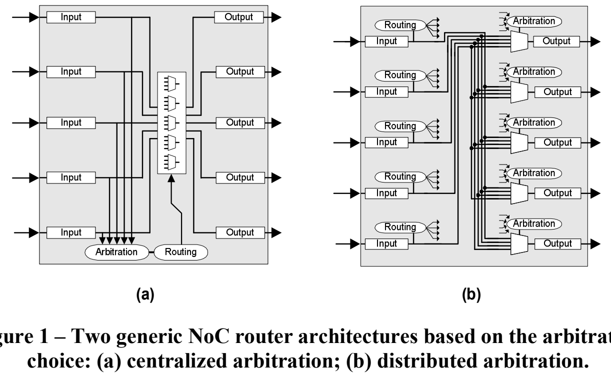

Figure 1 – Two generic NoC router architectures based on the arbitration choice: (a) centralized arbitration; (b) distributed arbitration.

Centralized arbitration produces routers which are simpler, while distributed arbitration trades increased router complexity for enhanced performance [6]. Centralized arbitration usually implies that the router contains only one single routing unit, for which all input ports compete. Arbitration and routing define a connection between an input and an output port, after which transmission in that connection starts and the routing unit is released to serve other pending input port requests. Distributed arbitration, on the other hand, implies that competition for resources occurs only at the output ports. This requires some hardware unit replication at the input and output ports (routing and arbiters, respectively), but may increase performance dramatically. The results in this paper support this statement.

The automated design of NoCs may quickly produce a generic communication architecture solution. However, NoC/SoC silicon area, power dissipation and performance may be optimized if architecture configuration and usage are planned [7]. This is especially true when the application communication patterns are known in advance. Considering this situation and restricting attention to path selection among communicating pairs for source routing NoCs, it is usual to employ bandwidth limits as a criterion to define a set of communication routes [8] [9]. Bandwidth limits are an efficient way to spread the overall link load. However, using only this criterion may result in communication loads that are badly distributed in time, which may in turn compromise the overall NoC performance. Moreover, the advantage of design time path selection when compared to runtime distributed routing algorithms is not clear. This paper reports isolated and joint comparisons between *source* versus *distributed routing* and *centralized* versus *distributed arbitration* strategies.

Remember that most, if not all, routing algorithms used in NoCs are deadlock free because they preclude the use of some routes from source to destination. Often, the justification to use adaptive routing algorithms in place of deterministic ones comes from the capacity of the former to avoid contention by using alternate routes when a conflict occurs at runtime. This paper values another important property of adaptive routing algorithms, which is the richer set of possible routes a packet can use to go from source to destination. This property makes sound combining source and adaptive routing mechanisms when traffic patterns are known at design time. The richer set of routes facilitates overall load balancing in the communication architecture, while deadlock freedom guarantees the integrity of operation for the communication architecture.

The remainder of this paper is organized as follows. Section II presents the process adopted for route mapping coupled to the Hermes-SR NoC. Section III depicts the Hermes-SR architecture. Section IV describes the experimental setup and results, while Section V displays a set of conclusions and directions for future work.

II. ROUTE MAPPING

The definition of communication routes may be guided by different requirements, aiming at power dissipation, area or performance. This work considers as key requirement communication performance, measured by the reduction of potential congestion, and evaluated by average packet latency. Basically, congestion is detected when the amount of incoming data (or request for incoming data) is larger than the outgoing data from a given communication element (e.g. a router). A reason for this is a bad distribution of communication flows, which may imply overloaded channels, called *hotspots*. To avoid hotspots, it is mandatory to: (i) *explore alternative paths* for each communicating modules, (ii) *combine paths of communicating pairs* in a traffic scenario and (iii) *evaluate route mappings* for each traffic scenario.

A. Exploring Alternative Paths

A path is defined here as the sequence of router output ports used to transmit packets from a source to a destination. Depending on the routing algorithm, more than one alternative path may exist between a given source and destination. The exploration of alternative paths has to guarantee that at least one path exists among each communicating pair and that no deadlock will occur when paths are combined into traffic. There are two ways to obtain deadlock freedom: (i) through formal verification or (ii) through the adoption of deadlock-free routing algorithms as basis for path computation. The present work uses the second approach. It employs four different routing algorithms, Pure XY (XY), and the three turn model variations: Negative First (NF), West First (WF) and North Last (NL) [4]. The first is deterministic, while the remaining are adaptive algorithms implemented in two flavors: minimal (NFM, WFM, NLM) and non minimal (NFNM, WFNM, NLNM). For XY routing, exactly a path exists between two communicating entities. For minimal adaptive algorithms, the number of distinct paths (***NPaths***) is either 1 or is defined by Equation (1), where Δx and Δy represent the distance in hops along the corresponding axis (x or y) between source and destination.

$$\boldsymbol{NPaths} = \frac{(\Delta x + \Delta y)!}{\Delta x! \Delta y!} \tag{1}$$

For non minimal adaptive algorithms, ***NPaths*** depends on the relative source and destination positions and on the routing algorithm rules. For example, for the non minimal negative first algorithm (NFNM), it is possible to use Equation (2).

$$\boldsymbol{NPaths_{nf}} = \sum_{x_r = x_s}^{0} \sum_{y_r = y_s}^{0} \frac{(\Delta x_{nf} + \Delta y_{nf})!}{\Delta x_{nf}! \Delta y_{nf}!} \tag{2}$$

When the destination of the packet is at a point (x_d,y_d), which is above and to the right of the source coordinates (x_s, y_s), Equation (2) is valid. In this Equation, the pair (x_r, y_r) represents the position resulting from the displacement from the source to the most negative position in the network. Also, Δx_{nf} and Δy_{nf} represent the distance in the x resp. y axes between position (x_r, y_r) and the destination, i.e. $\Delta x_{nf} = x_r - x_d$ and $\Delta y_{nf} = y_r - y_d$. For the other non minimal adaptive routing algorithms similar equations and considerations apply.

B. Combining Paths of Communicating Pairs

A route mapping is defined when for all pairs of communicating modules, one and only one path is chosen for each source and destination module (represented by the iteration variable $\boldsymbol{i}$). The amount of possible route mappings depends on the number of communicating pairs and the routing algorithm. The greater the number of alternative paths per communicating pair, the greater the number of achievable route mappings. Equation (3) defines the maximum number of route mappings (***NMapping***), where ***i*** identifies a communicating pair, ***NPairs*** is the total number of communicating pairs and ***NPaths(i)*** is the number of alternative paths for ***i***. As an example, ***NMapping*** is always equal to 1 when XY routing is adopted, since there is only a possible path for each communicating pair.

$$\boldsymbol{NMapping} = \prod_{\boldsymbol{i}=1}^{\boldsymbol{NPairs}} \boldsymbol{NPaths}(\boldsymbol{i}) \tag{3}$$

C. Evaluation of Route Mappings

The evaluation of route mappings is based on: (i) the *communication characteristic* that is modeled by all communications of the application in terms of source module, target module and transmission rate, (ii) the *alternative paths* for each communicating pair that is modeled by a graph containing all possible paths from each source to each destination module and (iii) the selected *cost function*, which considers: the average rate of path occupancy, the peak usage of the path and the path length. Smaller values for these aspects lead to better paths.

Initially, a valid path is randomly assigned to each communicating pair. In this step, no additional care is taken for path binding. The only guarantee offered by this assignment process is the existence of the path on the list of alternative paths for the given communicating pair. After a path has been assigned to each communicating pair, NoC occupation is estimated by accumulating the transmission rate of each communicating pair. The next steps seek to optimize this initial route mapping.

The route mapping optimization is carried by varying the paths of each communicating pair, trying all alternative paths. When a communicating pair is being evaluated, the remaining pairs have their path fixed.

A new path for a communicating pair is assumed if, compared to the current route mapping: (i) the average rate of path occupancy is lower and its peak usage is lower or equal, or (ii) the average rate of path occupancy is equal and the peak usage is lower, or (iii) the average rate of path occupancy is equal, the peak usage is equal and the path length is shorter. The first approach guarantees a better distribution of NoC communicating flows, equalizing the communication channel occupation. The second approach guarantees that if a low congestion zone cannot be found at least hotspots are avoided, bringing peak usage down. Finally, when using a non minimal routing algorithm, if the same average rate of path occupancy can be found in a shorter path, then lower power dissipation may be overlooked.

The process of route mapping optimization finishes in three possible situations: (i) when there is only one possible route mapping (e. g. when using XY routing), (ii) after reaching a given number of tries, which is parameterizable, and (iii) when no further optimization can be obtained after all communicating pairs have been evaluated at least once. The resulting route mapping is called ***planned source routing*** (PSR).

III. THE HERMES-SR ARCHITECTURE

Before describing Hermes-SR, it is necessary to approach two other NoC architectures used here as a basis for comparisons, Hermes and Hermes-M. Hermes and Hermes-M are 2D-mesh topology NoCs with routers using credit based flow control, input buffering, wormhole packet switching and centralized arbitration (round-robin algorithm). However, while the routing scheme of Hermes is distributed, Hermes-M uses a planned source routing scheme.

Hermes-SR differs from Hermes-M because it employs a *distributed arbitration* scheme with a first come-first served (FCFS) algorithm, which guarantees in-order packet servicing.

Depending on the PE mapping, on the specific (adaptive) routing algorithm and on some application characteristics, routing may overload several network regions, implying a decrease on the communication architecture efficiency, due to the increase of packet latencies.

The predefinition of paths supported by Hermes-SR source routing guarantees a known worst case for link loads. Also, it may help optimizing NoC area through the elimination of unused links and buffer dimensioning, both of which are outside the scope of this work.

All NoCs in this work assume a simple packet structure, composed by a header containing destination and size information and a payload. All three NoCs support arbitrary flit sizes, although all experiments here use only16-bit flits. Different NoC packets slightly differ in their header structure. In Hermes, the 2-flit packet header stores the router destination address as first flit, followed by a flit with the size of the payload. In Hermes-SR and Hermes-M, the header starts with an in order sequence of output ports necessary to arrive at the destination, followed by the single-flit payload size. While the Hermes header occupies exactly two flits, the other two NoCs' variable size headers comprise at least three flits, due to the use of an additional flit used as route terminator flag.

Concerning arbitration in Hermes-SR, each router input port directly notifies the desired output port to transmit a packet. Illustrated in Figure 1(b), this approach enables to serve multiple requests to distinct ports in parallel. Transmission requests are stored and served in arrival order by each output port. Figure 1(a) illustrates the other approach, used in Hermes and Hermes-M, which employs centralized round-robin arbitration. Here, each input port requests routing for a control unit and waits either for an output port assignment or for a denial of assignment, if the requested port is already busy. In any case, if arbitration serves a port it loses its priority. Because of this, input ports may suffer from starvation, depending on the network load and the amount of competition among communication flows.

IV. EXPERIMENTAL SETUP AND RESULTS

Hermes-SR, Hermes and Hermes-M were described in synthesizable RTL VHDL with fixed dimensioning (5x5), flit size (16 bits) and a 50MHz operating frequency, resulting in a bandwidth of 800Mbps per link. The experiments employed various routing algorithm and buffer size combinations.

Several synthetic and real traffic patterns allowed evaluating arbitration and routing schemes. While real traffic scenarios allow assessing the behavior of specific applications, synthetic traffic scenarios enable to explore the limits of the NoCs, such as saturation and behavior under congestion.

A set of text files describe each traffic pattern as a set of packets (header + payload) and the *ideal injection moment* for each packet. This is an integer informing the number of clock cycles after simulation start. Injection sources are implemented by cycle- and pin-accurate SystemC input modules, responsible for interpreting traffic files and injecting packets into the NoC. SystemC output modules absorb packets from the NoC outputs, storing their contents and arrival moment for statistical evaluation.

Latency values presented here are not limited to the NoC transmission delay. Figure 2 differentiates transmission latencies based on injection and reception distributions. A *planned injection* distribution is defined in the traffic scenarios text files, and depicts the ideal injection moment for each packet ***i***. The *accomplished injection* distribution considers the actual packet insertion moment into the NoC, which can be delayed by contention at the packet source. The *ideal reception* distribution represents the expected delivery moments of packets, taking network status into account or not. The *accomplished reception* distribution represents the real moment where packets are delivered to their destination. No contention at the destination is considered.

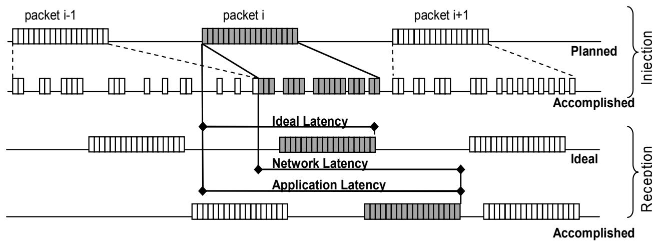

Figure 2 –Communication latency types.

A hypothetical distribution of such injection and reception scenarios is illustrated in Figure 2. *Ideal latency* is the minimum number of cycles a packet needs to reach its destination. This is based on the difference of the ideal injection moment and the expected delivery moment. *Network latency* is the delay verified by the packet during its traffic from source to destination, which may be influenced by competition for NoC resources (e.g. links, buffers, arbitration, routing). *Application latency* normally brings the most important impact on the ideal communication performance. This is computed as the difference between the ideal injection moment of packets and their effective delivery moment at the destination. Application latency is the value assumed for comparison in the next experiments.

A. Evaluating Performance under All to All Traffic Pattern

Two classes of experiments were performed to evaluate performance optimization. The first class compares the usage of PSR against distributed routing (DR) with the results summarized in Figure 3. The second class focus on centralized versus distributed arbitration, with results depicted in Figure 4.

The comparison values were captured from five distinct traffic scenarios, with injection rates 10%, 20%, 30%, 40% and 50% of the channel bandwidth capacity, corresponding to absolute rates of respectively 80Mbps, 160Mbps, 240Mbps, 320Mbps and 400Mbps. The temporal distribution of packet injection is uniform. Packets for Hermes have 20 flits, while for Hermes-SR and Hermes-M NoCs, the size varies around this value, depending on the amount of hops to reach the destination. The spatial distribution is one to all from each injection source, producing an all to all NoC traffic pattern.

The first experiment (Figure 3) assumed a traffic scenario where sources inject 30% of channel bandwidth capacity. It mainly compares Hermes running DR and Hermes-M using PSR with different routing algorithms. Both architectures employ a centralized arbitration scheme (round-robin).

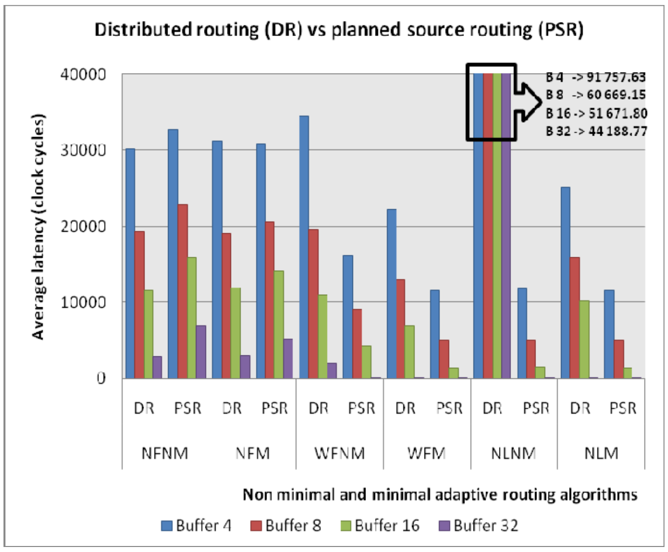

Figure 3 - Latency results obtained when comparing distributed routing (DR) versus planned source routing (PSR) approaches.

Figure 3 depicts that all PSR led to increased latency when compared to DR for NFNM and NFM, except for one case, which resulted in a small gain (1.12% - around 350 clock cycles faster in average). However, for the remaining routing algorithms latencies were reduced in all cases for PSR when compared to DR.

The behavior of WF and NL has three explanations. The first is the degree of freedom provided by WF and NL when compared to NF for route mapping exploration. This can be formally demonstrated, but intuitively speaking WF and NL determine a single direction that must be employed at the start (WF) or end (NL) of the routing process, while NF determines that only two directions (the negative ones) can be taken at the start of the routing process. The second explanation is the global knowledge of channels load when adopting planned routing. The third is the bad decision that DR may take, since judgments are made based on locally available information only to resolve congestion, possibly deviating packets to other congested regions.

During simulation, DR always achieved lower latencies when comparing WFNM to NLNM and WFM to NLM. However, when using PSR latencies are always lower when comparing NLNM to WFNM and NLM to WFM. These results show that the choice of routing algorithm strongly depends on the choice of routing strategy.

Figure 4 shows a second evaluation experiment that assumes traffic scenarios with packet injection rates varying from 10% to 50%. This evaluation essentially compares centralized (Hermes or Hermes-M) and distributed (Hermes-SR) arbitration schemes.

Routes were defined with the XY algorithm, guaranteeing the same packet distribution for all NoCs. No significant difference is observed up to 20% of injection rate. At the 30% of injection rate and above, it is noticeable that distributed arbitration can reduce the router control congestion, since latencies are significantly reduced when compared to a centralized approach. Additionally, it is observable that the bigger the buffer sizes, the lower the average latency, in all cases. However, it can also be observed that at each injection rate, the lowest average latency obtained when centralized arbitration employs the biggest buffer size (32-flit buffer), is greater than the average latency obtained with the same injection rate of distributed arbitration using the smallest buffer size (4-flit buffer), except at a 30% injection rate. This case is not really relevant because the difference is slight (around 10 clock cycles in average – less than 10% of difference).

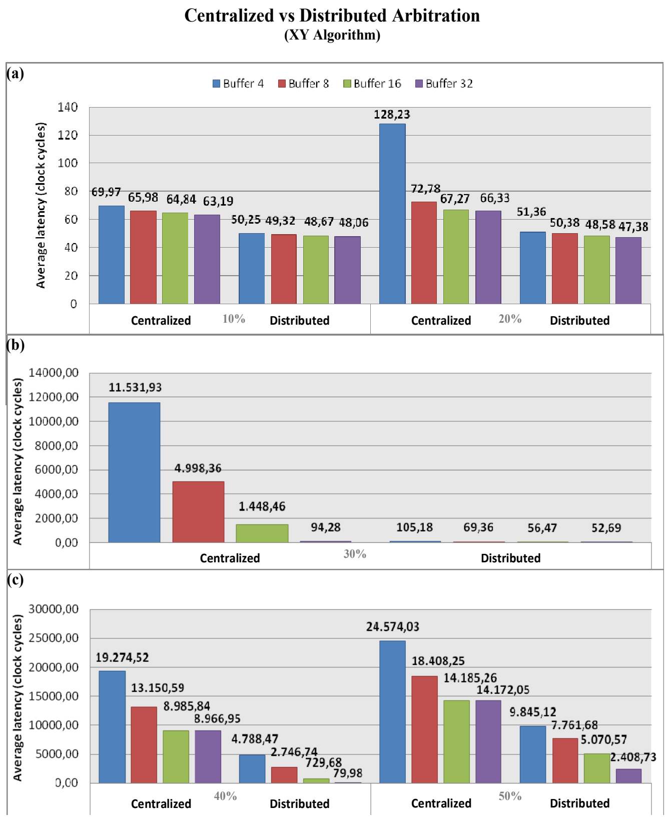

Figure 4 – Average latencies for centralized (Hermes and Hermes-M) and distributed arbitration (Hermes-SR). Injection rates vary from (a) 10%-20%, (b) 30% to (c) 40%-50%.

B. *Evaluating Performance under Hotspot Traffic Pattern*

This experiment allows evaluating the distributed or centralized arbitration architectural choices, and distributed or source routing decisions. Also, it enables measuring the effectiveness of statically planned routing. A hotspot traffic scenario is used, where two nodes concentrate all packet destinations on the NoC. A 100 Mbps injection rate with uniform temporal distribution per source is used. Table 1 presents the average latency computed during simulation. In the XY algorithm line of Table 1 only architectural decisions can be analyzed, since the same routes are used for all NoCs. Comparing Hermes and Hermes-M columns it can be seen that no gain is achieved when adopting distributed or planned source routing in the XY line. However, the comparison of Hermes-SR columns with the two previous ones shows that distributed arbitration led to more than 40% of latency reduction when compared to the centralized approach.

Table 1 –NoC average latencies comparison (in clock cycles).

	NoC	Hermes	Hermes-M	Hermes-SR
	Routing scheme	distributed	source (PSR)	
	Arbitration scheme	centralized		distributed
Routing algorithm	XY	15,047.06	15,047.06	8,746.96
	NF	3,364.56	1,894.57	582.29
	NFM	4,443.94	1,697.26	567.06
	WF	5,148.44	3,676.78	1,095.43
	WFM	11,081.46	5,344.77	2,382.24
	NL	2,730.85	2,050.10	754.18
	NLM	8,053.11	2,103.99	733.06

When comparing Hermes and Hermes-M columns of Table 1 – *the same arbitration and different routing schemes* – for all routing algorithms (except XY) the effectiveness of PSR stands out. The experiment showed latency reductions from 24.93% (NL) up to 73.87% (NLM) in these cases.

When comparing Hermes-M and Hermes-SR columns of Table 1 – *the same routing and different arbitration schemes* – the effectiveness of distributed arbitration is highlighted. The experiment showed an average latency reduction of 53.29%.

Additionally, it is noticeable the benefit of combining planned source routing and distributed arbitration, as supported by Hermes-SR. By comparing the first and last columns, the average latency is up to 11 times smaller (NLM) and the average latency reduction for all cases is 70.20%.

Figure 5 illustrates the link workload estimation when different routing algorithms are used as basis for route mappings in Hermes-SR. The XY picture presents two peaks of load concentration, resulting in high competition and consequent performance degradation. NF, NFM, NL and NLM are the most suitable algorithms to distribute the workload into the NoC links, followed by WF and WFM.

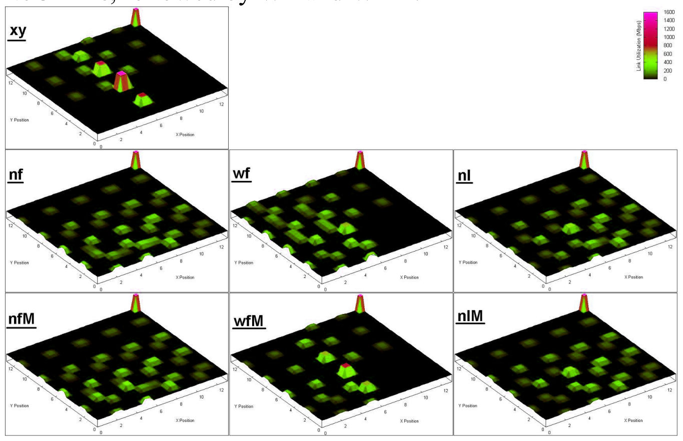

Figure 5 - Links workload estimation when employing distinct routing algorithms for path selection in a planned source routing applied to a 5x5 Hermes-SR NoC. The peak at the top of each square represents the maximum workload reached during simulation.

C. *Evaluating Area Costs*

Table 2 depicts the area usage of Hermes and Hermes-SR central routers (with 5 input and output ports) in a XC5VLX30 Virtex 5 FPGA. Area is expressed in terms of number of LUTs and flip-flops for four configurations of buffer size. Synthesis results come from the use of the XST tool, part of the Xilinx ISE 9.2i toolset. Nothing but default parameters were assumed in the synthesis tool. Hermes-M is not explicit referenced in Table 2, since its implementation is quite similar to Hermes. The difference implied by the routing mechanism implementation has no significant impact in terms of area.

Table 2 – Hermes and Hermes-SR area comparison (5-port router).

Buffer size	Hermes		Hermes-SR	
	LUTs	Flip-flops	LUTs	Flip-flops
4	1064	212	1437	240
8	1128	235	1505	260
16	1227	254	1634	280
32	1532	266	1915	300

* Results for a 5-port router

Although Hermes-SR improves performance it clearly penalizes area, since it presents an average of 31.7% more LUTs and 11.7% more flip-flops than Hermes. However, Figure 6 (extracted from Figure 4) points out that distributed arbitration NoCs, even implemented with 4-flit buffers, achieves better performance than centralized arbitration NoCs implemented with 32-flit buffer for all injection rates, except 30%. Comparing a 4-flit buffer Hermes-SR NoC with a 32-flit buffer Hermes, the average latency for all injection rates reduces by approximately 35.2%.

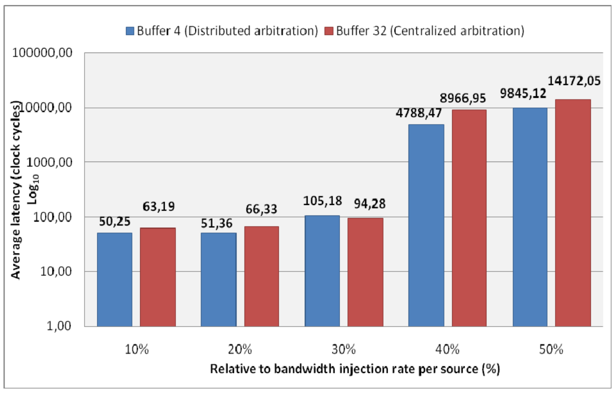

Figure 6 – Average latencies when comparing distributed arbitration in 4-flit buffer Hermes-SR to centralized 32-flit buffer Hermes for injection rates varying from 10% to 50%.

When considering an implementation of a 4-flit buffer Hermes-SR and an implementation of a 32-flit buffer Hermes, it is visible that choosing Hermes-SR implies area consumption reduction (6.2% less LUTs and 9.8% less flip-flops). Since the average latency is also reduced in Hermes-SR implementation, even with such a small buffer size, it is easy to conclude that centralized arbitration with planned source routing is a good design choice, reducing size and latency. Moreover, some works show that buffer size strongly contribute to power dissipation [10], while others show that NoC buffers may account for around 90% of the power dissipation in a router [11]. Therefore, the reduction of 32-flit buffer to 4-flit buffer probably also implies significant energy savings.

V. Conclusions and Ongoing Work

In this paper, the main contributions come from the performance evaluation of routing and arbitration architectural decisions and their impact on area costs. A secondary contribution is the proposition of a method for path computation, based on previously known application traffic behavior, which guarantees absence of deadlock and more balanced communication load.

The Hermes-SR NoC architecture was implemented to explore the use of distributed arbitration and source routing. This NoC served to compare the trade-offs involved in deciding about the use of source versus distributed routing strategies, as well as in the use of centralized versus distributed arbitration strategies. Additionally, the paper proposes a route mapping process that can be advantageous for enabling the estimation of link occupancy in NoCs and the consequent performance improvement. This allows solving congestion mitigation through hotspots avoidance. The results point out to a NoC design with performance improvement, power dissipation reduction and area saving.

Ongoing work centers on dynamic traffic scenarios, where applications are loaded at runtime and communication requirements are requested on the fly. Self adaptive NoCs based on global information knowledge are an interesting choice, since results definitely showed that decisions based on locally acquired information may lead to bad performance results.

Acknowledgements

The Authors acknowledge the support of the CNPq through research grants 141247/2005-3, 308924/2008-8, 309255/2008-2, 301599/2009-2 and of the FAPERGS grant 10/0814-9.

References

[1] Wolf, W. et al. “Multiprocessor System-on-Chip (MPSoC) Technology”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(10), Oct. 2008, pp. 1701-1713.

[2] Pasricha, S.; Dutt, N. “On-Chip Communication Architectures – System on Chip Interconnect”. Morgan Kaufmann Science, 2008, 544p.

[3] Moraes, F. et al. “Hermes: an infrastructure for low area overhead packet-switching networks on chip”. Integration VLSI Journal, 38(1), Oct. 2004, pp. 69-93.

[4] Glass, C.; Ni, L. “The Turn Model for Adaptive Routing”. Journal of the Association for Computing Machinery, 41(5), Sep. 1994, pp. 874-902.

[5] Hu, J.; Marculescu, R. “DyAD - Smart Routing for Networks-on-Chip”. In: DAC’04, 2004, pp. 260-263.

[6] Kulmala, A. et al. “Distributed bus arbitration algorithm comparison on FPGA-based MPEG-4 multiprocessor system on chip”. IET Computers & Digital Techniques, Jul. 2008, 2(4), pp. 314-325.

[7] Bertozzi, D.; Benini, L. “Xpipes: A Network-on-chip Architecture for Gigascale Systems-on-Chip”. IEEE Circuits and Systems Magazine, 4(2), 2004, pp. 18-31.

[8] Fen, G.; Ning, W. “A Minimum-Path Mapping Algorithm for 2D mesh Network on Chip Architecture”. In: APCCAS’08, 2008, pp. 1542-1545.

[9] Bolotin, E. et al. “Routing Table Minimization for Irregular Mesh NoC”. In: DATE’07, 2007, pp. 942-947.

[10] Banerjee, N. et al. “A Power and Performance Model for Network-on-Chip Architectures”. In: DATE’04, 2004, pp. 1250-1255.

[11] Palma, J. et al. “Mapping Embedded Systems onto NoCs – The Traffic Effect on Dynamic Energy Estimation”. In: SBCCI’05, 2005, pp. 196-201.