2011 14th Euromicro Conference on Digital System Design

Dynamic Flow Reconfiguration Strategy to Avoid Communication Hot-Spots

Romain Prolonge, Fabien Clermidy
CEA-LETI-MINATEC
Grenoble, FRANCE
{romain.prolonge, fabien.clermidy}@cea.fr

Leonel Tedesco, Fernando Moraes
FACIN-PUCRS
Porto Alegre, BRAZIL
{leonel.tedesco, fernando.moraes}@pucrs.br

***Abstract*— Application-specific Network-on-Chip allows optimization for the interconnection to minimize its cost. When used with streaming applications, large flows of data can be predicted. However, these flows can be modified during the applications providing a dynamic flow graph. In that case, applying on off-line optimization leads to an over-sizing of the NoC. On the other hand, dynamic reconfiguration leads to unordered data deliveries with costly re-ordering units. In this paper, we propose a coarse grain dynamic reconfiguration which avoids data re-ordering requirement. We show that the proposed solution is efficient to deal with communication hot-spots, with a small area overhead, and can save up to 33% of latency.**

Keywords: dynamic routing, hot spots, re-ordering

I. INTRODUCTION

Multimedia and telecom applications are some examples of systems requiring the most high performance computing part. For the first one, starting from 3D video to go to augmented reality leads to exponentially growing need for performance. Similarly, 3GPP-LTE applications and their successors are multiplying the complexity in two ways: increased bandwidth with multiple hundred of Mbits/s targeted; improved quality of service by using high-level coding schemes in Multiple Input Multiple Output (MIMO) antennas such as directional beam forming.

Both multimedia and telecom applications will be mapped on multi-core architectures consisting on many tens to hundreds of cores in the most advanced systems. In such a context, all these applications are sharing a common issue: the growing bandwidth demand on communication between cores. As an example, let's take a high resolution video (1080*1240 pixels image size) at 60 frames per second rate providing a raw 2.6Gbits/s video flow. If we consider 20 to 50 processing steps, it leads to 52 up to 130 Gbits/s internal bandwidth requirements. The problem to solve is twofold: providing the necessary bandwidth and associated quality of service while limiting the interconnect area and power consumption.

Network-on-Chip (NoC) architectures are an evident part to solve this problem as bus-based System-on-Chip (SoC) is far from providing the needed parallelism level. In a general purpose system with no special information on the traffic, these high bandwidth requirements will lead to over-designing and thus will not meet the low area/power consumption objective. Fortunately, the targeted applications are streamed-based: some important flows of data will be transferred from a core to another in a well-known sequencing.

As a result, some previous works [1],[2] have proposed application specific NoC architectures associated to dedicated routing algorithms. However, the variety of applications (large number of modes in telecommunication, many algorithms in video and image processing) associated to technology masks costs are leading to more flexible, more generic, more programmable SoC. Similarly, applications are moving to be more data-dependent and despite the fact that flows of data are still predictable, their sizes and their occurrence in time become difficult to compute off-line. This problem can lead to over-sizing the NoC, as a same connection used by different paths in different time slots will be designed in a worst case configuration. In most of the configurations, this worst case generating a communication hot-spot will never occur but must be taken into consideration for safety reasons.

One way to solve this issue is to propose a dynamic flow management. Such strategies are efficient but lead to two issues which must be solved: assuring that the new routing is deadlock-free; re-ordering the data at the arrival. The first problem has been solved in some cases (especially X-Y routing) by limiting the flexibility of dynamicity [3]. The second issue is solved by limiting packets sizes, putting tags on the packets and providing large registers at the destination for re-ordering the packets considering their tags. The complexity of such a mechanism is important, as well as the area overhead.

In this paper we propose a re-ordering free dynamic flow reconfiguration to avoid communication hot-spots for source-routing NoC. Section II shows some previous works related to dynamic reconfiguration. Section III presents the concepts of our proposal as well as the proof of the re-ordering free property. Section IV shows the implementation results and the area overhead of the proposed method while section V proves the efficiency of the proposed method.

II. RELATED WORKS

Routing algorithms have been widely studied in the past for different topologies and different purposes, such as average

978-0-7695-4494-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DSD.2011.72

congestion diminution, power reduction or fault-tolerance. In this section, we focus on dynamic routing works.

In [3], M. Li et al. present a deadlock-free dynamic routing algorithm named DyXY. With this algorithm, each packet follows the shortest path between the source and the target. As it is influenced by the XY routing algorithm, each router arbitrates routing decisions between two output ports. If the current router and the destination are aligned on X or Y axis then the packet is straightly routed to the destination. In that case, the congestion information is not taken into account. On the contrary, if the current router and the destination are not aligned, the congestion status of the neighboring routers toward the destination is checked. Then, the packet is sent to the less jammed router. Buffer occupation statistic is used to spot these congestions. This solution gives good results in an XY manner routing but does not consider the re-ordering issue in the target router and do not consider source routing.

J. Hu et al. propose, in [4], the DyAD routing scheme which uses the advantages of both deterministic and adaptive routing strategies. As in [3], routing decisions are taken in each router considering the close-by congestion environment. But contrary to this previous method, if no congestion is detected a deterministic routing strategy is used. As soon as a congestion is found, the routing switches to an adaptive strategy. As explained in the paper, the DyAD concept only is presented and it does not consider the re-ordering issue which can occur.

P. Lotfi-Kamran et al. elaborate in [5] the BARP protocol which operates on two mechanisms to avoid congestion. In the first one, it tries to use all the network resources. Each router uniformly distributes the incoming flow to all its output ports located on the shortest path to the target. With this distribution the probability of presences of congestions is decreased but not suppressed. The second mechanism aims at spotting the near congestion routers and modifying the associated routing path. The congestion status of each router is monitored accordingly to its input buffer occupation. When a router is in a near congestion state, it asks the upstream routers to change the routing path. The difference with [3] and [4] is that the congestion detection and the routing path modification are not done in the same router.

In [6], P. Gratz et al. present the Regional Congestion Awareness (RCA) which exploits local and non-local congestion information in order to provide a global picture of congestions in the network. RCA uses a monitoring network to transmit congestion information. The output port choice is made on a 50-50 weight assignment between local congestion status and non-local congestion information. As the proposed work relies on a minimal routing, if too much weight is put on the non-local information, the routing strategy is influenced by remote parts of the network which may not be reached. When compared with local congestion awareness, this approach is reducing latency and improving throughput.

L. Tedesco et al. propose in [7] an adaptive source routing mechanism based on a monitoring of congestions on the routing path. The monitoring phase is done by the header of each data packet. Once injected into the network, headers collect the congestion status of a specific router on the path. Congestion information is then gathered in the target node which performs a congestion detection based on a threshold determining the QoS requirement. Once congestion is detected, the source node receives a special packet containing the location of the congestion. As this study relies on a source routing mechanism, the source node computes a new path, congestion and deadlock-free based on turn-even model. However, this mechanism requires a re-ordering strategy in order to reconstruct the original message in the target router. Moreover, the cost in terms of hardware is not identified, leading to difficult estimation of the real benefit.

In this paper, we tackle the re-ordering issue due to the adaptive routing. As this re-ordering requires storing a potential high number of packets before delivering them to the target, it is consuming a large area not compatible with embedded systems constraints. Contrary to previous works, we propose a solution where, by construction, the re-ordering is not needed. In other words, we propose a method which performs an adaptive source routing mechanism and guarantees the right order in the data deliveries without requiring any re-ordering unit in the target node. We also propose an implementation of this mechanism showing its small overhead.

III. DYNAMIC FLOW RECONFIGURATION CONCEPTS

In the proposed model, we leverage on two concepts: a flow control by the mean of a credit/data exchange mechanism between the source of data and the corresponding target; the source routing which gives the possibility to change the path at the source node and avoid complex mechanisms in the routers which can decrease their frequency. The routers themselves are slightly modified for path monitoring purposes but do not tackle with the congestion detection.

A. Credit/Data exchange mechanism

The exchange of credits and data between the source and the destination has been introduced in a previous work presented in [8]. A credit flow is emitted by the receiver and sent towards the sender. Its purpose is to allow the source to send a given number of data flits. As a consequence, the source node cannot send any data into the network unless it is sure that the receiving node can accept it in its input buffer. In this way, no data can be blocked inside the NoC. Therefore, the communication is controlled by the receiving node through this credit flow. “Credit” packets convey the availability of the target node’s buffer to receive data. Each “credit” packet contains N credits. So, when the receiving node sends one “credit” packet, the source will be able to send N data. Moreover, this “credits/data” mechanism can be used for every communication, whatever its length.

Definition 1: a message is a communication of a length M flits between a dedicated source and its related target resource corresponding to a functional data exchange.

Definition 2: a “credit” packet is a single flit packet. So it has only a slim impact on the other existing flows. Each packet carries N credits. If $\frac{M}{N}$ is not an integer, the last “credit” packet to be sent carries $\frac{M}{N} - \left\lfloor \frac{M}{N} \right\rfloor$ credits in order to complete the communication.

An example of the credit transmission is shown in Figure 1. Here, the length of the communication is 13 packets and a "credit" packet allows the sender to transmit 4 packets. So 3 "credit" packets will be sent and an extra "credit" packet will allow the source node to inject one quarter of the usual credit amount, i.e. one last packet into the network.

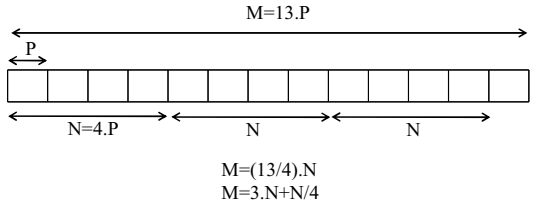

Figure 1 Credit transmission

B. *Dynamic reconfiguration strategy*

Due to the dynamic behavior of the communication in the NoC, hot spots can spontaneously appear decreasing the overall performance of the network. While the major part of the studies regarding the use of adaptive routing mechanisms limited the analysis of congestion on the close-by environment, our solution proposes similar congestion monitoring and source-based reconfiguration mechanisms than the work in [7]. It achieves a monitoring of the traffic on the entire routing path, detects possible congestions and modifies the path followed by the data flow in order to be congestion as well as deadlock-free.

As said previously, the monitoring phase is performed by the header of each data packet. No additional flow is required for this phase; therefore, the NoC utilization remains the same. The first data packet allocates a unique identification tag to each node on the routing path. Then, each packet sent on the same path collects the congestion status of a particular router identified by its tag. As a result, the target node gathers the congestion information of all the routers on the routing path after receiving N+1 packets. N corresponds to the number of hops between the source and target nodes. Each time the entire path is monitored, the target resource checks the presence of congestion. If detected, a new path is computed. However, [7] do not tackle the re-ordering issue which appears when adaptive routing is performed.

C. *Re-ordering free methodology*

Based on the credit/data system, the next part will show how to combine it with an adaptive source routing strategy in order to have a system that can dynamically avoid congestions and guarantee the order of data deliveries. The following process describes the mechanism and is illustrated in Figure 2 (the thin arrows symbolize the "credit/alarm" path while the wide arrows represent the data path):

1) The first step of every communication is to configure the source and the target. Configuration and communication protocols are described in [8]. Then, the first packet sent prepares the path for the reconfiguration by giving each router a unique identifier in the path.

2) The target transmits its first "credit" packet, which contains N credits, to the source node as shown in Figure 2a. The data flow resulting from these credits performs the monitoring of the path in order to spot congestions.

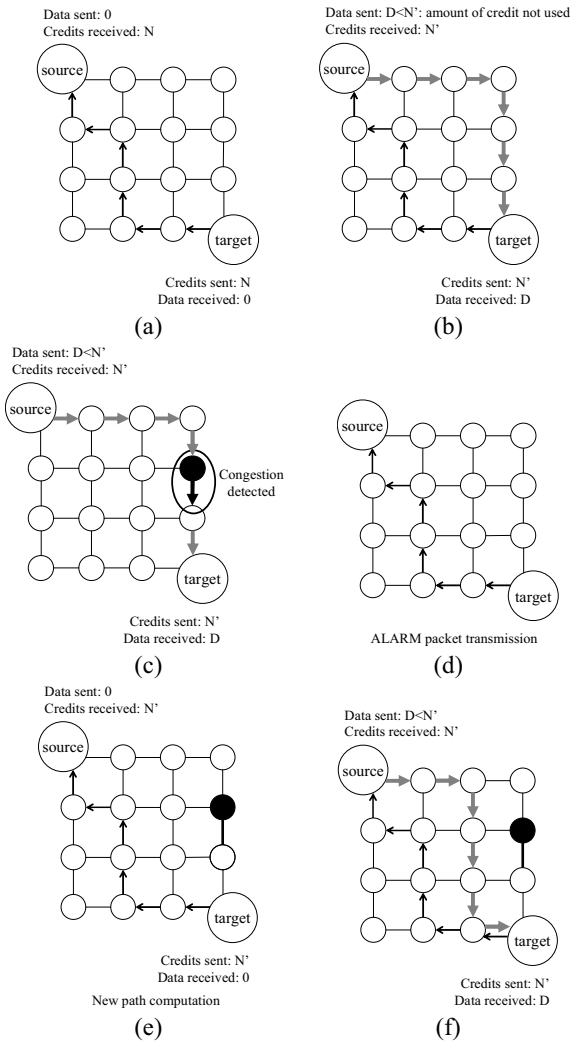

Figure 2 Non-blocking, re-ordering-free, adaptive source routing example

3.a) If no congestions are detected, the source node uses its credits and the target node sends credits accordingly to its buffer's availability (Figure 2b). The "credit/data" cycle continues.

3.b) A congestion due for example to a local hot spot can be detected at the target resource. The detection is made by a threshold applied to the information of average time spent in a router. The "credit/data" cycle breaks. The target node stops sending any credits (Figure 2c).

4) At this point, the sender has no idea that congestion has been detected and keeps sending data through the current path and the target node keeps receiving the data, waiting for the source to stop the transmission when no credit will be available. *The fact that the target resource waits until the amount of credits it sent equals the amount of data it received guarantees the right order in the data arrival*.

5) Once all data have been sent and received, the target node transmits a single flit: the "Alarm" packet described in [7]. This step is illustrated in Figure 2d. Its purpose is to inform the source node of the presence and location of congestions.

Once the "Alarm" packet has been transmitted, the target node starts sending credits again.

6) Once the sender receives the Alarm packet, it stops sending any data packet even if it receives credits. Then, it computes a new path avoiding the detected congestions, as shown in Figure 2e. This path will be taken into account for the next packet emission. When the computation is done, the "credit/data" cycle resumes.

7) As a result, the data are injected into the network and reach the target node through the congestion-free path. This final step is shown in Figure 2f.

This sequencing guarantees the ordering of packets at the destination resource. However, there is a cost to pay, both in terms of hardware overhead and latency overhead. Indeed, when congestion is detected, the target node stops the credit flow up to completion of the corresponding data flow, which leads to a certain latency overhead. In order to answer these overhead issues, an implementation has been made and the corresponding hardware overhead is discussed in the next section; correspondingly, section V investigates the gain of the system depending on the message size

IV. IMPLEMENTATION

A. Architecture presentation

For testing the proposed scheme, we added the dynamic reconfiguration scheme to the heterogeneous architecture dedicated to streaming applications presented in [8]. This architecture uses a 5x5 mesh-based topology. It deals with a minimal source routing and a wormhole packet policy is used. The whole architecture is Globally Asynchronous Locally Synchronous (GALS), i.e. each resource has its own clock domain and GALS interfaces are connecting the unit to the NoC, which is implemented in an asynchronous manner.

The Network Interfaces (NI) connects the resources to the NoC and performs the flow control including the credit mechanisms previously presented. As shown in Figure 3, this block also performs the configuration of the communication as well as the core attached to it. A streaming unit is used for handling the data flow sequencing, and thus giving a homogeneous programming model for the heterogeneous units. Finally, the NI also tackles the interruptions from the core and communication units through an interruption manager.

B. Monitoring step

The monitoring phase does not use any additional flow but the encoding of data packet has been slightly modified compared to [8]. The new packet encoding is illustrated in Figure 4. The "router id" field holds the identifier of the router this header has to monitor. When a data packet's header enters a router, this field is compared with the router's tag. If both of them match, the congestion information is stored in the "congestion" field of the header flit and transmitted to the target node. The congestion is evaluated through the flit's average crossing time (ACT) in the current router. The "congestion" field refers to a range of ACT.

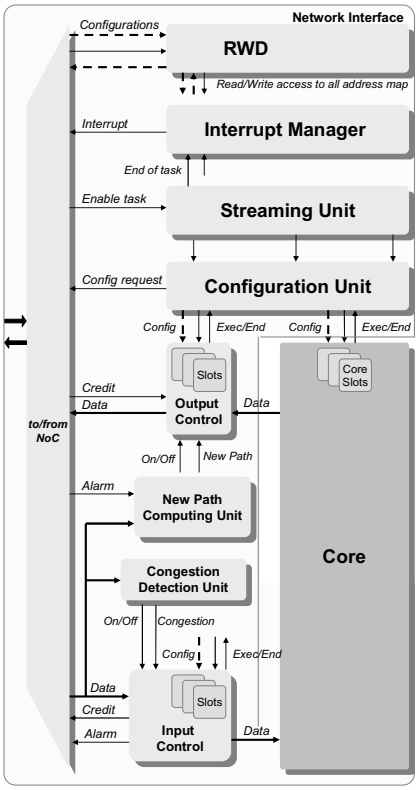

Figure 3 Network Interface with adaptive routing mechanism

Moreover, as mentioned in [8], initial and final packets of a message are tagged with bit 31, named Begin Of Message (BOM), and bit 30, named End Of Message (EOM). However, a new feature has been introduced for the adaptive routing implementation. When BOM=1 and EOM=1, the current packet monitors the last router of the routing path. This event will trigger a congestion analysis in the target node.

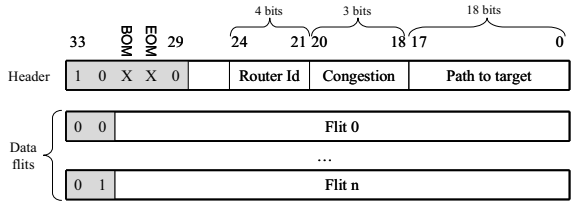

Figure 4 Data packet encoding

The "Alarm" packet is a new packet, designed for the reconfiguration issue. Its encoding structure is detailed in Figure 5. The "congestion" field holds a 7-bit vector. Each bit represents a router on the path and its congestion status. At the source node, this information will be used for the dynamic re-routing.

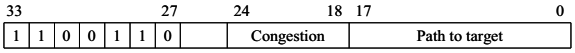

Figure 5 Alarm packet encoding

The "Alarm" flow is made of a single-flit packet. Moreover, this flow is emitted only when a congestion is

detected. Therefore, it has an even slimmer impact on the NoC than the credit flow.

C. RTL design

As stated in the previous section, an evaluation of the hardware overhead of the proposed scheme is important to judge its efficiency. To do so, we wrote an RTL (Register Transfer Level) description of the proposed scheme included in the NI proposed in [8]. At the end, two units were added. This NI architecture is illustrated in Figure 3.

The first new unit is the Congestion Detection Unit (CDU). It stores each congestion status of all routers on the routing path. When the congestion analysis is triggered, the CDU identifies jammed routers accordingly to a given threshold. In our implementation, the ACT in a congestion free context is 2ns and the threshold to detect a congestion is for ACT greater than 3ns. If at least one congestion is detected, this information is sent to the Input Control Unit which stops "credit" packets emission. When all data have been received, it creates and emits the appropriate "Alarm" packet and resumes "credit" packets emission. The second unit is the New Path Computing Unit (NPCU) which receives the alarm packet from the network and then performs the adaptive routing mechanism. Starting from an event which is the occurrence of an alarm, the unit stops data emission and extracts the congested router identifier before computing the congestion free path. When it is done, NPCU resumes data emission with the new path.

D. Implementation results

The NI was generated and synthesized with the logic synthesis tool Design Topologic Compiler from Synopsys in a CMOS Low-Power 65nm technology. The power consumption is computed on the back annotated netlist using the Synopsys Prime Power tool. The results regarding the overhead caused by the CDU and NPCU are presented in TABLE I. The results show a total overhead of approximately 11% in terms of area and 7% in terms of power consumption without impacting the performance of the initial NI (400 MHz). This small overhead is clearly compatible with such heterogeneous multi-core.

TABLE I. IMPLEMENTATION RESULTS

	CDU	NPCU
Area (65nm) [% of network interface]	1,827µm² [1.8%]	9,504µm² [9.2%]
Max. frequency	400MHz	400MHz
Power @fmax [% of network interface]	0.11mW [1.8%]	0.32mW [5.2%]

E. Comparison with re-ordering

Now, let's consider the equivalent area of the proposed method traduced in terms of registers. A Flip-Flop (FF) "DFPQ" with a driving factor of 9 has an area of 9.88 µm² in the technology used in this paper (LP 65nm). Considering all the area occupied by registers, this means 1147 FF. It results in a total of 36 32-bit words or 18 64-bit words. It results that only two to four 8-word packets can be re-ordered with the same area than the proposed method one, without taking into account logic overhead. This is clearly not enough when considering even a small-size NoC.

V. LATENCY IMPROVEMENT

Based on the previously presented architecture and its corresponding implementation, we have performed some experiments to see the impact, positive or negative, of the rerouting algorithm on the messages latency. Indeed, as stated in section III, the re-routing phase without re-ordering implies a preliminary latency overhead which must be compensated by the gain obtained thanks to hot spots avoidance. In order to evaluate the efficiency of the method, the question to answer is to know the minimum size of the messages from which the rerouting scheme has a gain compared to the hot spot duration. In the following of this section, a hot spot scenario is described and used for this evaluation.

A. Evaluation Scenario

Figure 6 displays the scenario used to evaluate the latency reduction caused by the adaptive routing strategy. The main communication occurs between the nodes labeled S and T. The hotspot communication occurs between H1 and H2. The platform used allows a SystemC-TLM/HDL co-simulation mode. In order to simulate the implementation of the adaptive routing mechanism, the VHDL version is used for the resources S and T and the rest of the network uses the SystemC TLM implementation.

The length of the message sent between S and T varies between 32 and 8192 flits. Each data packet contains 8 flits and a credit packet contains 8 credits. To avoid slack in the credit/data mechanism, the FIFOs on each side are fixed to 16 places. Similarly, the hot-spot is generated with 8-flit data packets. In order to see the effect of the hot-spot size, the total number of flits transmitted in a hot-spot message may vary from 32 to 8192. For the simulation, hot-spots and main communications start simultaneously in order to avoid long simulation time. However, it does not loose in generality, as it represents the moment the hot spot appears in a real application communication.

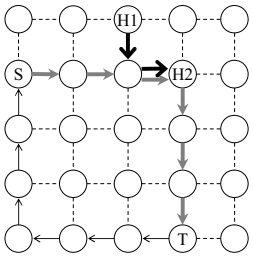

Figure 6 Scenario used for latency evaluation

The first packet performs the session establishment between the source and the target and the following packets realize the network monitoring. Consequently, a 16-flit message, divided into 2 8-flit packets can only monitor the first router of the path and will be of poor quality. On the opposite, a 1024-flit message, divided into 128 8-flit packets can monitor a path through 127 routers. In our case, the 7-router path will be monitor several times during long messages transmission. The results obtained through simulation are shown in Figure 7. As shown in Figure 6, a congestion is expected to be detected on the third of the six routers present on the routing path.

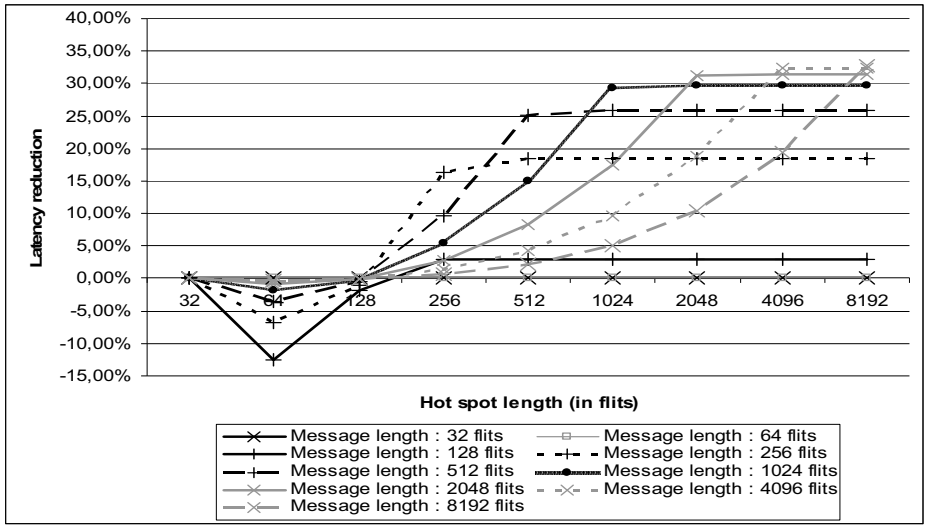

Figure 7 Latency reduction according to the message length and the hot spot communication length

For a 32-flit message, no latency reduction is observed. Indeed, this message is split into 4 8-flit packets. The first packet prepares the path for the monitoring phase and the three following ones monitor the three first routers. The congestion is detected by the last packet of the communication. Therefore, this detection occurs too late. For a 64-flit message, split into 8 8-flit packets, no latency reduction is observed. This message can monitor 7 routers and this path needs to monitor 6 routers. So, once the all path has been monitored, only one packet remains in the source node. However, it still has a few credits left. As a result, the last packet is sent while the target node waits for all the credits sent to be used.

A 32-flit hot spot is too short to be detected. Indeed, the time from the emission of the first packet and the emission of the fourth one, which will monitor the congested router, is greater than the hot spot duration. Therefore, no congestion can be detected and no latency reduction is observed. For a 64-flit hot spot, the reconfiguration system extends the overall latency. Indeed, the congestion is detected, the new path computation is triggered but the hot spot communication ends during this computation. So at the end, the latency is extended by the path computation and the "credit"/data flows stopping. In this scenario, 46 clock cycles are added to the latency. For a 128-flit hot spot, the reconfiguration system slightly extends the overall latency. In this case, the hot spot communication ends a small time after the new path has been used. The gain produces by the reconfiguration strategy cannot make up for its latency.

For a message length greater or equal than 128 flits and for a hot spot length greater or equal than 256 flits, latency reduction is observed. For a given message length, this reduction increases until the hot spot length becomes greater than the message one. After this point, the latency reduction levels off. This value represents the greatest latency reduction that can obtain a message of this length. Moreover, this value increases with the message length and tends toward 33%.

VI. CONCLUSION

In this paper, we have presented a coarse grain dynamic routing reconfiguration for streaming applications. The proposed scheme applied to a source-routing is based on a credit mechanism associated to a monitoring process. It avoids costly data re-ordering and shows an overhead of less than 4 packets re-ordering. Furthermore, we show that the proposed solution is efficient to deal with communication hot-spots when messages and hot-spots have sufficient length. As a result, up to 33% of latency can be saved.

REFERENCES

[1] M. Palesi, R. Holsmark, S. Kumar, et V. Catania, "Application specific routing algorithms for networks on chip," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, 2009, p. 316-330.

[2] K. Srinivasan, K. Chatha, et G. Konjevod, "Application specific network-on-chip design with guaranteed quality approximation algorithms," *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, 2007, p. 184-190.

[3] Ming Li, Qing-An Zeng, et Wen-Ben Jone, "DyXY - A proximity congestion-aware deadlock-free dynamic routing method for network on chip," *Design Automation Conference, 2006 43rd ACM/IEEE*, 2006, p. 849-852.

[4] Jingcao Hu et R. Marculescu, "DyAD - Smart routing for networks-on-chip," *Design Automation Conference, 2004. Proceedings. 41st*, 2004, p. 260-263.

[5] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, et Z. Navabi, "BARP - A dynamic routing protocol for balanced distribution of traffic in NoCs," *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, p. 1408-1413.

[6] P. Gratz, B. Grot, et S. Keckler, "Regional congestion awareness for load balance in networks-on-chip," *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, 2008, p. 203-214.

[7] L. Tedesco, F. Clermidy, et F. Moraes, "A monitoring and adaptive routing mechanism for QoS traffic on mesh NoC architectures," *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, 2009, p. 109–118.

[8] F. Clermidy, R. Lemaire, Y. Thonnart, et P. Vivet, "A communication and configuration controller for NoC based reconfigurable data flow architecture," *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, La Jolla, CA, USA: 2009, p. 153-162.