King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

The efficiency of buffer and buffer-less data-flow control schemes for congestion avoidance in Networks on Chip

Ahmed Aldammas*, Adel Soudani, Abdullah Al-Dhelaan

College of Computer and Information Sciences, King Saud University, Saudi Arabia

Received 27 May 2015; revised 24 November 2015; accepted 25 November 2015
Available online 15 December 2015

KEYWORDS

Network-on-Chip (NoC);
Congestion control;
Data flow control;
Flit's dropping;
Quality-of-service (QoS) insurance

Abstract The design of efficient architectures for communication in on chip multiprocessors system involves many challenges regarding the internal router functions used in Network on Chip (NoC) infrastructure. The on-chip router should be designed to provide per-flit processing with enhanced granularity. In fact, the quality of service experienced at the application level depends on the capabilities of the router to avoid congestion and to ensure efficient data-flow control. Consequently, an enhanced router architecture is needed to achieve the requested QoS.

This paper proposes an internal router architecture, for on chip communication, implementing flow-control mechanism for congestion avoidance with QoS consideration. It describes the internal functions of this router for optimal output flit scheduling and its capability to apply per-class service for inbound flows. The paper focuses mainly on the description and performance analysis of two proposed schemes for data flow control that can be used with the proposed router architecture. The results shown in this paper prove that the application of these proposed schemes in NoC achieves an interesting enhancement in the measured end to end QoS. We carried out an extensive comparison of the proposed solutions with the existing schemes published in the literature to show that the proposed solution outperforms these, maintaining an interesting tradeoff with the hardware characteristics when designed with 45 nm integration technology.

1. Introduction

The communication aspect of Multi-Processor Systems-on-Chip (MP-SoC) is one of the biggest challenges in the new generation of embedded systems. A lot of active research is directed toward designing an efficient communication architecture for these systems. In depth, Networks-on-Chip (NoC) used to interconnect the multi-cores, are characterized by high-bandwidth links between the routers due to the high frequency

* Corresponding author.
E-mail addresses: bindammas@student.ksu.edu.sa (A. Aldammas), asoudani@ksu.edu.sa (A. Soudani), dhelaan@ksu.edu.sa (A. Al-Dhelaan).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.11.002
1319-1578 © 2015 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

of the chip. However, the area cost of the memory part of the integrated circuit strongly limits the memory capacity of the router, which limits its ability to absorb heavy traffic bursts. The lack of memory in these routers confronts the networks to the problem of congestion. It induces substantial flit losses and overhead delays that heavily affect the perceived quality of service at the application.

The efficient design of these systems requires to take into account and combine different constraints. In particular, embedded hardware constraints have to be considered simultaneously with networking constraints in order to evolve the NoC to meet the requirements of end-to-end QoS (e.g. end to end delay, jitter variation, number of lost flits, throughput at the reception level, etc). In this regard, data flow and congestion control mechanisms designed for computer networks-oriented routers are not applicable in our context.

Recently, very-large-scale integration (VLSI) technology has offered a flexible and powerful environment for the designers of digital circuits (Chen et al., 2009). Indeed, it allows maintaining a tradeoff between the complexity of used algorithms and the cost of the designed circuits. Additional tasks to manage QoS in NoC routers are, therefore, possible with relatively acceptable hardware overhead. With this consideration, the main challenge lies then, in the specification and implementation of an efficient scheme for congestion control applicable to on-chip communication.

High flit arrival rate in NoC routers requires an enhanced scheme for internal queuing of the incoming flits that helps to process and forward them according to their QoS constraints taking into account at the same time the router instant load. To achieve this goal, a per-flit management process should be implemented according to the importance of the payload data type of the transported flit.

The main contribution of this paper is, then, to propose a new architecture of NoC router with integrated mechanisms for congestion control. It presents two schemes for congestion control with respect to the end-to-end QoS requirements. The first proposed scheme is based on output buffering of flits before they are injected into the next hop that is experiencing congestion. The second proposed scheme, called feedback signaling mechanism, is a buffer-less approach based on the interaction between the source and router in congestion to notify the source to reduce its sending window size. The paper evaluates the efficiency of the proposed congestion control solutions for different traffic loads by comparing the performances of the two proposed schemes regarding QoS.

The remainder part of this paper is organized in the following sections. We first survey the main existing contributions in the area of congestion and data flow control in network on chip. We secondly, present the proposed router architecture and the proposed schemes for data-flow and congestion control. We will then focus on the evaluation of the flow-control mechanisms in terms of QoS and congestion control. Finally, we discuss the hardware characteristics of our proposed router before ending with a conclusion and directions for future work.

2. Overview of related work

NoC communication, which is characterized by high bandwidth, has recently become an active research area, mainly regarding the problems of data-flow control and congestion management. The main goal is to improve the efficiency of bandwidth allocation while avoiding router saturation. Traffic balancing based on a congestion-aware, adaptive routing process is an interesting approach to achieve load fairness (Wang et al., 2012). This solution often involves an exchange of load states and link utilization between neighbors, optimizing the routing process according to these exchanged data.

The authors in Lotfi-Kamran et al. (2010) studied a new solution for congestion avoidance based on dynamic XY routing (DyXY). In their approach, called Enhanced Dynamic XY (EDXY) routing, each router signals congestion to its neighbors through a two-bit bus. The presented solution keeps all data flows from trying to share the same path, consequently reducing data traffic congestion. Performance analysis of this solution has demonstrated good results for packet latency across these routers, outperforming the direct XY routing approach. However, this scheme did not address the problem of congestion as a general phenomenon, which might affect a network even with traffic distribution.

In Wang et al. (2013), a scheme using an energy- and buffer-aware adaptive routing algorithm (EBAR) was proposed to distribute thermal energy in a NoC. The main idea is to share data traffic for optimal thermal distribution across the NoC, while maintaining fair communication performance. For that purpose, the routing of flits is based on the thermal state of the next hop, avoiding high-temperature routers in the communications path. Thermal management is a critical problem in the design of integrated systems on chips (SoCs), but from a communications point of view, the proposed approach does not address congestion problems.

Congestion-awareness techniques based on signaling have been proposed as an efficient method to share locally the state of router buffers (Aci and Akay, 2010; Kaddachi et al., 2008; Daneshtalab et al., 2012). The intention is to avoid congested areas in the communications path. Applying this method requires broadcasting congestion information using extra buses or with additional data carried by the flits. We think that this method is interesting; however, we believe that it should be enhanced with the application of an adequate flow-control mechanism.

Ebrahimi et al. (2012) have studied a new algorithm for the routing process that uses local and non-local network information. The proposed scheme, called the Congestion Aware Trapezoid-Based Routing Algorithm (CATRA), defines a set of nodes that are likely to be involved in the data-communication path based on their congestion status. The congestion information is diffused through an extra bus that allows real-time updating without increasing traffic. The paper discussed the hardware implementation of this scheme, demonstrating that the additional cost to implement the proposed idea is reasonable in terms of circuit area and power consumption. While this approach seems attractive for NoC implementation, the paper did not show any enhancement to end-to-end QoS at the application level, which would be the main argument to justify its adoption.

An approach to flow control was proposed by Becker et al. (2012). The presented solution aims to control the input buffers occupation per data flow. The proposed scheme determines the credit allowed for each virtual channel based on performance observations. The authors demonstrated the efficiency of this approach, but they did not study its power consumption.

The authors in Peh and Dally (2000) suggest a data-flow control process using a flit-reservation mechanism that reserves buffer space and link bandwidth for the data stream. The reservation mechanism increases the efficiency of buffer-space utilization. In fact, the reserved buffer space is dynamically changed, which means that the buffer will be free for the next transmission for another data flow. This method minimizes latency during the flit transaction.

A congestion-control approach for communication in NoCs has been proposed in van den Brand et al. (2007). The suggested method is based on the use of a new service defined by a strategy called Congestion-Controlled Best Effort (CCBE). CCBE controls bandwidth usage to deliver flits with constant and reduced latency, using link utilization as a congestion metric. In fact, a centralized predictive algorithm (Model Predictive Controller, or MPC) constantly supervises link utilization as a congestion metric to determine the load of the connection. The measurements obtained by hardware-analysis probes are sent to the MPC, which decides CCBE loads based on the predictions it makes using this information.

Additional research has addressed the problem of congestion avoidance and load sharing in NoCs (Mishra et al., 2011; Ascia et al., 2006; Kumar and Mahapatra, 2005; Rijpkema et al., 2003). However, we think that a few contributions have involved per-flit flow-management schemes that might efficiently manage congestion while keeping the best QoS granularity. We believe, then, that it is necessary to design an enhanced router architecture that implements an adequate data-flow control approach along with a suitable internal-processing scheme for each per-hop transaction. The remainder of this paper presents our contribution in this regard, addressing the design of a micro-architecture for an on-chip router with enhanced capabilities to avoid and solve congestion while considering the end-to-end QoS.

3. Proposed router architecture

Fig. 1 presents the general internal architecture of the proposed router for on-chip communication. It is an enhanced version of the architecture studied in Adel et al. (2014). The main new contribution concerns the integration of a flow-control mechanism and the adequacy of the architecture for efficient end-to-end QoS.

The proposed architecture is a modular, scalable system that integrates new control and data flow-management functions. It performs per-flit processing with respect to characteristics of the data flow, which is a novel approach. Flits layout includes a set of fields making it possible to perform granular routing, allowing the router to handle passing flits according to their constraints.

In this architecture, communication flows are processed according to their class of service, with classification examined as flits queue in the internal memory of the router. Scheduling flits into the routing process as well mechanisms related to congestion avoidance are based on this classification. When congestion is detected, a specific mechanism unloads the memory according to the importance of the flits. A tag is then associated with the importance of the flit payload in the packet (i.e. priority tag: pr_tag). In addition, a router experiencing congestion will instruct its neighbors to update their routing tables. This approach delegates more autonomy to the router to make local decisions regarding congestion management, thus optimizing the use of network resources for efficient data delivery.

The specified architecture comprises different blocks, each implementing an internal task designed to process flit traversing the hop. All of these internal components are synchronized by an external clock signal.

- *The input FIFO:* these input queues are used to keep asynchronous incoming flits from neighboring routers. Their depths define the capability of the routers to absorb bursts of incoming flits. In terms of structure, the input FIFO(s) are physically independent and managed separately.
- *The flit classifier:* this block is responsible for identifying flits according to their class-of-service identifier (id_serv). Every clock cycle, it extracts flits from the input FIFO(s) that connect the router to its neighbors. The id_serv is then used to in-queue the received flits to their corresponding virtual in-queue in a common class of services-memory used for this purpose. This classification helps later to perform flit-output delivery according to service constraints.
- *The central memory:* this memory in-queues flits in different classes of service (see Fig. 1). Extraction of the flits for forwarding to their destinations is performed by the output scheduling process, proportionally to the weight of their classes of service. *The memory occupancy manager* permanently controls the depth of the in-queues in memory,

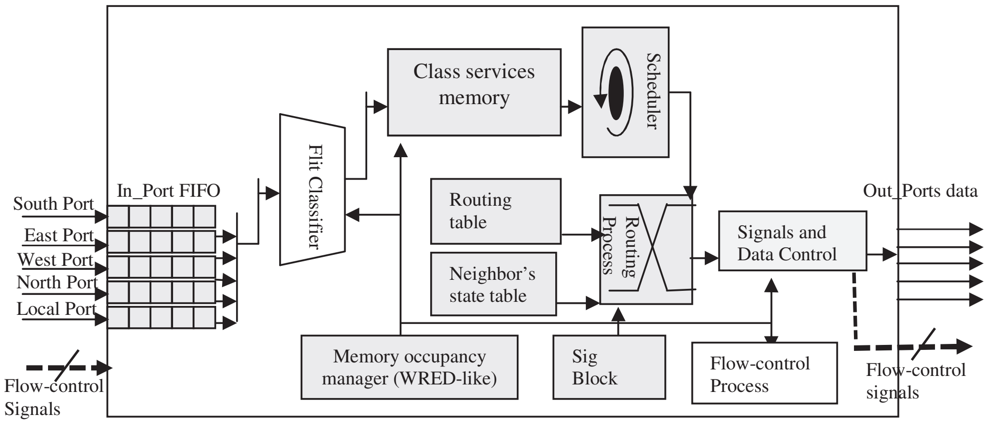


Figure 1 Internal hardware architecture of the router.

estimating the memory occupancy state. Above a certain occupancy threshold, the memory is considered fully loaded, and this process starts to select from the in-queue tails less significant flits from each communications process to be dropped in order to avoid router congestion.

- *The scheduler process:* this block links the memory with the routing process. It extracts flits from the memory outputs and forwards them to the routing process. It functions based on a Weighted Round Robin (WRR)-like algorithm. In fact, it associates a weight (W*i*) for each service class *i*. The flits are extracted from the four classes of services proportionally to their weight (W*i*). The idea is to handle the incoming flits according to their quality-of-service constraints while maximizing the router's output rate.

In the proposed architecture, in each clock cycle, the routing block is able to process at maximum five flits with different destination addresses. The scheduling mechanism aims to maximize the number of flits processed in each clock cycle. Optimizing the router's output delivery will reduce the memory load. In every clock cycle, the output scheduler extracts, from the memory, the highest number of flits with different destination addresses that can be forwarded in one clock cycle to the different output ports of the router. If we consider that, in an output scheduling cycle, the number of flits with different destinations in-queued into the class of service (*i*) is defined by *Flit_class_i* (Eq. (1)):

$$Flit\_class\_i = \sum_{j=0}^{(MaxClass_i)-1} \mathrm{flit}(j) \tag{1}$$

This equation expresses the maximum number of flits that the scheduler might extract from one class of service. $MaxClass_i$ in Eq. (1) is defined by the weight assigned to the class of service (*i*) when performing the WRR-like algorithm.

$$MaxClass_i = \omega_i; \quad \text{where } i \in \{1..4\} \tag{2}$$

In this proposed router architecture, the routing block triggers the output scheduler process. When the scheduler is triggered, the number of flits it will transfer to the routing block during one scheduling cycle is expressed by Eqs. (3) and (4). All selected flits should be assigned different output ports:

$$NbrFlit = \sum_{i=1}^{4} \sum_{j=0}^{(MaxClass_i)-1} \mathrm{flit}(j) \tag{3}$$

$$NbrFlit \leqslant 5 \tag{4}$$

As expressed in this equation, in one output scheduling cycle, all classes of service might be involved. So, while this approach based on a WRR-like mechanism allows the router to send out a variable number of flits from different classes of service according to their weights (W*i*), it also maximizes the total number of flits that are managed by the router in one routing cycle. Consequently, it minimizes the total in-queue length of flits waiting in memory, which in turn reduces transaction time and avoids congestion.

- *The routing block:* this is the main block in the NoC router. It routes the different incoming flits based on the wormhole technique and allows channel multiplexing using the concept of virtual circuits, which improves bandwidth allocation through the Time Division Multiple Access concept (Ascia et al., 2006; Palesi et al., 2006).

The communications process starts with the transmission of the *head-data flit*, which contains information about the process identifier (*id_proc*), source address, and destination address. The information in this flit is used to establish a virtual circuit for the remaining data flits. Each router has an internal representation of all the neighboring routers traffic load. The routing algorithm will, then, favor the application of a direct *X–Y* routing scheme to forward the header flit to the next router in the path. If the next hop is congested, an adaptive *X–Y* scheme based on neighbors' states will be used to find an alternative next hop. The current router will update the *routing table* with the two pieces of information (*id_proc, output_port*). All the following incoming data flits with the same *id_proc* will be forwarded through the same channel, using the updated information in the *routing table*. After receiving the *end_data_flit*, the created virtual channel is terminated.

- *The signaling block:* this block implements some tasks related to the signaling process in the network. It generates and broadcasts signaling from asynchronous flits in order to update neighboring routers with new traffic load states. This block analyzes newly received signaling flits in order to update the internal routing table with the neighboring routers states. The information stored in this table is required to manage a head-flit during the path-establishment process. This block interacts with the memory controller, in order to obtain the updated state of the memory.
- *Memory occupancy manager block:* this block uses an approach based on Weighted Random Early Detection to drop low-importance incoming flits when required in order to avoid memory congestion. When memory occupancy is above a maximum threshold, it starts to drop low-importance flits from memory and from the output of the flit classifier based on their service classes (*id_serv*) and importance tags (*pr_tag*). Priority is scaled out of four for all classes of service and attributed by IP cores (Intellectual property) when the application data stream is generated. The *memory occupancy manager* starts by dropping flits with the lowest priority (pr_tag = 0). If all the flits with the lowest priority are dropped and the router is still congested, incoming flits of the next highest priority begin to be dropped to reduce the time needed to resolve congestion.

4. General protocol structure and flit types

In the proposed communication protocol we define two flit types: data flits and signaling flits. In order to maintain scalability and match the hardware data bus and internal router buffer constraints, we have chosen a fixed 32 bits flit size for all flit types.

Data flits are dedicated to the transport of data between IPs. During the communication process, three types of data flit are required:

- *Head-data flit:* the wormhole commutation technique is based on route establishment (with a physical multiplexed channel) (Ni and McKinley, 1993) between source and destination IPs. In the proposed architecture, route establishment takes into account some information related to the expected process, such as the class of service identifier (*Id-serv*). Furthermore, route establishment is mainly enhanced by the study of the physical states of the expected in-path NoC routers. This information, quantifying the available memory space and the length of the in-queue waiting flits in the considered class of service, is provided through the signaling process between neighboring routers. Fig. 2 depicts the head-data flit used in our approach.
- *Continuation-data flit:* this flit type is processed after the establishment of the communication path between source and destination during packet transmission. It transports the data payload. In this flit, the process identifier *Id_proc* is used to identify the output port of the connected routers, using a local switching table that must be updated at the start and end of each process.
- *End-data flit:* this has the same structure as the data flit, but its arrival triggers the release of resources reserved for the current communication process.

Signaling flits are generated by a router upon state change. They carry useful information between neighboring routers in order to provide each router with a virtual representation of its neighbors.

5. Approaches to flow control and congestion avoidance

In a Multi-Processor System-on-Chip, the end-to-end QoS (e.g. end-to-end delay, jitter, and number of lost flits) depends heavily on the capacity of the router to handle, with required granularity, the flits of a communication process (Bolotin et al., 2004). The internal tasks associated with flit management during per-hop transition affect delay and jitter. The depth of the waiting flits is directly related to the gap between the input and output throughputs. Therefore, the first strategy for avoiding a router bottleneck is to reduce as much as possible the per-flit processing time. Even though the processing time of SoCs is very short due to their high frequency, the demands in bandwidth for multimedia applications are very high, which requires that internal processing tasks be adequately efficient in order to achieve short responses.

The QoS warranty in NoCs requires to be considered at different levels in the design of the router. Not only does it involve dimensioning of hardware resources, but it also requires a whole communication strategy for traffic control and congestion avoidance. The per-hop memory occupancy during communication is highly dependent on the strategy applied to balance the data-traffic load based on the communication-path state.

Our approach applies a flow-control mechanism along with a service-differentiation scheme suitable to schedule the flits for output according to application requirements. For that purpose, the services in the proposed MP-SoC architecture are divided into four classes according to QoS constraints defined by the application. Table 1 specifies the service differentiation and the associated scheduling-output weight.

Splitting the traffic into different classes of services allows the scheduler to apply generic output scheduling policies to control the output rate. This approach is suitable for the internal processing of QoS constraints associated with classes. During the communication process, flits are stored in the routers internal memory based on their class of service. Upon their arrival in the router, the *Service class classifier* reads the injected flits from the input ports of the router following the selected round-robin method. Based on the service identifier field (id_serv), the flit is in-queued in the appropriate class of service in the internal memory, which is organized into four waiting in-queues that are managed using specific pointers and indexes.

For the queued outgoing flits, the scheduler treats each class of service using a corresponding weight (Wi) in order to apply a WRR policy. Table 1 outlines the classes of service defined for this approach, along with their weights for the output scheduling process.

Congestion control in the NoC is basically dependent on the capability of the router to avoid overflow of its internal memory. In the proposed scheme, the *Memory occupancy manager* controls the memory load. When the *lower threshold* is reached, the process interprets the state of the memory as overloaded and starts selectively dropping incoming flits based on their importance. Moreover, a signaling flit is generated and sent to alert its neighbors about its overloaded state, asking

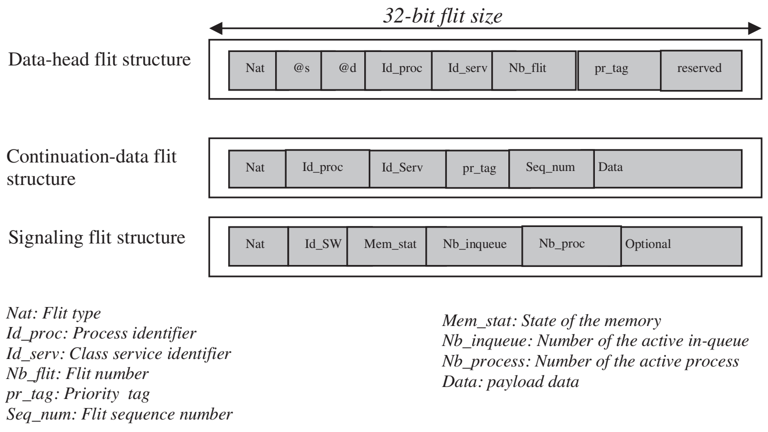


Figure 2 Flit structure in the proposed communication architecture.

Table 1 Service classifications and their associated weights at the output scheduler.

Classes of service	Service identifier	Description of the service	Associated weight at the output scheduler (W*i*)
Signaling flits	Class_1	Control and management of network IP cores	4
Real-time flits	Class_2	Time-sensitive applications, such as multimedia applications	3
Short data flits	Class_3	Used to transfer short information, like register content	2
Block memory transfer flits	Class_4	Streaming large amounts of data between IP cores	1

them to avoid using it as part of their communication paths. Low-importance flits are discarded using a WRED-like algorithm (Barbera et al., 2008; David et al., 2011). The dropping process selects low-importance flits according to the information embedded in the flit (*pr_tag*), which is generated by the application based on the importance of the payload. Fig. 3 illustrates the general organization of the internal processes during flit processing.

The proposed mechanism for discarding low-priority flits helps a router experiencing congestion to unload its memory with less impact on the QoS. Indeed, this selective process of packet discarding helps to reduce per-flit transaction time, while keeping significant data that is useful to the receiver application. However, we think that neighboring routers using the congested router as a next hop in a data-communication path should also reduce their upstream data rates to avoid burstiness. This paper presents, then, two mechanisms for NoC flow control that contribute to avoiding and solving congestion. The first proposed scheme for flow control is based on the use of an output buffer, slowing down the output stream when the next hop experiences a congestion problem. The second proposed scheme is a bufferless approach, based on a feedback control scheme between the congested router and the source core of the data in order to reduce the sending window size.

5.1. *Flow control based on the use of an output buffer*

In the output buffer flow-control approach, a buffer at the output level helps to slow down the frequency of the upstream data when the next router is congested (Fig. 4). For that purpose, we define three levels that quantify the load of a router: under-loaded, loaded, or congested. So, upon receipt of a signaling flit that indicates a loaded memory state in the next router, a router activates the output buffer to control the output stream rate. Flits are then injected into the flow-control buffer from either index_1 or index_2, representing two positions in the output buffer. Flits are injected to index_1 when the next router is over-loaded (congested), while index_2 is used when it is in a loaded state. At every positive edge of the clock signal, flits are time-shifted to the output port. The flits are injected at index_3 if the flow-control mechanism is not activated. The idea is to introduce some clock cycles before sending out the flits with the hope that giving more time to the next router to forward flits from its memory to their destination would avoid or resolve the congestion.

5.2. *Flow control using a feedback-signaling mechanism to the source core*

In this approach, the congestion event is signaled to the source of the data in order to reduce its throughput. The router experiencing congestion determines first from among all the carried communication processes which one is injecting the largest amount of data. It then informs its neighbors of its congested state to help avoid the arrival of more, newly routed flits. The signaling information in this scheme is carried in an extra band; in fact, a specific bus carries this information to the routers neighbors and consequently also identifies the process (P_c) that caused the congestion.

Among its neighbors, the router (R_{n-1}) that is sending the flits of the communication process (P_c) sends back the signaling event to the router R_0 and then to the source of data (*IPsource*; see Fig. 5).

The router experiencing congestion calculates the input injection rates for all carried flows at the time when congestion

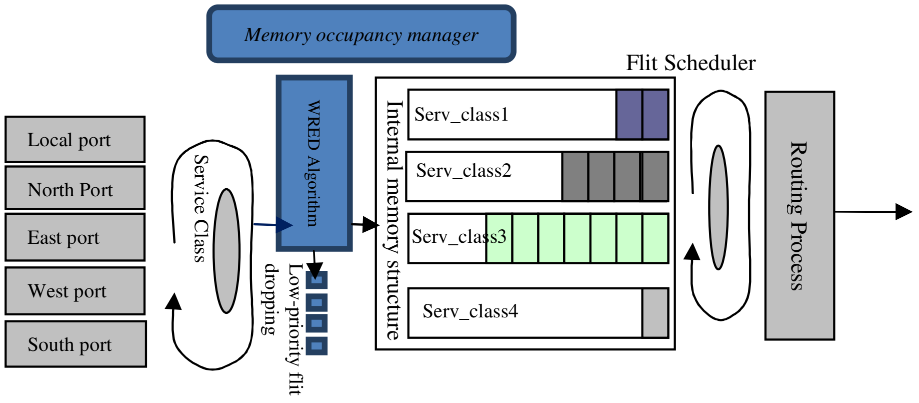


Figure 3 General approach for memory management.

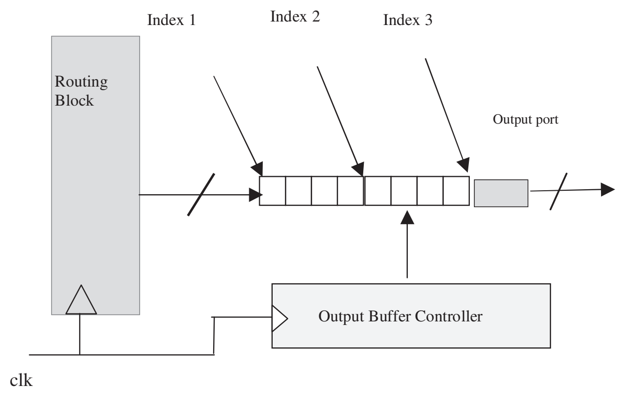


Figure 4 Architecture of output-buffer flow control.

is detected. It locates the flow with the highest input-injection rate and notifies it to reduce its traffic load. In our approach, a communications process is defined by a data Flow *Fi* and identified by (proc_id). The router maintains an internal table, the structure of which is shown in Table 2, where the input-injection rate is updated continuously for each communication process.

Let us note by *Fi* the data-flow of the communication process (*i*). We define the injection rate Proc_InjRat(*i*) as expressed in Eq. (5):

$$\mathrm{Proc\_InjRat}(i) = \frac{Num_{flits(i)}}{T_{Clock_{cycle}}} \tag{5}$$

For process I, $Num\_flits(i)$ represents the number of flits injected into the router over the last period of time (T_{Clock_cycle}).

The congestion bus carries the signal all the way through until it reaches the hardware interface of the IP source, as shown in Fig. 5. The IP source interface then reduces its injection rate according to the algorithm shown in Fig. 7.

The IP source interface continuously checks the *congestion_bus* in order to detect if it is the source of congestion for a router in the communication path (Fig. 6). When this happens, the interface adapts its inputs according to the control algorithm in Fig. 7. The source IP interface first reduces the injection rate by half (Fig. 7, line 3). After a period of T_{Clock_cycle}, the IP source keeps the same reduced injection rate if the state of the router is still in congestion (Fig. 7, line 7). Otherwise, the injection rate is increased slowly until it reaches the maximum value (*congestion_injection_rate*; Fig. 7, line 10). Every period of T_{Clock_cycle}, one more flit is injected in that cycle than in the cycle before. When the injection rate reaches the threshold, the interface increases the injection rate following the congestion-avoidance mode as expressed in (Fig. 7, line 13).

The pseudo code, $N * T_{Clock\_cycle}$ represents a multiple N of the time period on which we update the injection rate. The more we increase the value of N, the more slowly the injection rate increases, avoiding congestion. This value N can be a dynamic parameter to fine tune the injection rate. The optimal value of N should allow high throughput while avoiding congestion. In our study, we have chosen ($N = 4$) as optimal based on simulations that tested the impact of N on the input-injection rate. We note that greater values of N can be also applied in this algorithm.

The proposed algorithm favors best effort mode in the behavior of the routers which maximizes the bandwidth allocation. However, when a congestion is detected, the proposed scheme obliges the main source of this congestion to reduce heavily its injection rate (half of the current injection rate) that will be kept until the end of congestion. After solving congestion, the source starts increasing, periodically, its input injection rate with one flit every T_{Clock_cycle} until reaching the initial measured *congestion_injection_rate*. This phase avoids to quickly saturate the network again and aims to maximize the throughput. Beyond this threshold, the source applies the congestion avoidance mode to increase smoothly its input-injection rate as explained in the previous paragraph.

The characteristics of the two proposed schemes for congestion control in on-chip networks are different. The output-buffer scheme is a local mechanism operating between a congested router and its neighbors. It uses an in-bandwidth signaling mechanism based on the exchange of flits that carry congestion information. The second flow-control mechanism, on the other hand, involves an interaction with the source core to reduce its injection rate.

Table 3 summarizes the main characteristics of and differences between these two flow-control modes.

In the following part of the paper, we evaluate the performance for a multimedia application transmitted with the application these two proposed flow-control schemes for avoiding congestion in NoCs.

6. Performance analysis

The performance of the proposed router architecture was evaluated with both of the presented flow-control schemes for congestion avoidance. In a co-simulation environment based on Simulink-ModelSim tools, we built a 3 * 3 mesh NoC using the described router architecture. The Register Transfer Level (RTL) – VHDL description of the router was developed, simulated with ModelSim, and used to link the different cores of the network. For each core of this network, an adapter (interface) was designed in order to shape the flits and control the injection rate.

Between the cores of this network, many communication processes were defined in order to test the performance of the flow-control schemes under injection at variable rates, with changing data-traffic loads to create congestion states. We evaluated the schemes mainly based on their capabilities to manage QoS and to provide better performance at the application level when the network experiences congestion. The simulation focused on the evaluation of QoS metrics for a specific, real-time data-flow that was transported over the network with other, simultaneous communication processes. The specific test application involved sending an image and retrieving it at the receiver core.

6.1. Performance of the WRED-like algorithm

We first evaluated the performance of the designed WRED-like algorithm for discarding low-importance flits to unload memory and avoid congestion during heavy traffic. To test the capability of this algorithm to resist congestion under heavy loads, we injected into one router many communication processes on different input ports. Fig. 8 shows the output of

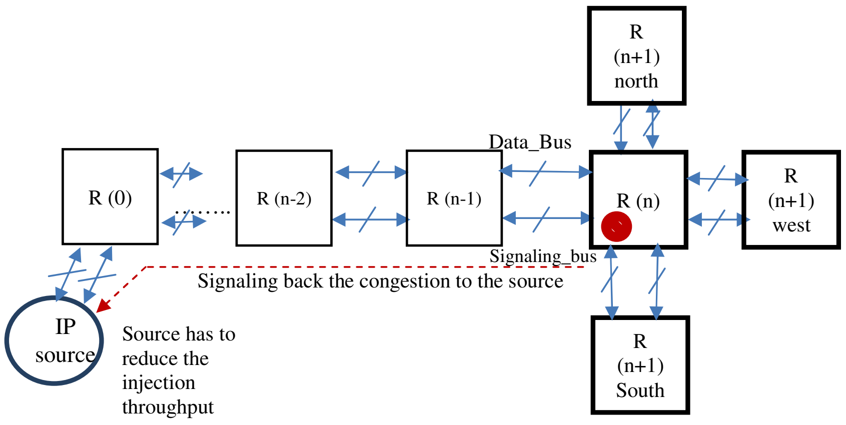


Figure 5 Flow control with feedback to the source.

Table 2 Structure of the routing table.

	In_port	Output _port	Class_id	Injection rate
Id_proc_1	Proc1_in_Port_id	Proc1_Out_pot_id	Proc1_Class_id	Proc1_inj_rate
Id_proc_2	Proc2_in_Port_id	Proc2_Out_pot_id	Proc2_Class_id	Proc2_inj_rate
Id_proc_n	ProcN_in_Port_id	ProcN_Out_pot_id	ProcN_Class_id	ProcN_inj_rate

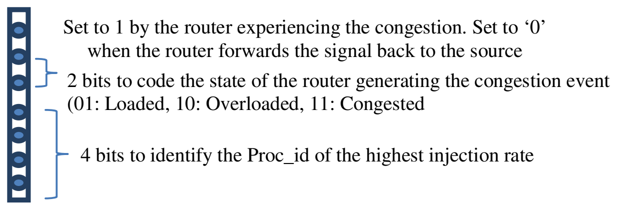

Set to 1 by the router experiencing the congestion. Set to '0' when the router forwards the signal back to the source
2 bits to code the state of the router generating the congestion event (01: Loaded, 10: Overloaded, 11: Congested
4 bits to identify the Proc_id of the highest injection rate

Figure 6 Structure of the congestion signaling bus.

the router for two different injection rates (0.23 flits/clk and 0.19 flits/clk). The instants T1 and T3 are the times when congestion is detected in the router memory for injection rates of 0.23 flits/clk and 0.19 flits/clk, respectively, without the use of the flit-discarding algorithm. When the WRED-like algorithm is applied, congestion occurrences are delayed for both input rates, respectively, to T2 and T4. By using the selective flit-discarding process, the router delayed the congestion by more than 180 and 200 clk cycles for input rates of 0.23 and 0.19 flits/clk, respectively.

The ability of the WRED-like algorithm to delay congestion is higher with a lower injection rate. In fact, in this case, more time is allowed for the routing process to forward flits from memory to their destinations. The WRED-like algorithm is applied locally, in the internal memory of the router experiencing the congested state. Even if this process is able to delay the congestion (as in the case of 0.23 flits/clk), it cannot avoid injecting a high data load to the router. Thus, we think that an appropriate data-flow control mechanism has to be applied along with this approach.

```
If (Congestion_bus ( router_state ) = "congested" then
    Congestion_injection_rate := current_injection_rate;
    New_injection _rate: = $\frac{\text{current\_injection\_rate}}{2}$
    Old_injection_rate =: New_injection _rate
    Wait ($T_{Clock\_cycle}$) ;
  If ( Congestion_Bus( router_state ) = " Congested" ) then
    New_injection _rate := Old_injection_rate;
  Else
    If ( new_ injection rate < Congestion _injection_rate ) then
      New_injection_rate := New_injection_rate + $\frac{1}{T_{Clock\_cycle}}$;
    End if ;
    If ( New_ injection rate >= Congestion _injection_rate ) then
   New_injection _rate := New_injection _rate + $\frac{1}{N * T_{Clock\_cycle}}$
    End if ;
  End if;
End if ;
```

Figure 7 Pseudo-code of the injection-rate control algorithm.

Table 3 General characteristics of the two proposed flow-control schemes.

	First scheme: output buffer	Second scheme: feedback control of source core
NoC architecture	Mesh architecture but also scalable for other architectures	
Buffer	With output buffer	Buffer-less approach
Signaling of congestion	In-bandwidth signaling	Usage of extra hardware bus
Congestion awareness	Fast broadcasting of, in bandwidth, signaling flits to neighbors	Very fast signaling mechanism based on hardware signals generated between routers
WRED-like algorithm for selective packet discarding	Applicable	
Scalability	The two proposed schemes are scalable for the size of the network. In fact, both of them aren't depending on the number of routers in the network	

To compare with similar work, the authors in Lee et al. (2012) implemented Globally-Synchronized Frames to ensure QoS in NoCs. In (Lee et al., 2012), congestion occurred for an input rate of 0.22 flit/clk while adopting different arbitration approaches. Our proposed solution without the application of a flow-control scheme became congested for a similar input rate (0.23 flits/clk). However, the WRED-like algorithm delayed the occurrence of congestion, which may facilitate the efficient delivery of QoS at the reception level. In addition, the authors of Lee et al. (2012) did not discuss a specific mechanism to manage or avoid congestion.

6.2. Performance of the router with the application of the flow-control mechanisms

The ability of the proposed router to avoid congestion in the network was evaluated with the two proposed flow-control schemes. Performance was analyzed in terms of its capability to enhance the QoS observed at the application level. For this purpose, a communication process was dedicated to the transmission of an image of 512 * 512 8bpp that was transformed using the Haar wavelet transform (HWT) and injected into the 4 * 4 NoC mesh architecture. The use of the HWT generates flits of variable levels of importance for the retrieval process (Fig. 9). The data flow representing the image after decomposition contains four sub-bands (LL, LH, HL, and HH). Accordingly, flits were generated and injected into the network with variable importance tags (*pr_tag*). The flits generated from the sub-band LL carry the low-frequency data most required for the retrieval process; therefore, they are assigned the highest importance. The two sub-bands LH and HL contain data that are less important, compared with the LL sub-band data, to reconstruct the image. The least-significant flits injected into the network are those transporting data from the HH sub-band.

The HWT decomposition tests the effect of the proposed WRED-like algorithm to select low-weight flits to discard from the data stream in order to avoid memory saturation in the routers. In addition, other communication processes were simultaneously carried on the network in order to increase the per-router loads and to cause congestion states in the data path. Table 4 sums up the features of the data streams injected on the network during the simulation.

The proposed flow-control schemes were applied, along with the WRED-like algorithm, to discard low-importance flits from transitional nodes when congestion symptoms were detected. We evaluated the capabilities of the two proposed schemes of flow control to enhance the QoS at the reception level. For the transmission of the Lena image, we have estimated variation in jitter, which expresses the capacity of the routers in the path to maintain periodic delivery of their incoming flits. Fig. 10 shows the maximum absolute values of jitter measured for the different schemes of flow control at

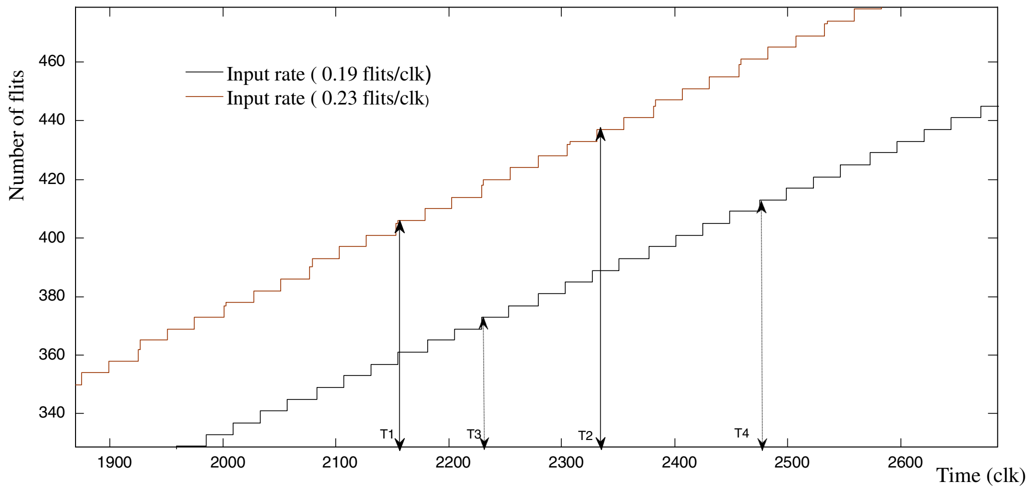
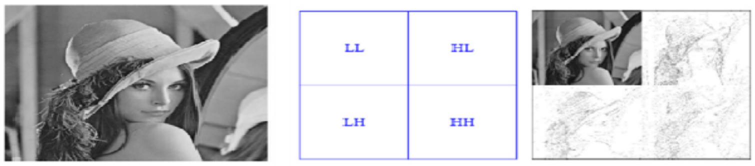


Figure 8 Capability analysis of the WRED-like algorithm for congestion delay in the router.



(a) Lena image (b) Frequency Sub-bands obtained with HWT

Figure 9 Output stream after the decomposition with HWT for the Lena image.

Table 4 Features of the test data flows.

Process identifier	ID class of service	Input injection rate to the network (flits/clk)	Source–destination (X–Y)	Period of time (T_{Clock_cycle})	General description
1	1	Variable from 0.01 to 0.1	(00, 00) to (01, 10)	100 CLK cycles	HWT applied to Lena output data stream
2	2	Variable from 0.015 to 0.1	(00, 01) to (10, 01)		Extra communication processes injected to the mesh network during simulation to load the network
3	3	0.02	(01, 00) to (01, 10)		
4	4	0.02	(10, 00) to (00, 10)		
5	1	Variable from 0.01 to 0.1	(10, 00) to (01, 10)		

the reception level under different network loads. Under low network load (0.12 flits/clk cycle), the network delivers the flits to the receiver with almost the same maximum jitter. However, under higher network load, which causes congestion (from 0.56 flits/clk cycles), the scheme based on a feedback-signaling mechanism to the source core ensures less jitter compared to the output buffer flow-control mechanism or compared to communication without any flow-control mechanism.

At a load of 0.56 flits/clk cycle, the maximum measured value of jitter is higher when applying the feedback-signaling control mechanism than when applying either of the two other schemes. In fact, at this load the memory of the router is lightly-loaded, the output-buffer flow-control mechanism starts time-shifting the flits at the output (index 2) without reducing the injection rate of the source. This high input-injection rate favors injecting more flits of the same data-flow in the congested router and consequently delivering more flits to the reception. However, the feedback flow-control mechanism reduces the injection rate of the source by half and then it starts increasing it slowly to avoid congestion. As a result, for this load of 0.56 flits, the jitter measured with the feedback flow-control mechanism will be increased because of this considerable reduction of the input injection rate as well as the application of the WRED-like algorithm. Fig. 10 shows that the jitter is higher than the measured values with the two others flow-control schemes.

When the network is heavily-loaded, the output buffer cannot avoid high values of jitter. It time-shits the flits using the position index 1 in the output buffer that will saturate quickly because of the high load of the router. Thus, this scheme will not be able to avoid high number of discarded flits with the application of the WRED-like algorithm (since the injection

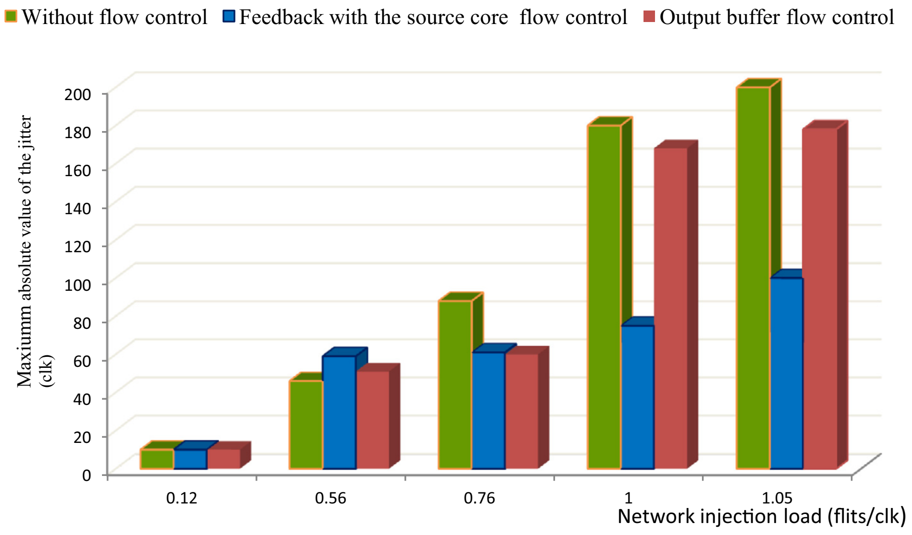


Figure 10 The maximum absolute value of jitter with different flow-control schemes.

rate will remain high into the router). Then, the jitter will increase significantly compared to the use of the feedback-signaling flow-control mechanism that will keep low measured values of the jitter compared to the two others flow control schemes.

The measured values of the jitter with the output-buffer based flow control scheme are slightly better than the communication without flow control schemes. But, the two schemes keep the same trend of increasing jitter with different data-traffic loads. In fact, without reducing the data-source injection rate, the memory of the router will be heavily-loaded which increases the probability of discarding flits. In that case, the output-buffer used in the neighbors will be saturated after a short time and its function of flits time-shifting will not reduce the injection rate in the congested router.

In Fig. 10, the feedback flow-control scheme was able to efficiently avoid high jitter experienced at the receiving application. This figure shows a gain of around 43% under a network load of 1.05 flits/clk cycle. This result attests to the efficiency of the feedback-signaling mechanism to the source core.

End-to-end delay is one of the most interesting QoS metrics used to evaluate the time efficiency of a communication system. We have evaluated the time performance of the discussed schemes and their capabilities to ensure delivery to the receiving application without delay. Fig. 11 presents the measured end-to-end delays for the two flow-control schemes, as well as the results obtained without the application of a flow-control mechanism. The end-to-end delay increases with increasing data-traffic load, reflecting more buffering time in the routers. The application of an output-buffer flow-control mechanism does not reduce end-to-end delay compared to the results obtained using the feedback-signaling mechanism to the source core. Buffering at the output of neighboring routers slightly reduces the measured end-to-end delay, but this gain is less than with the application of the feedback-signaling mechanism. Under network loads that cause congestion ($\geqslant 0.56$ flits/clk cycle), the measured end-to-end delay with the application of feedback signaling to the source core remains less than the measured values with the scheme using an output buffer.

We noticed also, from Fig. 11, that the end-to-end delay is not significantly enhanced with the application of the output-buffer flow-control mechanism compared to the communication without flow control. In fact, as explained in the interpretation of Fig. 10, since the input-injection rate at source is not reduced, the memory of the congested router remains with high data-load as well as the memory of the others router in the data path. Thus, the end-to-end delay is slightly reduced with the application of the output buffer-based flow- control approach.

The results shown in Fig. 11 suggest that the feedback-signaling mechanism is a more scalable and efficient flow-control mechanism compared with the output-buffer scheme.

The performance analysis also measured the number of flits lost through a congested router (Fig. 12). The application of the output-buffer flow-control scheme reduced the number of lost flits. The feedback-signaling scheme to the source core also yielded fewer lost flits in the congested router. When compared with communication without a specific flow-control mechanism, the number of lost flits is reduced by more than 40%, and the feedback-signaling mechanism reduces lost flits by more than 20% when compared with the output-buffer scheme. As an important reminder, the WRED-like algorithm is applied to both of the studied schemes to drop from congested memory flits of low importance to the retrieval of the image.

This selection of low-importance flits increases the quality of the received image. Fig. 13 shows the different measured Peak Signal-to-Noise Ratios (PSNR) for the retrieved image at the receiving application. The quality of the image is substantially enhanced by applying both proposed schemes of data-flow control. However, applying the feedback-signaling scheme specifically as a flow-control method obtained more improvement in the measured PSNR when the network is overloaded. In fact, the value of the PSNR remains higher than 25 dB in such a scheme, a threshold that corresponds to a good-quality received image. The gain in the PSNR of the received images with the application of the feedback flow-control scheme is more than 34% for a data traffic load of 1.05 flits/clk, compared with transmission without flow control.

Fig. 14 shows the results of the throughput measured at the reception core for the image-communication process. For an under-loaded network (0.12 flits/clk), the throughput at the receiver level is almost the same as that achieved by the two proposed flow-control schemes. As network load increases, some routers become congested (loads of 0.56 flits/clk and above). The two proposed flow-control schemes act differently in this case. In the scheme with feedback-signaling to the source core, the congested router asks the core source with the highest injection rate to reduce its injection rate, as described above. However, in the output-buffer flow-control scheme, the congested router just asks its neighbors to delay forwarding the routed flits. This explains why the reception throughput is low under the feedback-control scheme. However, when the congestion is severe (under high traffic loads), arrival throughput to the application is better than under the output-buffer flow-control scheme. In other words, combined with the other QoS metrics, even though the reception throughput is lower, the achieved QoS is better with the application of the feedback flow-control scheme.

7. Hardware characteristics of the proposed router

To estimate the hardware features of the studied router for on-chip communication, a NoC router has been designed for ASIC CMOS prototyping. We used a VHDL description at the RTL level to describe the functionalities of the router as sets of Extended Finite-States Machines. The Design Vision tool was used for synthesis, using CMOS cell libraries and the Silicon Encounter tool for placement and routing while laying out the circuit. We used the Silicon Encounter tool with integration technology 45 nm standard cell library in order to extract the main features of the proposed circuit. We estimate the area and power consumption of the NoC router.

Table 5 illustrates the main characteristics of the proposed router, assuming an internal memory size of 1024 bytes, which is able to in-queue a depth of eight flits in four internal classes of services. The results shown in that table were obtained for the router without the application of the two flow-control schemes (Fig. 1). The additional hardware cost of applying these two schemes is negligible, especially considering their QoS advantages. In fact, for a frequency of 500 MHz, they

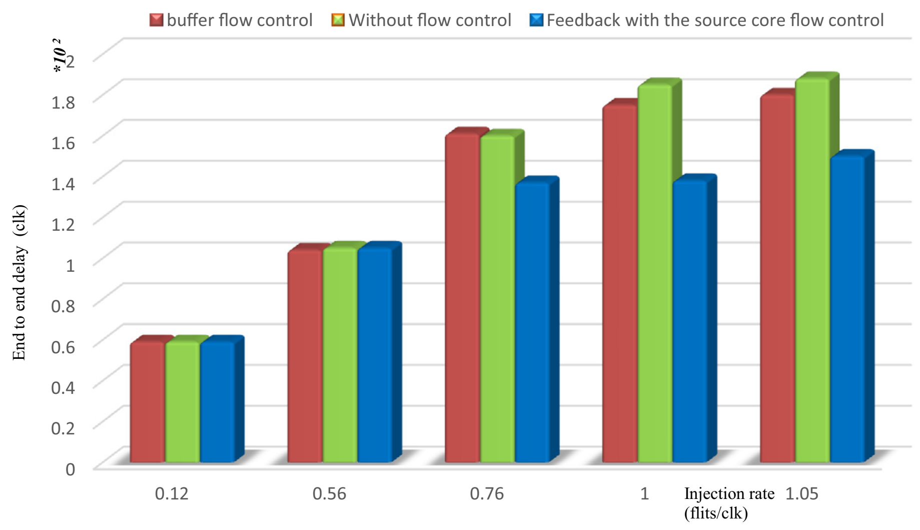


Figure 11 End to end delay measured for different flow control schemes.

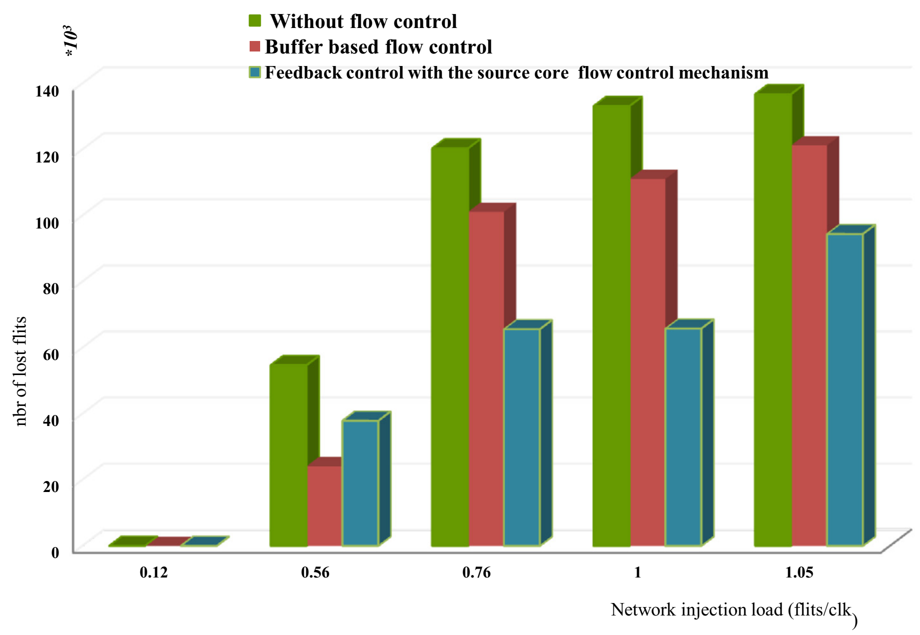


Figure 12 Dropped flits during congestion with different flow-control schemes.

occupy about 0.0002 mm^2 of the circuit area and around 0.1 mW of the dynamic power consumption.

These costs increase slightly with frequency increase, requiring about 0.004 mm^2 more between 330 MHz and 1 GHz using this integration technology. The required change in circuit area is not significant with increase in frequency. So, we think that the proposed circuit can provide high-bandwidth communication in MP-SoCs without compromising the area reserved for the processing cores (IPs).

A circuit power consumption is often determined by its dynamic part. Table 5 illustrates the dynamic power consumption of the router for different frequencies, which increases with the frequency of the circuit. As the frequency determines the available bandwidth-link between routers, it is important to note that a tradeoff has to be made between power consumption and the link communication characteristics, such as the delay and the per-router transaction time. For a frequency of 1 GHz, the proposed router will provide a bandwidth link of about 128 Gbits (frequency * link size * number of output ports).

In conclusion, the proposed NoC router achieves high communications throughput with low power consumption, which makes it a very suitable solution for efficient communication in embedded systems.

8. Comparison with related work

Coupled with QoS demands at the application level, different researchers have considered the very important topic of congestion-control mechanisms in NoCs, aiming to avoid

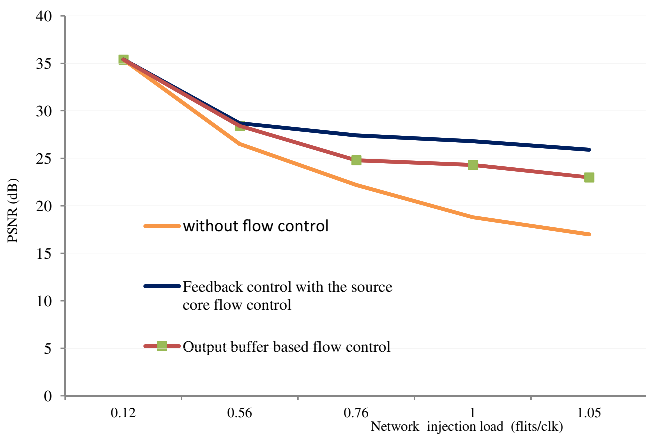


Figure 13 PSNR of the received images using different flow-control schemes.

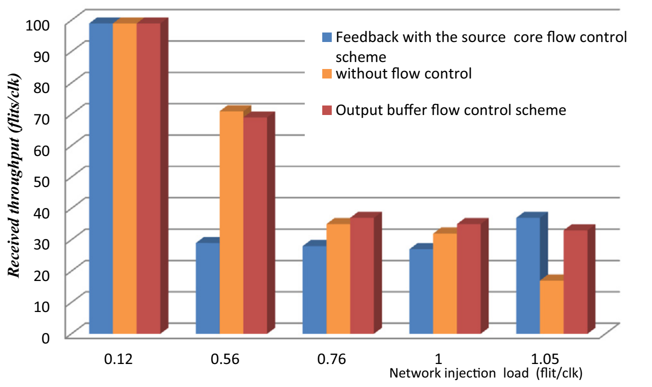


Figure 14 Received throughput at the application level for the image communication process.

Table 5 Circuit area and power estimation.

Frequency (MHz)	Area (mm^2)	Dynamic power (mW)
330	0.048	10.28
500	0.049	15.42
1 GHz	0.052	30.96

saturation and improve throughput. In this section of the paper, we compare our solution to different work from the literature.

Wang and Bagherzadeh (2014) propose a novel, QoS- and congestion-aware NoC architecture to manage QoS and to balance the traffic load inside the network to enhance overall throughput. The proposed approach differentiates the traffic into different service classes, and then bandwidth allocation is managed accordingly to fulfill QoS requirements. The proposed router incorporates a congestion-control scheme comprising dynamic arbitration and adaptive routing-path selection. High-priority traffic is directed to less-congested areas and is given preference over available resources. The simulated results of a network based on the proposed router show interesting performance in terms of load balance and per-node latency. When compared with our solution, we think that distributing the traffic load is an interesting approach, but it still cannot avoid congestion under high traffic loads. Therefore, we think that a flow-control mechanism such as the one we propose here to prevent congestion could enhance these results.

The authors in Jafri et al. (2010) address the problem of backpressureless and backpressured networks, charting the tradeoff between performance and energy consumption related to buffering. They propose a novel adaptive flow control (AFC) router that dynamically adapts between backpressured and backpressureless flow control using three new schemes: local contention thresholds, gossip-induced mode-switch, and lazy virtual circuit allocation. These different mechanisms are dynamically applied in the router according to the load state of the network. Their approach was evaluated in a 3 * 3 mesh network with a flit size of 32 bits, demonstrating through simulation that the AFC routers operate in backpressureless mode at low loads and as backpressured routers at high loads in order to avoid the significant energy/performance penalties that each of these two flow-control policies incurs when operating outside their sweet spots. Their proposed flow-control scheme looks very interesting as a concept. However, the authors did not study the hardware characteristics of an implementation of their proposed router, and they did not check its real impact on the end-to-end QoS, both of which would attest to the efficiency of the method.

The authors in Lee and Bagherzadeh (2009) designed a NoC router using 90 nm technology. The proposed circuit, using a FIFO depth of eight flits, was able to reach a frequency of 425.5 MHz with a circuit area of 0.143 mm^2 while maintaining relatively low power consumption (2 mW). The circuit design was oriented to reduce power consumption by applying the clock-boosting concept. The presented architecture was not based on deployment per class of service, and the authors did not explore the capacity of the router to avoid congestion under heavy data traffic.

Lotfi-Kamran et al. (2010) developed a routing scheme called EDXY to share traffic on the network and avoid overloading certain hops while others remain unloaded. This solution can avoid congestion and enhance the received QoS metrics, such as latency, for high injection rates. The circuit has a data link of 32 bits, the same flit size as used in our approach. It has an area of 0.089 mm^2 and the dynamic power consumption is 26.1197 mW. The approach presented in this paper looks attractive, but the hardware features of our solution are more interesting for SoC design.

Comparing our solution to the XHiNoC circuit studied in Samman et al. (2009), when using the same integration technology, our designed router occupies a bigger area. The XHiNoC router for a FIFO size of eight flits of 38 bits requires an area of 0.108 mm^2 using 130 nm integration CMOS technology and reaches a maximum frequency of 453 MHz. We think this difference in router area is due to the adoption of central memory for service classification in our approach. We note that in exchange for a larger area, our circuit offers more bandwidth and ensures higher efficiency and granularity of flow-control for enhanced QoS delivery.

In Wiklund and Liu (2003), the SoCBuS NoC was presented as a solution designed for real-time applications. The circuit clocks at 1.2 GHz, within the same frequency range as our solution, and it offers high available bandwidth between routers and enables real-time services. However, the proposed circuit implements no flow control for congestion management.

In Jovanovic et al. (2007), the CuNoC solution was presented for communication in a Multi-Processor SoC. The circuit was evaluated for FPGA prototyping. Interestingly, it provides high throughput—more than 500 Mbits—using a data link of 16 bits. However, the proposed solution integrates no specific mechanism for congestion management or QoS enhancement, so we think that our approach remains more interesting under high data traffic.

The authors in Michelogiannakis and Dally (2009) detailed the design of a router with an elastic buffer based on a master-slave flip–flop architecture. They presented three different hardware architectures (baseline two stage, enhanced two stage, and single stage) to implement this elastic buffer in order to optimize flit transaction time in the router. The application of a single-stage approach significantly reduced the latency compared with the two-stage approaches. The authors of this paper did not apply a specific flow-control mechanism and did not study the end-to-end QoS achieved by their approaches. Thus, we suggest that their proposed approaches be enhanced with an efficient flow-control mechanism to manage QoS, such as the one presented in this paper.

9. Conclusions and future work

In this paper, we have addressed the problem of flow control for congestion management and for efficient QoS delivery in NoCs. We proposed an enhanced architecture that allows per-flit differentiation and offers more processing granularity during flit transactions. Two flow-control schemes were studied to avoid and solve congestion with low impact on QoS.

The first flow-control scheme involves an output buffer to delay flit forwarding to a congested router based on a signaling process between neighbors. The second studied scheme applies a feedback-signaling mechanism to the source core to reduce the injection rate. These two proposed flow-control schemes were applied alongside a WRED-like algorithm that selects low-importance flits to be dropped out of memory during congestion avoidance and resolution. This algorithm to unload the memory of congested routers is intended to reduce distortion on the QoS at the reception level. We measured the performance of the two proposed flow-control schemes when applied to a multiprocessor architecture, which demonstrated the efficiency of the two proposed schemes. In particular, the feedback-signaling scheme has shown attractive performance compared to the output buffer flow-control process.

The hardware features of the proposed router have been checked for ASIC circuit prototyping with 45 nm integration technologies. The obtained results show that our approach outperforms many similar solutions for on-chip communication while ensuring attractive end-to-end QoS.

For future work, we think that the design of a dynamic flow-control mechanism which takes into consideration the constraints of the specific data stream that overloads the network could be an interesting approach to extend the scalability of the proposed router. In addition, we believe that designing the router with an asynchronous internal architecture would definitely improve the power consumption of the proposed architecture, which might make it more attractive for usage in low-power applications.

References

Aci, C.I., Akay, M.F., 2010. A new congestion control algorithm for improving the performance of a broadcast-based multiprocessor architecture. J. Parallel Distrib. Comput. 70, 930–940.

Adel, Soudani, Aldammas, Ahmed, Al-Dhelaan, Abdullah, 2014. Efficient scheme for congestion control in network on chip with QoS consideration. J. Circuits Syst. Comput. 23, 09.

Ascia, G., Catania, V., Palesi, M., Patti, D., 2006. A new selection policy for adaptive routing in network on chip. In: Proceedings of the 5th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications, pp. 94–99.

Barbera, M., Lombardo, A., Schembra, G., Trecarichi, A., 2008. Improving fairness in a WRED-based DiffServ network: a fluid-flow approach. Perform. Eval. 65, 759–783.

Becker, D.U., Jiang, N., Michelogiannakis, G., Dally, W.J., 2012. Adaptive backpressure: efficient buffer management for on-chip networks. In: IEEE 30th International Conference on Computer Design (ICCD), pp. 419–426.

Bolotin, E., Cidon, I., Ginosar, R., Kolodny, A., 2004. QNoC: QoS architecture and design process for network on chip. J. Syst. Architect. 50, 105–128.

Chen, Y.-J., Yang, C.-L., Chang, Y.-S., 2009. An architectural co-synthesis algorithm for energy-aware network-on-chip design. J. Syst. Architect. 55, 299–309.

Daneshtalab, M., Ebrahimi, M., Liljeberg, P., Plosila, J., Tenhunen, H., 2012. A systematic reordering mechanism for on-chip networks using efficient congestion-aware method. J. Syst. Architect.

David, L., Sood, M., Kajla, M.K., 2011. Router based approach to mitigate DOS attacks on the wireless networks. In: Proceedings of the 2011 International Conference on Communication, Computing & Security, pp. 569–572.

Ebrahimi, M., Daneshtalab, M., Liljeberg, P., Plosila, J., Tenhunen, H., 2012. CATRA-congestion aware trapezoid-based routing algorithm for on-chip networks. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 320–325.

Jafri, S.A.R., Hong, Y.-J., Thottethodi, M., Vijaykumar, T., 2010. Adaptive flow control for robust performance and energy. In: Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 433–444.

Jovanovic, S., Tanougast, C., Weber, S., Bobda, C., 2007. CuNoC: a scalable dynamic NoC for dynamically reconfigurable FPGAs. In: International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 753–756.

Kaddachi, M.L., Soudani, A., Tourki, R., 2008. Signaling approach for NOC quality of service requirements. In: 2nd International Conference on Signals, Circuits and Systems, SCS 2008, pp. 1–5.

Kumar, A., Mahapatra, R.N., 2005. An integrated scheduling and buffer management scheme for input queued switches with finite buffer space. Comput. Commun. 29, 42–51.

Lee, S.E., Bagherzadeh, N., 2009. A variable frequency link for a power-aware network-on-chip (NoC). Integr. VLSI J. 42, 479–485.

Lee, J.W., Ng, M.C., Asanović, K., 2012. Globally synchronized frames for guaranteed quality-of-service in on-chip networks. J. Parallel Distrib. Comput. 72, 1401–1411.

Lotfi-Kamran, P., Rahmani, A.-M., Daneshtalab, M., Afzali-Kusha, A., Navabi, Z., 2010. EDXY–A low cost congestion-aware routing algorithm for network-on-chips. J. Syst. Architect. 56, 256–264.

Michelogiannakis, G., Dally, W.J., 2009. Router designs for elastic buffer on-chip networks. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pp. 1–10.

Mishra, A.K., Yanamandra, A., Das, R., Eachempati, S., Iyer, R., Vijaykrishnan, N., Das, C.R., 2011. RAFT: a router architecture with frequency tuning for on-chip networks. J. Parallel Distrib. Comput. 71, 625–640.

Ni, L.M., McKinley, P.K., 1993. A survey of wormhole routing techniques in direct networks. Computer 26, 62–76.

Palesi, M., Kumar, S., Holsmark, R., 2006. A method for router table compression for application specific routing in mesh topology NoC architectures. In: Embedded Computer Systems: Architectures, Modeling, and Simulation. Springer, pp. 373–384.

Peh, L.-S., Dally, W.J., 2000. Flit-reservation flow control. In: Proceedings Sixth International Symposium on High-Performance Computer Architecture, HPCA-6, pp. 73–84.

Rijpkema, E., Goossens, K., Rădulescu, A., Dielissen, J., van Meerbergen, J., Wielage, P., Waterlander, E., 2003. Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip. IEE Proc. Comput. Digital Tech. 150, 294–302.

Samman, F.A., Hollstein, T., Glesner, M., 2009. Networks-on-chip based on dynamic wormhole packet identity mapping management. VLSI Des. 2009, 2.

van den Brand, J.W., Ciordas, C., Goossens, K., Basten, T., 2007. Congestion-controlled best-effort communication for networks-on-chip. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 948–953.

Wang, Chifeng, Bagherzadeh, Nader, 2014. Design and evaluation of a high throughput QoS-aware and congestion-aware router architecture for network-on-chip. Microprocess. Microsyst. 38, 304–315.

Wang, C., Hu, W.-H., Bagherzadeh, N., 2012. Scalable load balancing congestion-aware network-on-chip router architecture. J. Comput. Syst. Sci.

Wang, J., Gu, H., Yang, Y., Wang, K., 2013. An energy-and buffer-aware fully adaptive routing algorithm for network-on-chip. Microelectron. J. 44, 137–144.

Wiklund D., Liu, D., 2003. SoCBUS: switched network on chip for hard real time embedded systems. In: Proceedings International on Parallel and Distributed Processing Symposium, 8 pp.