Distributed Flit-Buffer Flow Control for Networks-on-Chip

Nicola Concer, Michele Petracca, and Luca P. Carloni *
Dept. of Computer Science, Columbia University, New York, NY 10027
{concer,petracca,luca}@cs.columbia.edu

ABSTRACT

The combination of flit-buffer flow control methods and latency-insensitive protocols is an effective solution for networks-on-chip (NoC). Since they both rely on backpressure, the two techniques are easy to combine while offering complementary advantages: low complexity of router design and the ability to cope with long communication channels via automatic wire pipelining. We study various alternative implementations of this idea by considering the combination of three different types of flit-buffer flow control methods and two different classes of channel repeaters (based respectively on flip-flops and relay stations). We characterize the area and performance of the two most promising alternative implementations for NoCs by completing the RTL design and logic synthesis of the repeaters and routers for different channel parallelisms. Finally, we derive high-level abstractions of our circuit designs and we use them to perform system-level simulations under various scenarios for two distinct NoC topologies and various applications. Based on our comparative analysis and experimental results, we propose a NoC design approach that combines the reduction of the router queues to a minimum size with the distribution of flit buffering onto the channels. This approach provides precious flexibility during the physical design phase for many NoCs, particularly in those systems-on-chip that must be designed to meet a tight constraint on the target clock frequency.

Categories and Subject Descriptors

B.4.3 [**Hardware**]: Input/Output and Data Communications Interconnections (subsystems).

General Terms

Design, Performance.

Keywords

Network-on-chip, latency-insensitive protocols.

1. INTRODUCTION

Networks-on-chip (NoC) have been proposed as a new paradigm for both chip-multiprocessors (CMP) and systems-on-chip (SoC) [1,8,10]. In the case of SoCs for embedded applications designers use standard industrial CAD-tool flows for the synthesis of a platform-specific NoC and must cope with an increasing number of timing-closure exceptions due the differences in size across its heterogeneous processing cores. This problem becomes particularly hard when using nanometer technology processes [20] as the impact of global interconnect wires raises exponentially the number of *wire exceptions*, i.e. timing-closure violations due to the delay of a global wire exceeding the target clock period T_{clk} [4, 11].

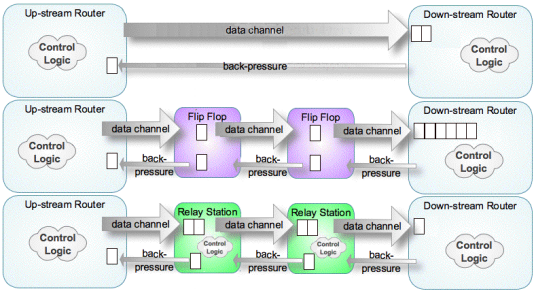


Figure 1: Alternative ways to pipeline NoC channels.

A method to fix wire exceptions is *wire pipelining*, i.e. the insertion of sequential elements (or clocked buffers) to pipeline long wires in shorter segments whose delays meet T_{clk} [3, 6, 14, 15, 21]. By providing one or more extra clock periods to traverse long distances, wire pipelining trade-offs latency for throughput. As proposed by Jalabert *et al.* [13], the use of latency-insensitive protocols [3] in NoC design allows channels to be pipelined to an arbitrary degree, thus decoupling T_{clk} from the worst-case channel delay.

Latency-insensitive protocols are implemented using *relay stations*, clocked repeaters of unit latency and twofold storage capacity. Relay stations can be used instead of regular flip-flops to enable arbitrary wire pipelining between two routers in a NoC (Fig. 1). Further, when combined with flit-buffer flow control methods [9], relay stations can store flits in the presence of persistent congestion because they actively process the flow-control signals. Hence, their use effectively increases the total storage capacity of the channel, thereby opening the way for interesting design optimizations.

We study in detail the interaction between wire pipelining and NoC flow-control methods and we propose *distributed flit-buffer flow control* as a technique that combines the simplest form of ack/nack protocol with the distribution of relay stations on the NoC channels for both buffering and wire pipeline purposes. We show how this approach provides NoC designers with both better options to optimize performance/area trade-offs and precious flexibility to complete efficiently the NoC physical design stage.

Related Work. In [19] Pullini *et al.* studied the interaction between wire pipelining and flow-control focusing on providing fault-tolerant communication on the NoC channels, a goal that is outside the scope of this paper. In [12] Hu *et al.* proposed an algorithm for optimal buffer sizing in packet-switched or virtual-cut-through NoCs. In [17] Ogras *et al.* proposed a technique to improve the performance of a Mesh NoC by incrementally inserting additional long pipelined channels. Methods to optimally size queues in on-chip global communication channels are presented in [5, 16].

*N. Concer is also with the Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy. M. Petracca is also with the Dipartimento di Elettronica, Politecnico di Torino, Italy.

CODES+ISSS'08, October 19–24, 2008, Atlanta, Georgia, USA.

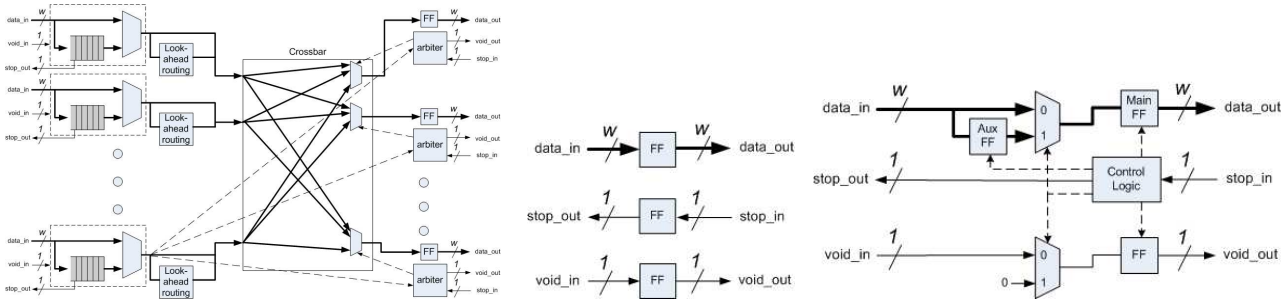


Figure 2: NoC components: (a) router; (b) FF-repeater; (c) RS-repeater.

2. BASIC NOC COMPONENTS

We summarize here the main characteristics of the three basic NoC components used in the rest of the paper.

The **router** is the key component of a packet-switched NoC. Fig. 2(a) shows the basic structure of a router implementing a XY-routing algorithm supported by wormhole flow-control. Solid lines show the data plane while dashed lines show the control plane. A crossbar switch separates the input from the output part. Each input port is equipped with a *look-ahead routing* module and a *bypassable queue* of size Q and parallelism W (*flit width*). Each output port has W *output registers* to store the forwarded flit and an *arbiter* to allocate the port among competing input worms. With look-ahead routing each router pre-computes the output port for the next downstream router: the information is carried in the worm head-flit, which can now be forwarded directly to the output port (if available). This leads to better performance by reducing the router critical path and allowing the routing task to be executed in parallel to arbitration. Without congestion a flit traverses the router in one clock cycle. In case of congestion, the flits of a worm that loses the arbitration are temporarily stored in the queue. When the queue gets filled, *backpressure* is triggered according to the given low-level flow-control mechanism.

An **FF-repeater** is the simplest type of channel repeater (Fig. 2(b)). It consists of a number of flip-flops (FF) equal to the flit width W plus two FFs: one for the *void* signal distinguishing valid flits from void ones and one for the *stop* signal carrying backpressure information backward on the channel. At each clock cycle a FF-repeater samples a new flit and makes it available on the output ports *without* processing the void/stop signals. Thus, each flit spends *exactly* one cycle on each FF-repeater without the possibility of being stored. Hence, in the case of a persistent congestion the flits of a worm end up being stored in the router queues while the channel FF-repeaters are empty. The overall channel latency is equal to the number of channel repeaters K.

An **RS-repeater** is a more complex repeater based on the relay station (RS) circuit that was first proposed for latency-insensitive design [3]. Relay stations are used implement a latency-insensitive protocol, but when used in NoC design enable also a distributed implementation of flow control. Fig. 2(c) shows the structure of a relay station: it consist of a battery of W *main FFs* in parallel with W *auxiliary FFs* plus one FF for the void signal, one additional FF that is used both to sample the stop signal and to implement the two-state finite-state machine governing the flow-control mechanism. At each clock cycle a RS-repeater samples a new flit into its main flip-flops to make it available on its output port. However, if it samples also the asserted *stop_in* value then it goes in a stalling state to: (a) keep the present flit on the main FF (to make it available again on the output port in the next cycle), and, (b) sample any newly-arrived valid flit in the auxiliary FF while asserting *stop_out* so that the upstream node will be stalled too. Hence, in the case of a persistent congestion a channel of K RS-repeaters contains $2 \cdot K$ flits.

In Sec. 4 we study in detail the effective ratio between the area of an NoC based on RS-repeater and an equivalent NoC based on FF-repeater as a function of the flit width W. Meanwhile, as a *rule of thumb* we assume that this ratio is equal to two.

3. WIRE PIPELINING & FLOW CONTROL

Flow-control methods can be classified based on their granularity of channel bandwidth allocation and of buffer allocation [9]. The basic unit of bandwidth and storage allocation is a *flit (flow control digit)*. Packets are divided in sequences of flits. Differently from packets, flits carry no routing and sequencing information. Flit-buffer flow control allocates both bandwidth and buffers in units of flits. This has three advantages: it (a) reduces the storage required for correct operation of a router, (b) provides stiffer backpressure from a point of congestion back to the source of a flit stream, and (c) enables more efficient use of storage. Since these advantages match well the characteristics of on-chip communication, flit-buffer flow control methods are seen as a promising solution for NoC, where typically the size of a flit matches the parallelism of a channel.

The two main high-level flow-control methods are *wormhole flow control* and *virtual-channel flow control*. These need to be supported by one of three main low-level flow control mechanisms that provides buffer management and backpressure, namely: *on/off, credit-based* and *ack/nack* [9]. In our analysis we focus on the combination of wormhole flow control with each of these low-level mechanisms.

Besides allocating the NoC bandwidth and storage resources, flow-control methods should provide good performance by guaranteeing a high bandwidth for the transmission of a stream of flits in the presence of possible *intervals of stalling cycles* caused by congestion in the downstream nodes. The choice of the flow-control strategy has consequences on the design of the network components and, particularly, on the size of the flit-buffering queues in the routers. If we want to avoid dropping flits, the correct operation of a particular flow-control method sets a constraint on the minimum size Q_{min} of Q. Once this constraint is met, raising the value of Q leads generally to better performance.

Rep. type	On/Off		Credit Based		Ack/Nack	
	Q_{min}	S	Q_{min}	S	Q_{min}	S
FF	$2+4K$	$2+5K$	$2+2K$	$2+3K$	$1+2K$	$1+3K$
RS	2	$2+2K$	2	$2+2K$	1	$1+2K$

Table 1: Queue size and channel total storage per flow-control/repeater type.

This, however, varies depending on the network traffic as discussed in Sec. 5. Also, in practice, raising it beyond a certain value leads to diminishing returns.

On/Off is a simple flow-control mechanism that minimizes the amount of backpressure signaling in exchange for larger queue size. The upstream node has a single-bit state register that switches between *on* and *off* states based on the last backpressure signal received from the downstream node. The latter sends an *off* signal back whenever the number of free slots in its flit-buffering queue goes below a threshold F_{off} and sends an *on* signal whenever it goes above a threshold F_{on}. Hence, the minimum size Q_{min} depends on the number of flits that can be received during the time T_{rt} from the instant when an *off* signal leaves the downstream node until the instant when the downstream node has received the last flit transmitted by the upstream node before stalling due to the processing and reception of signal *off*. In a synchronous NoC, if the latency of the channel is K clock cycles, then $T_{rt} = (2 + 2 \cdot K) \cdot T_{clk}$. Hence, for protocol correctness, $Q_{min} = 2 + 2 \cdot K$. However, to optimize the bandwidth we must consider also the dual case when the downstream node sends an *on* signal upstream to resume transmission. In this case at least T_{rt} cycles must pass by before a new flit arrives downstream. Meanwhile, in order to have sufficient flits to forward to the next hop, the downstream queue must be able to contain as many additional flits for a total size $Q_{min} = 2 + 4 \cdot K$.

Credit Based is a flow control mechanism where the upstream node has a counter to track the number of available free slots in the downstream queue. The counter state is decremented whenever a flit is transmitted and incremented whenever a credit signal arrives from the downstream node, which in turns sends the credit whenever it has succeeded in forwarding a flit from its queue to the next hop. With respect to on/off, credit-based flow control requires more "backpressure signalling", but smaller queues. Specifically, for protocol correctness, a $Q_{min} = 1$ is enough. However, such small size leads to the insertion of void flits (*bubbles*) at every hop because only one credit is available on each channel at any given time. Hence, only one flit can be forwarded per each round-trip of this credit and the higher is the value of $K > 1$ the lower the performance. To avoid bubble insertion the queue must be sized based on the round-trip latency. $T_{crt} = (2 + 2 \cdot K) \cdot T_{clk}$, which leads to $Q_{min} = 2 \cdot (1 + K)$.

Ack/Nack does not require any state in the upstream node to indicate buffer availability in the downstream node. Instead flits are optimistically sent whenever they become available: if the downstream node has a slot available in the queue it accepts the flit by sending an *ack* signal, otherwise it drops it and sends a *nack* signal. Ack/nack flow control mechanism is traditionally considered inefficient both in terms of storage (it requires that each transmitted flit be held waiting for an acknowledgement) and bandwidth (due to the potential retransmissions) [9]. Further, since it is based on acknowledging the reception of each specific flit, it works well for a channel of unit latency. But, if the channel contains K FF-repeaters it becomes suboptimal with respect to credit-based, where an acknowledgment denotes the successful forwarding of a generic flit. Still, ack/nack was effectively used to implement fault-tolerant *Go-Back-N* protocols [19] with routers having *output queues* of size $Q_{min} = 1 + (2 \cdot K)$.

What are the best combinations? The first row of Table 1 summarizes the requirements on the queue size for the three flow control methods as well as the corresponding values for the *channel total storage* S. The value of S is obtained by adding the queue size and the amount of storage provided by the channel repeaters, which is independent from the flow control method. In the case of a channel containing K FF-repeaters, each repeater provides storage for one flit, thus resulting in K flit buffers distributed on the channel. For FF-repeaters, the credit-based flow control method is the best choice in terms of sustainable bandwidth per unit of storage.

The second row of Table 1 shows the corresponding numbers for the case when the repeaters are implemented as relay stations. Since a RS-repeater can store up to two flits, the distributed storage on a channel of K RS-repeaters is equal to $2 \cdot K$. But, the fact that a relay station contains the logic implementing the low-level flow control mechanism makes it possible to reduce the size of the downstream queue to a minimum value that is *always* as if $K = 0$. In particular, ($Q_{min} = 1$) is sufficient when combining RS-repeaters with the ack/nack flow control mechanism. Ack/nack signalling naturally matches the *stop_in*/*stop_out* signalling proposed for latency-insensitive protocols [3] and, indeed, we will show that it is the best design choice when using RS-repeaters.

In summary, *independently from the chosen flow-control mechanism, for any value of* $K > 0$ *the value of the channel total storage* S *that is needed for a correct behavior when using RS-repeaters is always smaller than the value needed when using FF-repeaters.*

This result, which is reached with an analytical model based on the rule of thumb that the RS-repeater area is twice the flip-flop area, is validated by our experiments with the semicustom design of many channel subsystems for various flit widths (Sec. 4). While queue sizes larger than Q_{min} generally benefit the network performance, the optimal size depends on the network topology and application traffic. Still, as shown by the system-level experiments of Sec. 5, the ability of working with a lower Q_{min} gives an important advantage to a NoC that employs RS-repeaters instead of FF-repeaters.

4. AREA OCCUPATION ANALYSIS

We completed VHDL parameterized designs for the NoC components presented in Sec. 2 and synthesized many versions of them with a $90nm$ industrial standard-cell library.

Table 2(a) reports the area occupation of a RS-repeater versus a FF-repeater as function of the flit width W varying from 16 to 512 bits. The target clock frequency was set equal to $2Ghz$ and met by all repeaters under all configurations. Each output port was loaded with a wire capacitance that was previously characterized by considering an optimally-buffered wire implemented in an intermediate metal level.

Generally the higher is the flit width the lower is the ratio of the RS-repeater area over the FF-repeater area. The ratio goes from 3.19 (for $W = 16$) to 1.79 (for $W = 512$). This is not surprising since the additional overhead due to the flow-control logic becomes less important with respect to

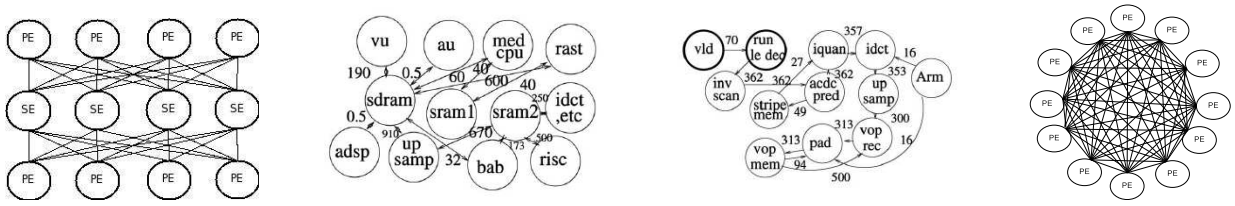
(a)

W	FF	RS	RS/FF
16	735	2348	3.19
32	1390	4100	2.95
64	2699	6514	2.41
128	5318	13903	2.61
256	10557	21316	2.02
512	21034	37599	1.79

(b)

W	$K=1$			$K=2$			$K=3$		
	RS	FF	ratio	RS	FF	ratio	RS	FF	ratio
16	29k	34k	0.84	38k	43k	0.88	48k	54k	0.89
32	42k	52k	0.81	58k	66k	0.89	75k	84k	0.89
64	66k	88k	0.75	92k	116k	0.80	118k	159k	0.74
128	123k	151k	0.81	178k	217k	0.82	234k	296k	0.79
256	203k	284k	0.72	288k	408k	0.71	374k	531k	0.70
512	363k	545k	0.67	513k	788k	0.65	664k	1006k	0.66

Table 2: Area $[um^2]$ as function of W: (a) FF-repeater vs. RS-repeater and (b) FF-system vs. RS-system.



(a) (b) (c) (d)

Figure 3: Application task graphs: (a) 4-Rooted Tree Forest (4RTF), (b) MPEG4 decoder, (c) VOPD decoder, and (d) random uniform traffic (URT).

the fact that a RS-repeater has twice the number of FFs as an equivalent FF-repeater [1]. Still, based on these results one may think that the rule of thumb of considering this ratio equal to two is justified only for channel with large width. Before drawing this conclusion, let's consider what happens when the repeaters are instanced as part of a long channel in a NoC.

We instanced a 5×5-port version of the router of Fig. 2 and we connected four of its five output ports to as many long repeated channels (while we assume that the fifth port is used for the local connection). This subsystem corresponds exactly to one "tile" of a 2D-Mesh NoC and, therefore, its area is a good estimate of the overall NoC area. Indeed, we considered two subsystems:
RS-subsystem: a router plus 4 channels pipelined using RSs.
FF-subsystem: a router plus 4 channels pipelined using FFs.

However, for both subsystems we used the same router implementing ack/nack flow control. This is advantageous for the FF-subsystem since a credit-based router would have a larger area than an equivalent ack/nack router because it needs an additional counter at each output to store status information on the outstanding credits.

Table 2(b) reports the area occupation of a RS-subsystem versus a FF-subsystem as function of the flit width W, which varies from 16 to 512, and the number K of repeaters on each channel, which varies from 1 to 3. The input queues of the router are set to the minimum value Q_{min}, i.e. 1 for the RS-subsystem and $2 + 2 \cdot K$ for the FF-subsystem. As the value of W grows, the area ratio approaches the theoretical limit of $\frac{2}{3}$, which is obtained from the analytical model by dividing the corresponding values of S from Fig. 1. In conclusion, *under every condition the RS-subsystem is always significantly smaller than the FF-subsystem.*

[1]The fact that for very-high values of W, i.e. 512 bits, the ratio goes below two can be explained as follows: both RS- and FF-repeaters are equipped with an array of output buffers to drive the wire load capacitance. The auxiliary FFs inside the RS-repeater, instead, do not need them because they are directly connected to the local multiplexers. Therefore, for high-values of W the area of the main FFs and the buffer dominates the area of a RS-repeater and it is comparable to the area of an equivalent FF-repeater.

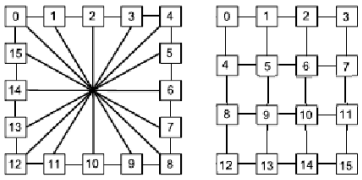


Figure 4: NoC topology examples: a 16-node Spidergon and 4×4 2D Mesh.

5. SYSTEM-LEVEL SIMULATIONS

For our system-level simulations we considered two NoC topologies (Fig. 4): a 2D Mesh, which broadly represents a class of NoCs that have been proposed for various general-purpose chip multiprocessors and SPIDERGON, an NoC architecture aimed at SoC for embedded applications [7]. SPIDERGON is a bidirectional ring with an even number of nodes enriched by "across" bidirectional channels between opposite nodes.

We built detailed models from the parameterized NoC components (routers, FF-repeaters, RS-repeaters) in the OMNET++ event-driven network simulator [18] and we combined them with high-level abstractions of the processing elements (PE) and memory elements (ME) that are on the chip. Each node in both the 2D Mesh and SPIDERGON contains either a PE or an ME attached to the local port of the router via a network interface that performs the operations of fragmenting packets into flits (and vice versa). The 2D Mesh uses 5×5 router implementing the well-known XY-routing algorithm [9], while SPIDERGON uses 4×4 routers implementing a discrete minimal routing algorithm that forwards the incoming flits along the across channels if their destination is "closer" to the opposite half of the ring, and sends them along the ring otherwise. Wormhole flow control is used in both NoCs. We simulated the two NoCs with four different traffic patterns taken from the literature (Fig. 3):

1. the *4-Rooted Tree Forest (4RTF)* models a scenario where 4 MEs are uniformly shared as communication targets by 8 PEs: Each PE initiates a communication by sending either a load or a store request to a given ME that replies

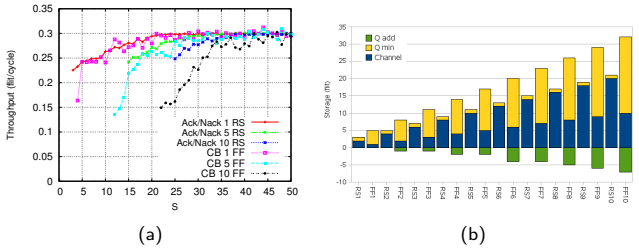


Figure 5: Results with Spidergon supporting the 4RTF traffic: (a) average bandwidth and (b) breakdown of channel total storage.

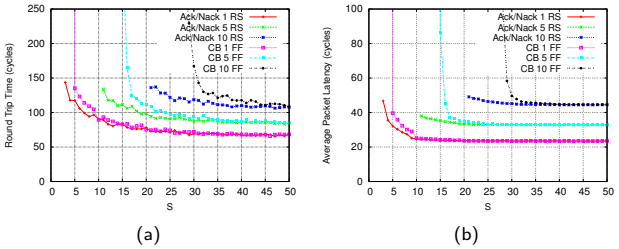


Figure 6: RS-repeaters with ack/nack vs FF-repeaters with credit based: (a) round-trip time for Spidergon with 4RTF traffic and (b) average packet latency for 2D Mesh with URT.

with either data or an acknowledgement. This is a typical scenario in many embedded applications where a shared memory bank becomes a central hot spot of the NoC.

2. a central memory hot spot is present also in the *MPEG decoder* SoC where various PEs exchange data by means of three memories (SDRAM, SRAM1 and SRAM2) [2].

3. PEs in the *Video Object Plane Decoder (VOPD)* SoC, instead, exchange data via point-to-point communication [2]. In this case, as in the following, a communication initiated by a PE is not followed by a reply from the target node.

4. in the *uniform random traffic (URT)* case, each node is a PE that communicates with every other PE in the system.

We use the following performance metrics:

• *packet latency*: time taken by a packet to enter the network, traverse it, and reach the destination;

• *round-trip time*: time elapsed from the transmission of a request packet and the reception of the corresponding replay packet;

• *bandwidth*: number of flits reaching a node per time unit.

Bandwidth Analysis. We report experimental results only for the 12-node Spidergon NoC supporting the 4RFT traffic pattern because the results for the other topology/traffic combinations are similar. Fig. 5(a) shows the average bandwidth as function of the channel total storage S. Each PE has a fixed injection rate that is higher than what the NoC can sustain if the router queue sizes are kept at the minimum value Q_{min}. From this graph it is clear that the amount of storage available on a channel significantly influences the system performance: as the storage increases, more flits can be stored in the routers' queues reducing the channel contentions (with wormhole switching single packets are stored along multiple routers). Hence, the saturation threshold is raised and the performance of the NoC improved.

The bar diagram of Fig. 5(b) reports the breakdown of the channel total storage that is required to obtain maximum bandwidth in a non-saturated NoC as function of the number K of channel repeaters. In particular, the sequence of points on the x-axis corresponds to 20 different design scenarios for K that varies from 1 to 10. For each value of K there are two bars: one corresponding to the RS-repeaters and one corresponding to the FF-repeaters. In each design scenario K repeaters are uniformly distributed on each NoC channel. Each bar includes up to three components:

• the blue (dark) part is the amount of storage provided by the repeaters, i.e. K for FF-repeaters and to $2 \cdot K$ for RS-repeaters;

• the yellow (light) part is the size Q_{min} of the router's input queues that is necessary to correctly support the given flow control, as explained in Sec. 3. This is always 1 for a RS-system and it is equal to $2 + (2 \cdot K)$ for a FF-system;

• the green (grey) part is the additional amount of storage Q_{add} that queues must have to reach the maximum bandwidth. Notice how for FF-systems under the analyzed traffic scenario, Q_{add} is lower than zero indicating that the maximum bandwidth can be reached with less storage than the one provided to satisfy the Q_{min} optimization constraint. In other words, the network can tolerate the insertion of a certain amount of bubbles per hop. Since bubbles increase the worm length, the worm can lock more channels during the time it passes through the NoC. Still, when the storage is high enough, the impact of those bubbles on the channel occupation is reduced.

In all the scenarios the RS-system reaches the maximum bandwidth using an amount of storage smaller than the corresponding FF-system, with an improvement that goes from 40% in case of small values of K down to 15% for K equal to ten.

Latency Analysis. The charts in Fig. 6 compare the RS-system and the FF-system with respect to the round-trip delay for Spidergon with 4RTF traffic and average packet latency for 2D Mesh with URT as we vary the channel total storage S. Again, we report data only for two pairs of topology/traffic since the other combinations show similar trends. As a theoretical exercise we let vary S up to 50 flit slots, but the curves reach a *minimal latency* value much earlier than that. The RS-system performs better than the corresponding FF-system for a given S and requires a smaller value of S to meet a given maximum latency constraint. In Fig. 6(a), for instance, when $K = 5$ the RS-system does not exceed a latency of 17 cycles using 35% less storage than the FF-system, while for a fixed $S = 17$ it delivers 12% less latency. Further, for the case FF-repeaters if the queues have size $Q < Q_{min}$ then the credit-based flow control creates many bubbles that increase dramatically the average NoC latency.

Next, we measured the channel total storage S_{delay} that is required to stay within 10% of the above-mentioned minimal latency. This value depends on the specific NoC, the application task graph, the PEs' injection rate, and the number of repeaters K. The bar diagrams in Fig. 7 report S_{delay} for all design combinations. Generally, higher values of S are needed than for bandwidth optimization (Fig. 5(b)). Again, RS-systems shows better performance than corresponding FF-systems, e.g. requiring up to 30% less storage in the case of Spidergon/4RTF. Reaching the minimal latency with a lower storage amount indicates that the given resources are better exploited. Indeed, this is the case for the RS-system where the storage deployed on the NoC can be used also to buffer the flits traversing the channels.

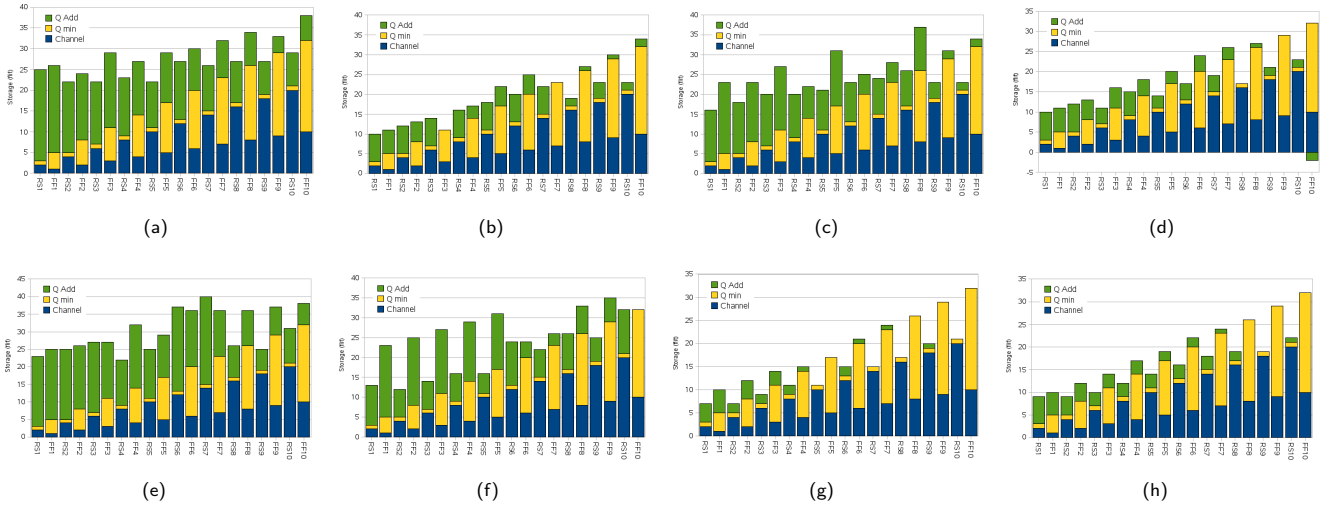


Figure 7: RS-repeaters with ack/nack vs FF-repeaters with credit based: breakdown of channel total storage required to obtain the minimal latency in a non-saturated NoC for: Spidergon with (a) 4RTF, (b) MPEG4, (c) VOPD, (d) URT, and 2D Mesh with (e) 4RTF, (f) MPEG4, (g) VOPD, (h) URT.

6. CONCLUSIONS

Given the particular constraints imposed by nanometer technologies, we propose *distributed flit-buffer flow control* for NoC design as a method that combines the simplest form of ack/nack protocol with the distribution of relay stations on the channels. Relay stations act both as clocked repeaters to pipeline the channels and as flit buffers to enable a distributed implementation of flow control. Consequently, they provide precious flexibility during the physical design of the NoC by allowing designers to use smaller routers and to manage long wires better. Experimental results, including semicustom implementations and system-level simulations, show that across many scenarios:

- for an equivalent amount of storage capacity a RS-based NoC performs better than a FF-based NoC;
- a RS-based NoC needs less storage capacity to deliver the same performance as a FF-based NoC.

Future work includes the study of the proposed approach in combination with virtual-channel flow control.

Acknowledgments. The authors thank Marcello Coppola and Riccardo Locatelli of STMicroelectronics for useful discussions. This work was supported in part by the National Science Foundation (Award #: 0541278), STMicroelectronics Inc, and the GSRC focus center, one of the five research centers funded under the FCRP, a Semiconductor Research Corporation program.

7. REFERENCES

[1] L. Benini and G. D. Micheli. Networks on chip: A new SoC paradigm. *IEEE Computer*, 49(2/3):70–71, Jan. 2002.

[2] D. Bertozzi et al. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. on Parallel and Distributed Systems*, 16(2):113–129, Feb. 2005.

[3] L. P. Carloni et al. A methodology for "correct-by-construction" latency insensitive design. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 309–315, 1999.

[4] L. P. Carloni and A. L. Sangiovanni-Vincentelli. Coping with latency in SoC design. *IEEE Micro*, 22(5):24–35, Sept./Oct. 2002.

[5] V. Chandra, A. Xu, H. Schmit, and L. Pileggi. An interconnect channel design methodology for high performance integrated circuits. In *Conf. on Design, Automation and Test in Europe*, pages 21138–21143, 2004.

[6] P. Cocchini. Concurrent flip-flop and repeater insertion for high-performance integrated circuits. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 268–273, 2002.

[7] M. Coppola et al. Spidergon: A NoC modeling paradigm. In *Proc. 2004 International Symposium on System-on-Chip*, page 15, Nov. 2004.

[8] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. of the Design Automation Conference*, pages 684–689, June 2001.

[9] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, San Mateo, CA, 2004.

[10] A. Hemani et al. Network on chip: An architecture for billion transistor era. In *18th IEEE NorChip Conference*, Nov. 2000.

[11] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr. 2001.

[12] J. Hu, U. Ogras, and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Trans. on Computers*, 25(12):2919–2933, Dec. 2006.

[13] A. Jalabert, L. Benini, S. Murali, and G. D. Micheli. ×*pipesCompiler*: a tool for instantiating application-specific NoCs. In *Conf. on Design, Automation and Test in Europe*, Feb. 2004.

[14] X. Liu, Y. Peng, and M. C. Papaefthymiou. Practical repeater insertion for low power: what repeater library do we need? In *Proc. of the Design Automation Conference*, pages 30–35, 2004.

[15] R. Lu et al. Flip-flop and repeater insertion for early interconnect planning. In *Conf. on Design, Automation and Test in Europe*, 2002.

[16] R. Lu and C. Koh. Performance optimization of latency insensitive systems through buffer queue sizing of communication channels. In *Proc. Intl. Conf. on Computer-Aided Design*, page 227, 2003.

[17] U. Y. Ogras and R. Marculescu. It's a small world after all: Noc performance optimization via long-range link insertion. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 14(7):693–706, July 2006.

[18] OMNeT++ discrete event simulation system. available online at http://www.omnetpp.org/.

[19] A. Pullini, F. Angiolini, D. Bertozzi, and L. Benini. Fault tolerance overhead in network-on-chip flow control schemes. In *Proceedings of SBCCI*, pages 224–229, 2005.

[20] A. Pullini et al. Bringing NoCs to 65*nm*. *IEEE Micro*, 27(5):75–85, 2007.

[21] L. Scheffer. Methodologies and tools for pipelined on-chip interconnect. In *Proc. Intl. Conf. on Computer Design*, pages 152–157, Oct. 2002.