Buffer Sizing for Multimedia Flows in Packet-Switching NoCs

Leonel P. Tedesco, Ney Calazans, and Fernando Moraes

Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS)
Av. Ipiranga, 6681 – P. 32 – 90619-900 – Porto Alegre – BRAZIL
e-mail: {ltedesco, calazans, moraes}@inf.pucrs.br

ABSTRACT

Wormhole packet switching, used in many NoC designs, introduces jitter. This may produce violations of application deadlines. Several works in the literature propose stream workload models, and techniques for buffer sizing. These works do not consider the concurrency between different flows, or the NoC model is too abstract, masking the jitter introduced by data packaging and router processing. One technique to deal with jitter is to introduce a decoupling buffer (D-buffer) on the target IP. This buffer receives data from the NoC with jitter, while the target IP consumes data from this buffer at the application rate, without jitter. Two problems must be solved to implement D-buffers: (*i*) which size must the buffer have? (*ii*) how much time should be expected before data consumption starts (threshold)? This work proposes a general method to define D-buffer size and threshold, considering the influence of packaging, arbitration, routing and concurrency between flows. A traffic model for stream applications is detailed and used to characterize jitter sources in wormhole packet switching. Experimental results demonstrate the impact on multimedia flows with fixed and variable packet sizes (from real traffic traces). Simple traffic models employing constant frame sizes result in small D-buffers. On the other hand, employing multimedia frames from application traces (i.e. real application data) increases buffer size and threshold while still suppressing jitter. Another parameter analyzed in the paper is the percentage of deadline violations as a function of D-buffer size. The method guarantees throughput with no deadline violation and does not modify the NoC structure. The cost of adding such D-buffers is an increased latency and extra silicon area.

Index Terms: Buffer Sizing, Networks on Chip, Quality of Service, Traffic Modeling, Performance Evaluation.

1. INTRODUCTION

Networks on Chip [1] (NoCs) have been investigated in the last few years as a promising alternative to bus-based and point-to-point architectures to interconnect Intellectual Property cores (IPs) in Systems on Chip (SoCs). NoCs allow simultaneous transactions between several pairs of cores (parallelism) and the connection of new IPs does not imply redesigning the communication infrastructure and cause no significant performance reduction on the overall system (scalability).

The typical workload of complex SoCs includes control signals, large memory block transfers and multimedia streams. Temporal requirements (deadlines to send/receive data) and data volume to be transmitted characterize applications. However, the transmission of a given flow through a wormhole packet switching NoC may modify the original flow rate, resulting in missed deadlines at the target IP. This is called the *load fluctuation phenomenon*. Three processes introduce load fluctuation: *data packaging*, *router processing*, and *concurrency between flows*. Variations in latency caused by load fluctuation are called *jitter*.

Figure 1 illustrates a typical example of data arriving at a target IP with load fluctuation. In the Figure, IP0 generates data to IP8, in a 3x3 mesh NoC. Figure 1(b)-A shows a 2-frame stream being generated at a given rate. Figure 1(b)-B represents data injection into the network after packaging (adding header information to each packet), the first source of load fluctuation. Then, this flow traverses the NoC, and at each hop routing and arbitration is executed, the second source of load fluctuation. This flow arrives at IP8, Figure 1(b)-C, with different delays between flits. If this flow is blocked by another flow, the third load fluctuation factor, concurrency between flows, arises.

One technique to minimize or even suppress jitter is to introduce a decoupling buffer (*D-buffer*) on the target IP, which receives data from the NoC, with jitter. The target IP consumes data from this buffer, at the application rate, without jitter. Two problems must be solved to implement the D-buffer: (*i*) which *size* must the buffer have?; (*ii*) how much time should be expected before data consumption starts (*threshold*)? Buffer size and threshold must be correctly dimensioned to avoid: (*i*) *data loss*, which happens if the NoC cannot write into the D-buffer (*buffer full*); (*ii*) *starvation*, which happens if the IP has no data to read from the D-buffer (*buffer empty*); (*iii*) *latency increasing*, which happens if data consumption by the target IP starts too late.

This paper has two objectives. The first one is to characterize sources of jitter introduced by data packaging, arbitration/routing and concurrency with other flows. The second one is to propose a buffer sizing method to restore the temporal characteristics of QoS flows.

This work is organized as follows. Section 2 discusses related work in buffer sizing for NoCs. Section 3 proposes a new method of traffic modeling. Section 4 discusses jitter sources. Section 5 describes the proposed D-buffer sizing method, the main contribution of this paper. Section 6 presents experimental results and Section 7 concludes the paper suggesting some directions for future work.

2. RELATED WORK

This Section reviews works concerning buffer sizing and its relation with QoS parameters. Different methods have been proposed for traffic generation and buffer management.

Hu et al. [2] presented a method for buffer sizing on intermediate nodes of NoCs, using queuing theory formalisms. The main objective is to minimize the average latency of all communications that occur on the NoC, with reduced buffer area occupation. The Authors considered data storage using packets as atomic unit, i.e. store-and-forward switching mode. The store-and-forward technique is not frequently used in NoCs, since it increases latency and area, because NoC buffers size must be at least the size of the maximum size packet. In terms of traffic modeling, the Authors employ a Poisson synthetic model to characterize telecommunication networks. The disadvantage of this model is reduced accuracy compared to trace-based models or even self-similar models.

Varatkar and Marculescu [3] demonstrated the self-similar characteristic of MPEG traffic, one typical application on current SoCs. The Authors showed that it is possible to define the optimal size for buffers of MPEG decoders modules to avoid buffer overflow. In terms of traffic modeling, a method for synthetic traffic generation is presented where traces of traffic and their statistical properties are combined on a synthetic trace generation procedure. However, experimental traffic scenarios considered only point-to-point communication, discarding the possible influence of concurrent flows. In addition, the Authors give no data about the buffer threshold value.

Chandra et al. [4] presented a method to size buffers considering data production and consumption rates of packets transmitted in burst. Throughput is the performance metric employed to compare *atomic* and *distributed* buffers. Atomic buffers are those at target IPs, while distributed buffers are those found on intermediate nodes. Results presented in [4] show a higher throughput for the distributed buffer strategy. The main weakness of this method is that the NoC is optimized for a fixed traffic scenario, which is inadequate to SoCs that accept applications defined after design.

Manolache et al. [5] propose a method based on traffic heuristics for buffer space optimization on size and occupation, considering latency constraints and packet drop estimations. In this method, communication events are mapped on the network and/or packets are delayed at the source, in order to avoid

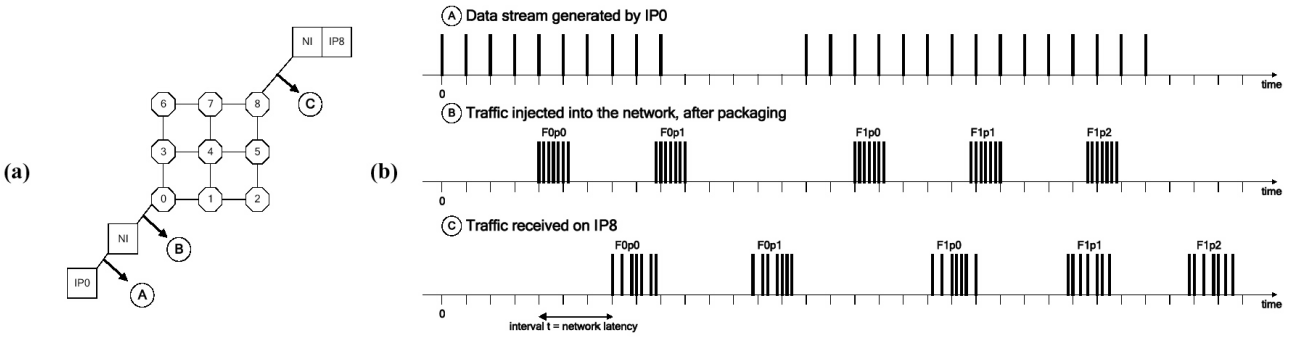

Figure 1. Load fluctuation phenomena in NoC-based SoCs. (a) traffic source IP0 sends data to the traffic target IP8; (b) illustration of jitter arising at different points on the path between source and target. NI stands for Network Interface.

concurrence between different flows. The Authors use a set of synthetic applications in the experiments to validate the method. The obtained results show the effect of traffic shaping to reduce the total amount of required buffer space on intermediate routers. Another result is an increase in the number of applications that can be implemented on a specific buffer space. This work, however, does not account for throughput guarantees.

Nicopoulos et al. [6] propose a unified buffer structure that dynamically allocates virtual channels and buffer resources according to traffic conditions. The Authors apply packet injection using fixed and variable (self-similar) rates in the experiments. They also vary the spatial traffic distribution. This work presents results emphasizing latency reduction and throughput increase. However, the method does not account for guarantees on throughput and latency values on the target routers.

Coenen et al. [7] present an algorithm to size buffers at the target IP in a NoC using virtual channels and credit-based flow control. The objective is to guarantee a constant consumption rate on target IPs without data loss. The Authors consider the periodicity properties of both data production and consumption from buffers using two arrays which store information expressing data arrival times and the required data processing rate of a target IP. However, they do not consider concurrency between different flows, which probably turns the target traffic aperiodic in time.

Table I presents a summary of related works on buffer dimensioning. These works present basically three methods for buffer sizing. Most of these [2] [3] and [5] execute buffer sizing on intermediate NoC routers. The main objective is the overall latency reduction. Nicopoulos et al. [6] propose a unified buffer structure, to reduce latency. Coenen et al. [7] propose a method to size buffers at the target NI (Network Interface), with the objective of guaranteeing application constant throughput. In [4], experiments evaluate both internal and external buffer dimensioning.

In terms of traffic modeling, some works employ synthetic traffic [2][4][6], while others use trace-based workloads [3][5][7]. Methods for strict QoS requirements are implemented in [5] and [7] for latency- and throughput-sensitive applications, respectively.

A significant part of the reviewed buffer sizing works does not consider concurrency between distinct flows. Besides, when abstract NoC models (e.g. SystemC descriptions) are employed, the jitter induced by packaging and arbitration/routing is not taken into account in the sizing methods. This work proposes a general method for traffic modeling and for defining the size and threshold value for *D-buffers*, considering the influence of packaging, arbitration, routing and concurrency between flows.

3. TRAFFIC MODELING

Traffic modeling is the phase on the NoC design cycle where communication requirements are captured [8]. Multimedia applications, such as audio and video streams, represent the typical workload of present SoCs. Works like [3] and [9] describe traffic modeling algorithms for video streams. Streams are composed by frames, generated at constant intervals, characterizing an ON-OFF traffic model. The size of each ON period may vary, as induced for example by the use of compression algorithms. The rate of the ON period is usually constant.

Table I. Characteristics of related work in buffer sizing.

Author	**Hu et al. [2] 2006**	**Varatkar and Marculescu [3] 2004**	**Chandra et al [4] 2004**	**Manolache et al. [5] 2006**	**Nicopoulos et al. [6] 2006**	**Coenen et al. [7] 2006**
Goal	Minimize average latency, with reduced buffer area occupation	Optimize buffer size for MPEG modules, under self-similar traffic	Algorithm to reduce search space for optimal buffer sizing	Buffer space optimization in size and occupation, based on traffic heuristics.	Dynamic virtual channels and buffer resources allocation, according to traffic conditions	Buffer sizing on target NI buffers, to guarantee a constant consumption by the application
Traffic model	Synthetic	Trace-based	Synthetic	Trace-based	Synthetic	Trace-based
Internal or external buffer dimensioning?	Internal	Internal	Both	Internal	Unified structure	External
Metric for performance evaluation	Latency	Latency	Throughput	Latency	Latency and throughput	Throughput
Concurrent traffic effects considered?	Yes	No	No	Yes	Yes	No

Due to the higher transmission rate of NoC channels, compared to individual IP rates, it is recommended to encapsulate application data into packets, in a process called *packaging*. Figure 2 details a packaging process. Figure 2(a) presents the stream under generation by the source IP, started by two frames with different sizes. Each 4μs a new frame starts. Assume frames are composed by 16-bit words, with flit size equal to the word size. In the example, *frame0* contains 10 flits and *frame1* contains 15 flits. During each ON period, the IP generates data at a constant rate. In the Figure, the data generation rate for *frame0* and *frame1* is 160 Mbps (10 16-bit flits transmitted in 1μs, for *frame0* and 15 16-bit flits transmitted in 1.5μs, for *frame1*). Figure 2(b) is a detailed view of *frame0*. A flit is produced every 100 ns. For a 50 MHz clock (20 ns period), both flit generation and flit consumption should occur every 5 clock cycles. Figure 2(c) shows *frame0* segmented in two packets, with header information included. These packets are injected into the network and stored in a *D-buffer* on the target IP.

The traffic generation for the ON-OFF traffic model uses the following parameters:

- ***Number of frames***, a fixed parameter.
- ***Frame_start_time***, the timestamp of the first payload flit injected into the NoC (440 ns in Figure 2).
- ***Inter_frames_arrival***, a fixed parameter, corresponding to the distance between two consecutive frames.
- ***Frame_size***, which corresponds to the size of the ON period.
- ***Rate of the ON period***, a fixed parameter. In Figure 2 the rate of the ON period is 160 Mbps.
- ***Number of packets per frame or packet size***. If the number of packets per frame is fixed, the packet size varies according to the rate and the ON period size. If the packet size is fixed, the number of packets per frame varies, as illustrated in Figure 2.

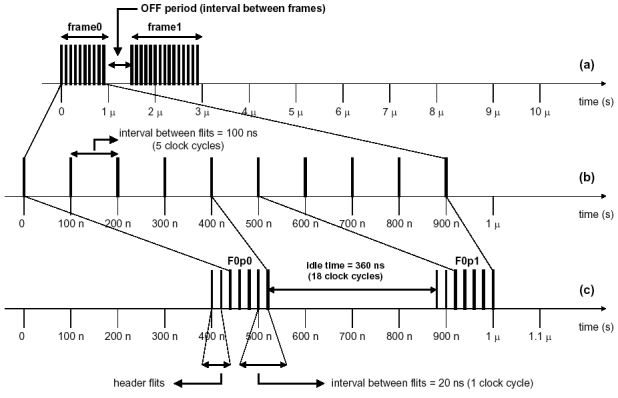

Figure 2. Data packaging process exemplified using a video stream.

```
1:frame_ts <- frame_start_time;
2:for i=0 to nframes-1 do    // Number of frames
3:  pkt_ts <- frame_ts;
4:  if packet_size_fixed==true then
5:    npacks <- framesize(i)/pcksize;
6:    for j=0 to npacks-1 do
7:      pkt_ts <- prv_tmp+prv_pcksize*ncyclesflit+idle;
8:      generate(pkt_ts,pcksize)
9:      prv_tmp     <- pkt_ts;
10:     prv_pcksize <- pcksize;
11:   end for
12: else if packets_per_frame_fixed==true then
13:   pcksize <- framesize(i)/pckspframe
14:   for j=0 to pckspframe-1 do
15:     pkt_ts <- prv_tmp+prv_pcksize*ncyclesflit+idle;
16:     generate(pkt_ts,pcksize)
17:     prv_tmp     <- pkt_ts;
18:     prv_pcksize <- pcksize;
19:   end for
20: end if
21: frame_ts <- frame_ts + INTER_FRAMES_ARRIVAL;
22:end for
```

Figure 3. Traffic generation algorithm for the ON-OFF traffic model for video frames.

The algorithm in Figure 3 describes the traffic generation process for the ON-OFF traffic model, considering the packaging process (Figure 2 (c)).

Line 1 initializes the timestamp to inject the first frame. The timestamp is the ideal moment to transmit a packet or a frame, and its value is based in a global counter, that registers the number of clock cycles spent since the beginning of a simulation. The next line starts the frame transmission loop. Lines 4 to 11 execute when the packet size is a fixed parameter. Line 5 computes the number of packets for the current frame, according to the execution of the *framesize* function (it may employ fixed frame sizes, probabilistic functions, or traces from real applications). The loop between lines 6 to 11 injects the packet into a FIFO (function *generate*), and computes the timestamp of the next packet (line 7 – detailed in the next paragraph). The generate function inserts the new packet, with timestamp *pkt_ts* and size *pcksize*, in a FIFO. This FIFO is used only for traffic generation. Another process reads this FIFO, injecting the packet into the NoC. Note that if the network is congested, the injection timestamp may be different from *pkt_ts*, increasing latency and jitter. Lines 12 to 20 execute when the number of packets per frames is fixed. Line 21 computes the timestamp for the next frame, according to the *inter_frames_arrival* parameter. The timestamp of the packet, computed in lines 7 and 15, considers the timestamp of the previous packet (*prv_tmp*), the previous packet size (*prv_pcksize*), the number of clock cycles to transmit one flit (*ncyclesflit*) and the idle time between packets. The idle time follows the equation below [10].

$$idle = \left(\frac{chr}{ipr} - 1 \right) * pcksize * ncyclesflit$$

In this equation, *chr* is the channel transmission rate and *ipr* is the IP transmission rate, both measured in Mbps.

In the previous example, *chr*=800 Mbps, *ipr*=160 Mbps, *pcksize*=5 and *ncyclesflit*=1, resulting in an idle time between packets equal to 20 clock cycles. However, Figure 2 shows an idle time equal to 18 clock cycles, due to the influence of the header flits. The injection process reads the timestamp of the next packet, subtracting from it the number of clock cycles necessary to insert the header information (in the example, 2 clock cycles).

4. JITTER SOURCES

As stated before, three processes introduce jitter in wormhole packet switching NoCs: *data packaging, router processing, and concurrency between flows.* In contrast, circuit switching does not introduce jitter, and provides latency and throughput guarantees. The reason NoCs adopt wormhole packet switching despite the difficulties to ensure temporal constraints for QoS flows is that wormhole requires no static resource reservation, enabling to optimize NoC resources (e. g. to share the same physical channel between two distinct flows).

Section 3 presented the packaging process, exemplifying it with a video frame in Figure 2. Incoming flits are stored in a FIFO buffer at the source NI. When the number of stored flits is equal to the packet size, these flits are injected into the NoC, in burst. *Header flits* occupy the beginning of the packet, containing for example the target router address and the payload size. The *idle time* between packets must be respected, to keep the original application data rate. Data packaging optimizes channel bandwidth use, due to burst transmission. As illustrated in Figure 2, the addition of the two header flits in the beginning of the packet reduces the idle time between packets from 20 to 18 clock cycles, introducing jitter *before* injecting packets into the NoC. The effect of header insertion in the jitter depends on the packet size. It is important to mention that the dimensioning of the FIFO buffer at the source NI is not a subject approached in this work. The subject of this work is to suppress the jitter at the target IP by means of the *D-buffer*.

This paper assumes the use of distributed routing schemes. In these schemes, a packet injected into the network traverses different routers to reach the target IP. At each intermediate router, a process to select which input port must be served first (arbitration) and the selection of the output port (routing) is executed (only for the header flits). Considering that this process takes some clock cycles to execute, some flits are blocked in the meantime, changing the original packet rate, and consequently increasing jitter. Router designs include input or output buffers to reduce congestion during arbitration and routing, not to reduce jitter.

The third process that introduces jitter is the concurrency between different flows sharing the same NoC resources. When two or more flows are assigned to use the same router output port, one flow transmits first, while the other(s) must wait the end of this transmission. This also modifies the original rate of blocked packets, introducing jitter. A technique used to minimize this problem is the adoption of virtual channels, a time division multiplexing method. Virtual channels transmit flits from different flows in an interleaved manner. This technique improves NoC performance, reducing the total latency and the jitter.

Jitter introduces several problems for applications: (*i*) received data may not respect temporal requirements, such as the interval between bursts; (*ii*) packet latency increases; (*iii*) the target IP may lose data. Even though the available bandwidth in NoC channels is usually high (around Gbps [11]), it is necessary to recover the original rate produced by the source IP respecting temporal deadlines. This can be achieved using decoupling buffers, the *D-buffers*, attached to the NI of the target IP.

5. BUFFER SIZING

This Section presents the method to compute the size and the threshold values for the *D-buffers*. The *D-buffers* are part of the target IP network interface, as illustrated in Figure 4. The correct definition of the buffer size avoids data loss, and the correct definition of the threshold value avoids starvation. *Data loss* happens when the NoC cannot write into the *D-buffer* (buffer full). *Starvation* happens when the IP has no data to read from the D-buffer (buffer empty). These two parameters are obtained using data collected after the NoC simulation, using the modeled traffic as workload. The simulation data is collected at the output port of each router.

A global counter is responsible to count the number of clock cycles, *globaltime*, since the beginning of NoC simulation (clock cycle accurate simulation, with the NoC modeled in synthesizable VHDL, and the test bench in SystemC). A new flit arrives at the output port of a given router when the

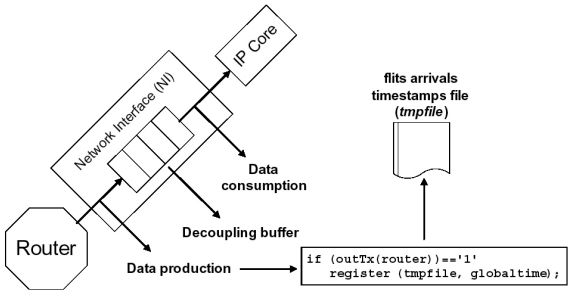

Figure 4. Data collection at the NI decoupling buffer.

signal *outTx* is asserted (the assertion of this signal corresponds to the flow control used in most NoCs: handshake or credit-based). For each new flit arrival, the current timestamp (value of the *globaltime* counter) is stored in the router timestamp file, (*tmpfile*, in Figure 4).

The proposed method fills two arrays. The first array, named *recv*, collects data generated after simulation at the output port of the target router (*data production*, in Figure 4). The second array, named *cons*, is filled according to the rate at which the IP should receive data. These arrays contains at each *index* position the value '1' or '0' according to the following rules:

- **recv[index] = '1',** if a flit is received at the output port at timestamp index, **'0'** otherwise;
- **cons[index] = '1',** if a flit must be consumed by the target IP at timestamp index, **'0'** otherwise.

The size of the *recv* and *cons* arrays is equal to the number of clock cycles corresponding to the inter frames arrival (IFA) parameter, which is the time between two frames (Figure 5). This parameter, together with *size_ON*, form the QoS parameters for the target application, and are obtained at design time, from the trace files which describe the traffic that will run on the NoC. Timestamp TR0 corresponds to the timestamp of the first flit received at the router output port.

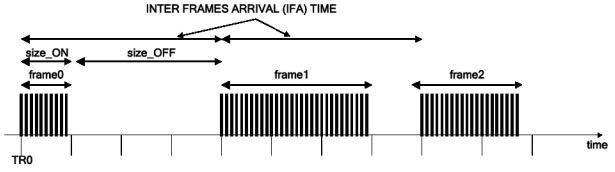

Figure 5. Parameters for buffer size and threshold definition.

```
1:  Tbegin_k ← TR0 + k*IFA
2:  Tend_k ← Tbegin_k + IFA - 1
3:  recv ← 0
4:  for all i ∈ tmpfile
5:    if i >= Tbegin_k  and i<= Tend_k  then
6:      index ← i - Tbegin_k
7:      recv[index] ← 1
8:    elsif i > Tend_k then
9:      exit
10:   end if
11: end for
```

Figure 6. Algorithm to fill the *recv* array, for the *kth* frame.

Figure 6 presents the algorithm to fill the *recv* array. $Tbegin_k$ and $Tend_k$ in lines 1 and 2 define the interval where flits belonging to the frame *k* should be received, according to the parameters IFA and TR0. Line 3 initializes the *recv* array, filling all positions with value '0'. Line 4 starts a loop seeking in the timestamp file, *tmpfile*, a value within the frame *k* (line 5). The difference between the flit arrival timestamp, value *i*, and the initial frame timestamp, $Tbegin_k$, (line 6) defines the *recv* index. The *recv* array at the index position receives '1' (line 7). The algorithm finishes when the value read in the *tmpfile* corresponds to a flit belonging to the next frame (line 9).

The consumption array, cons array, considers for each frame k the $size\_ON_k$ parameter (in clock cycles) and the consumption rate. The consumption rate is normalized to the channel bandwidth. Figure 8 presents the algorithm to fill the *cons* array. Line 1 initializes the cons array, filling all positions with '0'. Line 2 starts a loop, storing '1' in the *cons* array according to the consumption rate.

Figure 7(a) illustrates the execution of the algorithm to fill the *recv* array. The flits arrival time, stored in the *tmpfile*, are {130, 131, 132, 134, 146, 147, 149, 150, 166, 167, 168, 169}. Executing the algorithm presented in Figure 6, the *recv* array is filled according to Figure 7(a). Figure 7(b) illustrates the execution of the algorithm to fill the *cons* array. The defined *size_ON* is 28 clock cycles and the consumption rate 25% (0.25). Each 4 cycles (1/0.25) a flit should be consumed from the buffer. The algorithms are executed for all simulated frames.

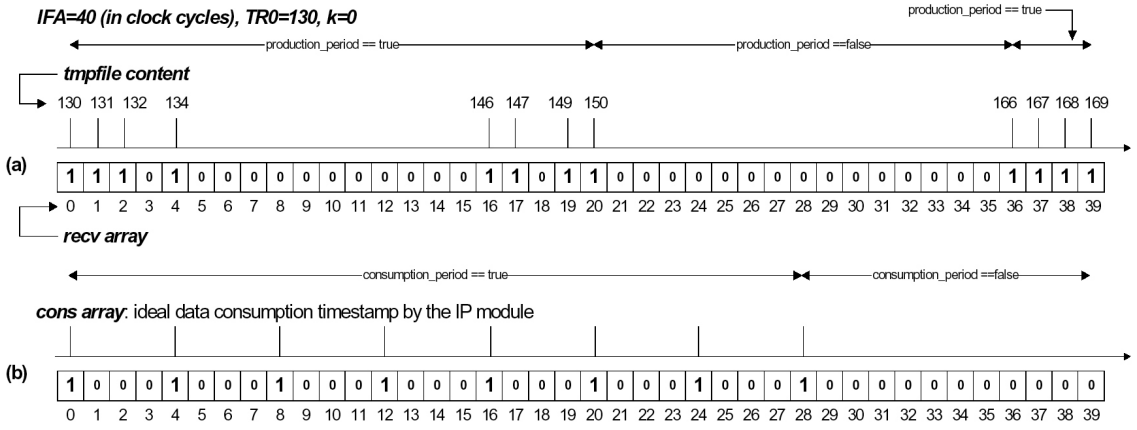

Figure 7. (a) *recv* (reception) and (b) *cons* (consumption) arrays.

```
1:  cons ← 0
2:  for all i ∈ SIZE_ON_k
3:    cons[i] ← 1 ∀ (i MOD 1/rate)=0
```

Figure 8. Algorithm to fill the *cons* array, for the *kth* frame.

To define the *D-buffer* size and threshold values two Boolean variables are considered: *production period* and *consumption period*. The production period is true in the interval comprising the reception of all flits belonging to the current frame, or during the reception of flits belonging to the next frame. The consumption period is true during the ON period of the IFA interval. In Figure 7(a) it is possible to observe that part of the *recv* array cells is filled with ones inside the consumption period. The last four flits arrived belong to the next frame, but appear in the present frame due to the load fluctuation produced by the concurrent traffic.

Table II illustrates the conditions that may arise according to the values of *recv* and *cons* array, and the values of production and consumption variables. The condition recv[i]='1' and cons[i]= '1' indicates a flit reception and consumption at time-slot *i*. In this case, no increase on D-buffer size and threshold is necessary.

The algorithm presented in Figure 9 computes the D-buffer size and threshold values. It computes the instantaneous difference between the two vectors, storing it in the *acc* variable. The *acc* variable is initialized with the number of flits received in the previous frame that belong to present frame (line 2). Another variable, *nbflits*, counts the number of received flits on the target NI, and it is initialized with zero at each new frame analysis. According to Table 2, line 6 of the algorithm expresses the condition to increase the buffer size and the threshold values. When this condition is true, the *acc* variable is updated (line 7), the same happening to the maximal (lines 8-10) and minimal (lines 11-13) *acc* values. Line 14 increments the number of flits belonging to the ON period. When the condition expressed in line 6 is false and a flit is received, the *first_flits* variable is incremented, to be used in the next frame (line 16). The D-buffer size corresponds to the addition of the acc maximum value (*higher_acc*) with its absolute minimal value (*lower_acc*) (line 20). The absolute minimum value of acc multiplied by the consumption rate indicates how much time must be expected before the beginning of flits consumption, i.e. the threshold value (line 21).

```
1:  for all k ∈ set of simulated frames
2:    acc ← first_flits
3:    first_flits ← 0
4:    nbflits ← 0
5:    for all i ∈ frame_k then
6:      if cons_period==true or
            (prod_period==true and nbflits<size(frame_k))
7:          acc ← acc + (recv[i]-cons[i])
8:          if acc > higher_acc  then
9:              higher_acc ← acc
10:         end if;
11:         if acc < lower_acc then
12:           lower_acc ← acc
13:         end if;
14:         nbflits++
15:     else if (recv[i]==1)
16:        first_flits=first_flits+1
17:     end if;
18:   end for;
19: end for;
20: Bsize ← higher_acc + |lower_acc|
21: Bthreshold ← |lower_acc|*1/rate;
```

Figure 9. Algorithm to define the size and threshold values for the D-buffer.

6. RESULTS

The HERMES [12] NoC, a wormhole packet-switched mesh topology NoC, was used to evaluate the buffer sizing technique proposed in this work. The

Table II. Conditions for buffer size and threshold definitions (*x* stands for don't care).

recv[i]	cons[i]	Production Period	Consumption period	Situation
0	1	any	false	Not applicable, the *cons* array is true only during the consumption period.
		any	true	The IP must consume a flit at timeslot *i*, but it was not received at this timeslot. This arises when flits are blocked inside the network. In this case, it is necessary to increment the amount of time to wait to start data consumption, i.e., *increment the threshold value* (e.g. index 28 in Figure 7).
1	0	true	true	A flit reception at timeslot i, before the moment to consume it. This condition frequently arises after a congestion situation. In this case, *increment the D-buffer size* (e.g. index 2 in Figure 7).
		true	false	If the received flit belongs to the current frame, but it is outside the consumption period, *both D-buffer size and threshold values must be incremented*. If the received flit belongs to the next frame, an extra buffer slot is required if the difference between the number of flits produced and consumed until this moment is smaller than the buffer size. The number of such flits must be considered for the D-buffer dimensioning in the next frame (e.g. index 36 to 39 in Figure 7).
		false	*x*	Not applicable, all '1' values stored in the *recv* array are within some production period equal to true.

fixed network design parameters are: 8x8 mesh; 16-bit flit size; 8-flit depth internal buffers; credit-based flow control; deterministic XY routing algorithm; 2 or 4 virtual channels associated to each physical channel. The employed NoC frequency is 50MHz, corresponding to an 800 Mbps channel rate.

The experiments employ three synthetic traffic models: control signals, HTTP and HDTV. Table III summarizes the characteristics of traffic models. Control signals are used as noise traffic, being transmitted using a complement spatial distribution. HTTP traffic is used to disturb the real time streaming traffic, HDTV. The HDTV traffic is characterized with temporal requirements, and *D-buffers* must be connected to the IPs receiving this traffic, to guarantee throughput with no deadline violation.

Control signal packets are modeled with small 15-flit packets, which represent a typical size for this kind of traffic. These packets are continuously generated in time, characterizing a CBR flow. The injection rate is 16 Mbps, corresponding to 2% network bandwidth consumption. HTTP traffic is modeled using Pareto ON-OFF injection process, where packets are generated during the ON period at 160 Mbps. The ON-OFF model describes the HDTV traffic. The ON period generates 10 packets at 200 Mbps, corresponding to 10 HDTV channels, consuming 25% of a NoC channel bandwidth. Each HDTV initiator transmits 20 frames. The simulated scenarios employ fixed-size and traced-based packets [13].

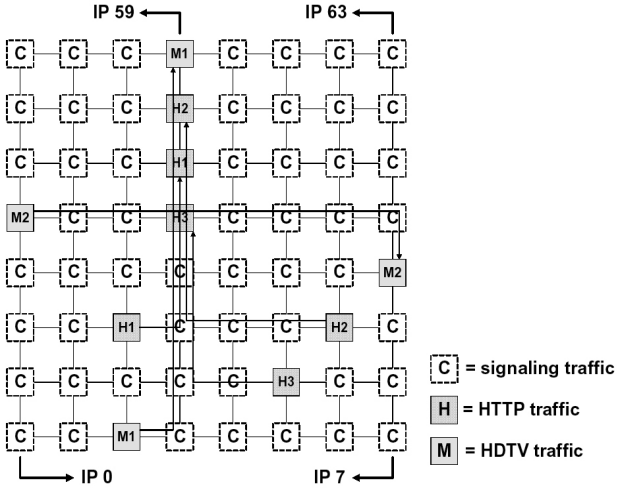

Figure 10. Spatial traffic distribution, with HDTV M1 being disturbed by three HTTP flows.

Figure 10 illustrates the spatial traffic distribution used in experiments, with two HDTV initiators (M1-M2), being the HDTV flow M1 disturbed by three HTTP flows. The complement distribution employed by the noise traffic increases the communication volume on the network bisection. HDTV and HTTP flows are placed next to the middle of the network, to maximize the noise traffic influence in these flows. The number of competing flows must be superior to the number of virtual channels to enable evaluating the impact of concurrence in latency and jitter.

A. Buffer sizing for fixed and variable packet sizes

This scenario models HDTV flows using the ON-OFF traffic model, for both fixed and variable packet sizes. The fixed packet size scenario employs a 1500-flit packet size. Traces collected from [13] are used to define the variable packets sizes. Traces have an average packet size of 1500 flits, varying from 15 to 6200 flits. Table 4 presents results related to the D-buffer threshold and size, as well as latency, considering packaging, concurrence between flows and the number of virtual channels.

1) Impact of packaging

When packaging is not adopted, there is no significant need for buffer slots, because there is almost no difference between the original data rate transmitted and the one arriving on the target IP. The adoption of packaging significantly reduces latency values and network contention, at the cost of increased *D-buffer* size (*Bsize*). The D-buffer size is proportional to the injection rate and the maximum packet size. The experiment adopts an injection rate of 25% of the channel bandwidth, meaning that one flit must be consumed each 4 clock cycles. Therefore, for each consumed flit, three flits remain stored in the *D-buffer*.

Table III. Traffic modeling for the conducted experiments (PS: packet size, IR: Injection Rate).

Application	Requirement	Traffic characteristics	
Control Signals	Signaling	PS	15 flits
		IR	Constant injection rate: 16 Mbps
100 HTTP sessions	Real-time block transfer	PS	600 to 1500 flits (fixed sized per experiment)
		IR	Pareto ON-OFF: 160 Mbps ON period
10 HDTV channels	Real time streaming	PS	Fixed (1500 flits) and trace-based (1500 flits on average)
		IR	Transmission of 10 simultaneous HDTV flows per frame. Rate during a frame period: 200 Mbps (25% load)

Table IV. Values of threshold and buffer size for IP 59.

Line	Traffic model	Packaging?	Concurrence?	Two virtual channels			Four virtual channels		
				Bthreshold (cycles)	Bsize (flits)	Latency (cycles)	Bthreshold (cycles)	Bsize (flits)	Latency (cycles)
1	Synthetic	No	No	0	9	22547	0	9	22550
2	(fixed	No	Yes	0	9	24794	0	10	22552
3	packet	Yes	No	0	1125	2300	0	1125	2300
4	sizes)	Yes	Yes	2664	1817	5059	0	1150	2407
5	Traces	No	No	0	9	21501	0	9	21502
6	(variable	No	Yes	20	704	24112	4	11	21301
7	packet	Yes	No	16048	4650	14193	16048	4650	14914
8	sizes)	Yes	Yes	14520	4654	14192	16048	4650	14915

Multiplying the maximum packet sizes (1500 for synthetic and 6200 for trace-based traffic) by 0.75 (percentage of idle time) the buffer size is 1125 and 4650 for synthetic and trace-based traffic respectively. These values can be observed in lines 3 and 7 of Table 4. For fixed packet sizes or when there is no concurrence between flows, the threshold value is zero. This means that flits can be consumed as soon as they arrive at the *D-buffer*.

When variable packets sizes are used together with packaging, the threshold values are greater than zero (line 7-8). Figure 11 shows packets arriving in the *D-buffer* and being consumed by the application, considering a threshold value equal to zero. The interval between packets is proportional to the packet size. This gap corresponds to a starvation condition, since the IP has no data to consume. The threshold value solves this problem, since it defines the amount of time to wait before data consumption starts. This value is high because a significant variation in packet length is present in the trace files.

2) Impact of concurrence

For the synthetic traffic model, concurrent traffic leads to a considerable increase in both threshold and buffer size (compare lines 3-4 of Table 4). Increasing the number of virtual channels, the effect of concurrence is minimized. For trace-based traffic, the impact of concurrence between flows is higher when no packaging is adopted (lines 5-6).

The concurrence between flows does not affect the dimensioning of the *D-buffer*, since the buffer size is large enough to store data within the threshold period (lines 7-8). Increasing the number of virtual channels does not change the *D-buffer* dimensioning, for the same reason: the buffer is sized according to the largest packet. The concurrence affects mainly the threshold value for both traffic models. The *D-buffer* size is greatly affected by the size of the largest packet. Latency is minimized employing packaging with fixed packet sizes, as in ATM networks.

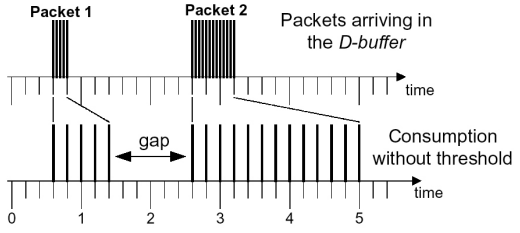

Figure 11. Data consumption without threshold, causing starvation.

3) Impact of routing/arbitration

The impact of routing and arbitration is very small compared to the other sources of jitter. This effect is caused by the larger size of HDTV packets (1500-6200 flits), compared to the amount of clock cycles spent for arbitration/routing (about 7).

This impact can be noted in lines 1-2 and 5-6, when no packaging is adopted. The obtained buffer size corresponds to the number of clock cycles to route/arbitrate packets in one router. Note in line 6 the reduction of the *D-buffer* size, when four virtual channels are adopted.

4) Buffer occupation

Figure 12 shows the number of filled buffer slots (*x* axis) versus the number of clock cycles these slots are filled during a simulation run (*y* axis), when

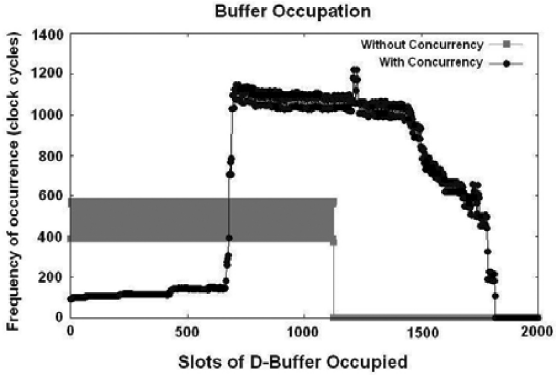

Figure 12. D-buffer occupation using HDTV with fixed 1500-flit packet sizes, without (gray) and with (black) concurrency of 1500-flit HTTP packets.

fixed packets size for HDTV traffic and two virtual channels are used. In the absence of concurrency, the buffer is homogeneously used in time. This behavior changes with concurrent traffic. When a given packet is blocked, it is possibly sent with a smaller idle time w. r. t. the next packet, in this way requiring buffer space to avoid deadline violations. Comparing the *D-buffer* sizes (3rd and 4th lines, 6th column of Table 4), the difference is 692, corresponding to the right shifting of the curve. These 692 extra buffer slots are required to absorb the jitter introduced by the concurrency between flows.

Figure 13 presents the buffer use for a flow having variable packet sizes. As shown in Figure 12, flows with fixed packet sizes use the buffer in a uniform way. Figure 13 shows a distinct behavior, since the buffer occupation is in average 70% during the application simulation. This is induced by larger packets, which must be buffered in order to allow the consumption at the application rate.

This first set of experiments demonstrated the effect of the concurrency between flows in the jitter, buffer size and buffer occupation, even when virtual channels are employed. The curves of Figure 12 and Figure 13 were obtained simulating the network with the *D-buffer*, after sizing it. The number of lost packets was zero, with a constant buffer consumption rate, validating the method to size the buffers.

B. Violated deadlines

The second set of simulation scenarios concern the percentage of violated deadlines due to buffer size reduction.

Figure 14 presents the latency and the percentage of flit deadline violation for fixed-size packets, considering packaging and 3 HTTP flows disturbing the QoS flow, as a function of the *D-buffer* dimension. If no *D-buffer* is used, a 68% flit deadline violation is observed, due to the jitter. A 40% reduction on buffer size and threshold produces an important latency decrease, for a small number of flit deadline violations (5%). This graph shows that: (*i*) if a small percentage of deadline violations is allowed, the buffer can be reduced, minimizing area and latency; (*ii*) the deadline violations strongly increases with the buffer size reduction.

Figure 15 presents an experiment similar to the one presented in Figure 14, using variable packet sizes. Despite the higher latency values, the latency decrease has the same behavior in both experiments. Comparing Figure 14 to Figure 15, the main difference observed is the flit deadline violation growth. When the *D-buffer* size is reduced in 60%, the fixed-sized packets traffic has 22% of deadline violations, while in the variable packet-size 10% of deadline violations arise.

7. CONCLUSIONS AND FUTURE WORK

This work presented a method to dimension decoupling buffers (*D-Buffers*), attached to the NoC output ports, guaranteeing throughput and no jitter to QoS flows. The advantage of adopting *D-buffers*

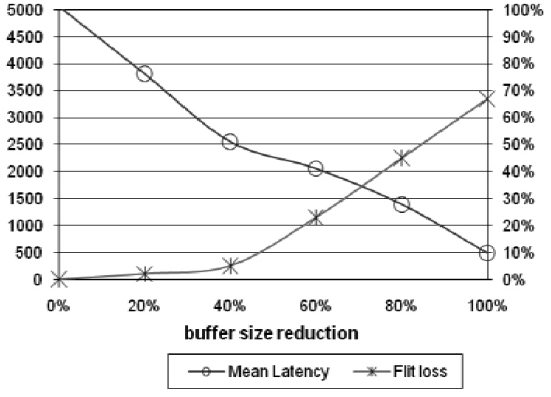

Figure 14. Latency and percentage of flit violated deadlines for fixed-size packets, disturbed by 3 HTTP flows (1500-flit packets).

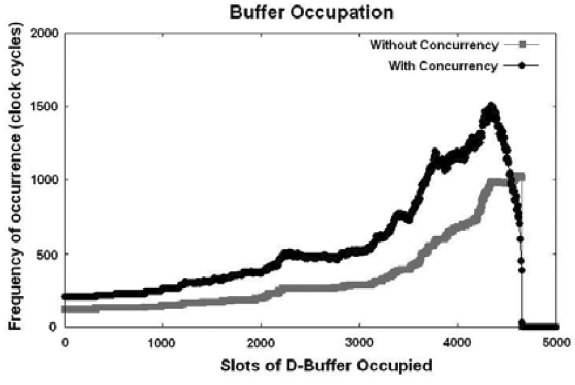

Figure 13. D-buffer using HDTV with variable packet sizes, without (gray) and with (black) concurrency of 1500-flit HTTP packets.

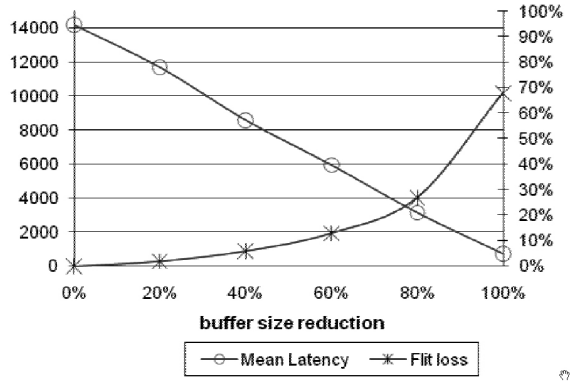

Figure 15. Latency and flit loss for trace-based frames HDTV traffic, disturbed by 3 HTTP flows (1500-flit packets).

instead of sizing the internal NoC buffers is reuse and flexibility, i. e. the same NoC may be used for different traffic scenarios without NoC re-synthesis.

Three sources of jitter were identified in this work. The *D-buffer* size is a function of the largest packet of the flow, assuming the use of packaging. The *D-buffer* threshold is a function of the concurrence between flows. The jitter introduced by the intermediate routers processing (arbitration and routing) may be neglected when compared to the other two jitter sources.

Despite the fact that *D-buffers* increase the latency, streaming applications require guaranteed throughput, not necessarily small latencies [11]. Consequently, latency increase is not a drawback for most multimedia applications. The latency may be minimized reducing the *D-buffer* size, if some deadline violation is allowed.

The main drawback of the proposed method is the simulation time, since an important number of frames should be used to correctly size the *D-buffers*. Therefore, future work includes the development of techniques to estimate the *recv* array contents, based on statistical properties of the incoming traffic.

ACKNOWLEDGEMENTS

This research was supported partially by CNPq (Brazilian Re-search Agency), project 300774/2006-0.

REFERENCES

[1] Benini, L.; De Micheli, G. "Networks on chips: a new SoC paradigm". *IEEE Computer*, 35-1, January 2002, pp. 70-78.

[2] Hu, J.; Ogras, U. Y.; Marculescu, R.. "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25-12, December 2006, pp. 2919-2933.

[3] Varatkar, G.; Marculescu, R. "On-Chip Traffic Modeling and Synthesis for MPEG-2 Video Applications". *IEEE Transactions on VLSI Systems*, 12-1, January 2004, pp. 108-119.

[4] Chandra, V.; Xu, A.; Schmit, H.; Pileggi, L. "An interconnect channel design methodology for high performance integrated circuits" in *Proceedings of the Design, Automation and Test in Europe, DATE'04*, 2004, pp. 1138-1143.

[5] Manolache, S.; Eles, P.; Peng, Z. "Buffer Space Optimization with Communication Synthesis and Traffic Shaping for NoCs" in *Proceedings of the Design, Automation and Test in Europe, DATE'06*, 2006, pp. 95-98.

[6] Nicopoulos, C. A.; Dongkook, P.; Jongman, K.; Vijaykrishnan, N.; Yousif, M. S.; Das, C. R. "ViChar: a Dynamic Virtual Channel Regulator for Network-on-Chip Routers" in *39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO'06*, 2006, pp. 333-346.

[7] Coenen, M.; Murali, S.; Radulescu, A.; Goossens, K.; De Micheli, G. "A Buffer-Sizing Algorithm for Networks on Chip using TDMA and Credit-Based End-to-End Flow Control" in *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'06*, 2006, pp. 130-135.

[8] Tedesco, L.; Mello, A.; Giacomet, L.; Calazans, N.; Moraes, F. "Application driven traffic modeling for NoCs" in *Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems Design, SBCCI´06*, 2006, pp. 62-67.

[9] Hu, J.; Marculescu, R. "Dyad – Smart routing for networks on chip" in *Proceedings of the 41st Design Automation Conference, DAC'04*, 2004, pp. 260-263.

[10] Tedesco, L.; Mello, A.; Garibotti, D.; Calazans, N.; Moraes, F. "Traffic Generation and Performance Evaluation for Mesh-based NoCs" in *Proceedings of the 18th Annual Symposium on Integrated Circuits and Systems Design, SBCCI'05*, 2005, pp. 184-189.

[11] Guerrier, P.; Greiner, A. "A Generic Architecture for On-Chip Packet-Switched Interconnections" in *Proceedings of the Design, Automation and Test in Europe, DATE'00*, 2000, pp. 250-256.

[12] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. "Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip". *Integration, the VLSI Journal*, 38-1, October 2004, pp. 69-93.

[13] Fitzek, S.; Reisslein, M. "MPEG-4 and H.263 Video Traces for Network Performance Evaluation". Available at *http://www-tkn.ee.tu-berlin.de/research/trace/trace.html*, 2008.