Buffer Sizing for QoS Flows in Wormhole Packet Switching NoCs

Leonel Tedesco
ltedesco@inf.pucrs.br

Ney Calazans
calazans@inf.pucrs.br

Fernando Moraes
moraes@inf.pucrs.br

Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS)
Av. Ipiranga, 6681 - 90619-900 - Porto Alegre – BRASIL

ABSTRACT

Networks on chip (NoCs) are communication infrastructures that offer parallelism and scalability. Most NoC designs employ wormhole packet switching, since this switching mode optimizes the use of NoC resources. However, this mode may introduce jitter, possibly producing packet loss, due to the violation of temporal QoS constraints. 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 buffer space should be filled 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. Before presenting the method, the paper extends a previous traffic model for stream applications and characterizes jitter sources in wormhole packet switching. The experimental results obtained with the proposed method showed that simple traffic models employing constant frame sizes result in small D-buffers. On the other hand, employing video frames from application traces (i.e. real application data) increases buffer size and threshold, still suppressing jitter. Application traces highlight the threshold parameter importance.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – advanced technologies, algorithms implemented in hardware, VLSI (very large scale integration).

General Terms

Design, Experimentation, Measurement, Performance, Theory, Verification.

Keywords

Buffer Sizing, Networks on Chip, Quality of Service, Traffic Modeling

SBCCI'07, September 3–6, 2007, Rio de Janeiro, Brazil.

1. INTRODUCTION

Some advantages offered by NoCs [1] are that simultaneous transactions may occur between distinct pairs of cores (parallelism) and that connection of new IPs does not imply redesigning the communication infrastructure and cause no significant performance reduction on the overall system (scalability).

Typical data flows occurring in complex SoCs include control signals exchange, large memory block transfers and multimedia streams. Temporal requirements (deadlines to send/receive data) and the data volume to transmit characterize applications. However, the transmission of a given flow through the NoC may modify the original flow rate, resulting in missed deadlines at the target IP. This flow modification is called the *load fluctuation* phenomenon. Three processes introduce load fluctuation: data packaging, router processing, and concurrency between flows. Variation in latency caused by load fluctuation is called *jitter*.

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. 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 buffer space should be filled 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 increase*, which happens if data consumption by the target IP starts too late (threshold value oversized).

This paper has two main 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 method for buffer sizing to restore the temporal characteristics of the flow.

This work is organized as follows. Section 2 presents related works in buffer sizing for NoCs. Section 3 presents traffic modeling used for streaming application, and the jitter sources. Section 4 describes the D-buffer sizing method, the main contribution of this work. Section 5 presents experimental results. Section 6 concludes the paper and suggests directions for future works.

2. RELATED WORK

Varatkar and Marculescu [2] demonstrate the self-similar characteristic of MPEG traffic, a typical application on current SoCs. The Authors show that it is possible to define the optimal size for buffers of MPEG decoder modules, in order to avoid buffer overflow. In terms of traffic modeling, they present a method for syn-

thetic traffic generation. In this method, traces of traffic and their statistical properties are combined on a synthetic trace generation procedure. However, experimental traffic scenarios consider only point-to-point communication, discarding an eventual influence of concurrent flows. In addition, there is no mention concerning the buffer threshold value in this work.

Hu and Marculescu in [3] present a method for buffer sizing on intermediate nodes of NoCs, using formalisms originated from queuing theory. The main objective is to minimize the average latency of all communications occurring on a NoC, while reducing buffer area occupation. The Authors consider data storing in terms of packets, i.e. store-and-forward switching mode. The store-and-forward mode is not frequently used in NoCs, since it increases latency and area, due to the needed buffer size. In terms of traffic modeling, the Authors employ a Poisson synthetic model to characterize telecommunication networks [4]. The disadvantage of this model is the lower accuracy, compared to trace-based models or even self-similar models.

Chandra et al. [5] present a buffer sizing method considering: (*i*) data production and consumption rates; (*ii*) size of packets transmitted in burst. The Authors compare throughput when atomic (at the target IP) and distributed (on intermediate nodes) buffers are used. They do not employ statistical properties to the modeled traffic, and the burst sizes are fixed for each simulation. The effect of network concurrency, and consequently jitter introduction, is not mentioned.

Coenen et al. [6] present an algorithm to size buffers at the target IP, in a NoC using virtual channels and credit-based flow control. The objective of this work is to guarantee a constant consumption rate on the target IPs, without data loss. The Authors consider the periodicity properties of both data production and consumption from buffers. They detect this periodicity using two arrays, storing information about data arrival times on the target IP, and the required data processing rate. Again, this work does not consider concurrency between different flows.

In short, the reviewed buffer sizing works do not consider concurrency between distinct flows. Besides, when abstract NoC models (e.g. System-C descriptions) are employed, jitter induced by packaging and arbitration/routing is not taken into account in the sizing methods. This work proposes a general method to define the size and threshold value for *D-buffers* considering the influence of packaging, arbitration, routing and concurrency between flows.

3. TRAFFIC MODELING AND JITTER SOURCES

Traffic modeling is the phase on the NoC design cycle where communication requirements are captured [7]. Multimedia applications, such as audio and video streams, represent the typical workload of present SoCs. Works like [2] and [8] 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. Still, the rate of the ON period is usually constant.

Figure 1 details a packaging process. Figure 1(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 30 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 30 16-bit flits transmitted in 3μs, for *frame1*). Figure 1(b) is a detailed view of *frame0*. A flit is produced every 100 ns. For a 50 MHz clock, both flit generation and consumption should occur every 5 clock cycles. Figure 1(c) shows *frame0* segmented in two packets, with header information included. These packets are injected into the network and stored at a D-buffer on the target IP.

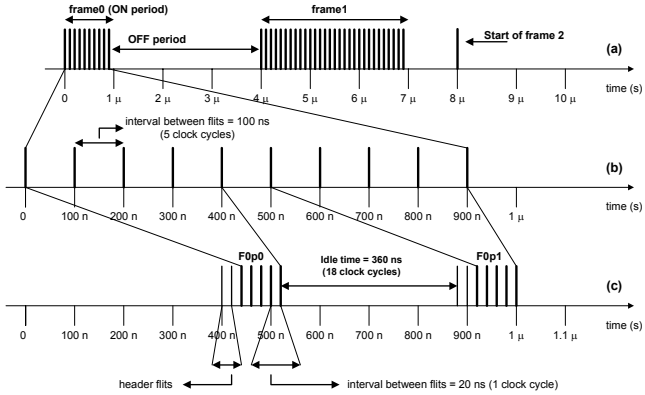

Figure 1 – Data packaging process exemplified using a video stream.

The traffic generation for the ON-OFF traffic model, deals with the following parameters [9]:

- ***Number of frames***, a fixed parameter.
- ***Frame_start_time***, the timestamp of the first payload flit injected into the NoC (440 ns in Figure 1).
- ***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 1 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 1.

3.1 Jitter Sources

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. share the same physical channel between two distinct flows).

Due to the higher transmission rate of NoC channels, compared to individual IP core rates, it is necessary to encapsulate application data into packets, in a process called *packaging*. Incoming flits are stored in a FIFO buffer at the network interface (NI). When the number of stored flits is equal to the packet size, these flits enter 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 each packet must be respected, to keep the original application data rate. Data packaging

optimizes channel bandwidth use, due to burst transmission. As illustrated in Figure 1, 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.

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 others 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. Event though the available bandwidth in NoC channels is usually high (around Gbps [10]), it is necessary to recover the original rate produced by the source IP respecting temporal deadlines. This can be achieved using decoupling buffers, *D-buffers*, attached to the NI of the target IP.

4. BUFFER SIZING

This Section proposes a 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 2. 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 (*data production* in Figure 2).

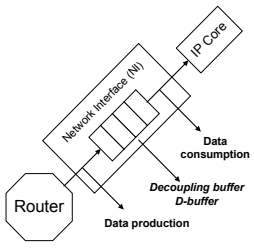

Figure 2 – NI decoupling buffer.

The proposed method starts by filling two arrays. The first array, named *recv*, collects data generated after simulation at the output port of the target router (*data production*, in Figure 2). The second array, named *cons*, is filled according to the rate 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 3). Timestamp TR0 corresponds to the timestamp of the first received flit at the router output port. All flits received within the IFA period should belong to the same frame. Flits belonging to a given frame received outside the IFA period are discarded, corresponding to a deadline violation. Figure 6 illustrates the parameters used to fill recv and cons arrays.

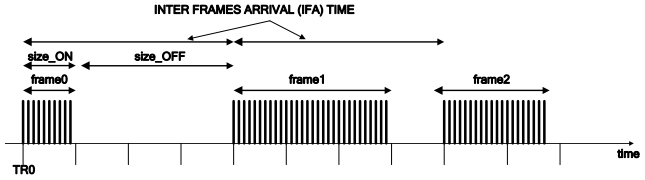

Figure 3 - Parameters for buffer size and threshold definition.

The global counter responsible to count the number of clock cycles, *globaltmp*, starts counting with the 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 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 *globaltmp* counter) is stored in the router timestamp file, named *tmpfile*.

Figure 4 presents the algorithm to fill the *recv* array. $Tbegin_k$ and $Tend_k$, 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).

```
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 4 - Algorithm to fill the *recv* array, for the *kth* frame.

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 5 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.

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

Figure 5 - Algorithm to fill the *cons* array, for the *kth* frame.

Figure 6(a) illustrates the execution of the algorithm to fill the recv array. The flits arrival time, stored in the *tmpfile*, are {130, 133, 134, 138, 141, 145, 146,150, 160}. Executing the algorithm presented in Figure 8, the recv array is filled according to Figure 6(a). Figure 6(b) illustrates the execution of the algorithm to fill the cons array. The defined size_ON is 16 clock cycles and the consumption rate 50% (0.5). Each 2 cycles (1/0.5) a flit should be consumed from the buffer. In the figure, it is possible to observe that the cons array cells filled with ones are those that are inside the size_ON period. The algorithms are executed for all simulated frames.

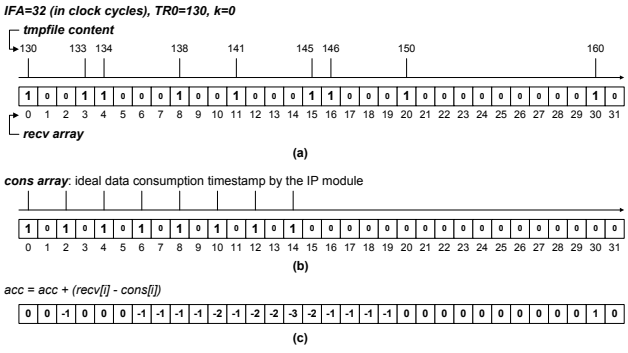

Figure 6 – (a) recv array filling; (b) cons array filling; (c) recv and cons comparison.

The *D-buffer* size and threshold parameters are computed according to the state of the *recv* and *cons* arrays. The following conditions may arrive:

- ***recv*[i]='1' ∧ *cons*[i]= '1'**: indicates a flit reception and consumption at timeslot *i.*
- ***recv*[i]='1' ∧ *cons*[i]= '0'**: indicates a flit reception at timeslot *i*, before the moment to consume it. This condition frequently arises in wormhole packet switching NoCs, since flits may arrive in burst after a congestion situation. In this case, a buffer slot is required to store the incoming flit.
- ***recv*[i]='0' ∧ *cons*[i]= '1'**: 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 threshold value, to avoid starvation.

The algorithm presented in Figure 7 defines the D-buffer size and threshold values. It computes the instantaneous difference, *acc* value, between the two vectors (line 3) for all timestamps, and the maximal (lines 4-6) and minimal (lines 7-9) values of *acc*. The D-buffer size corresponds to the addition of the *acc* maximum value with its absolute minimal value (line 12).The absolute minimum value of *acc* indicates how many flits must be stored in the D-buffer before start consumption, i.e. the threshold value (line 13).

```
1:  for all k ∈ set of simulated frames
2:    for all i ∈ frame_k then
3:      acc ← acc + (recv[i]-cons[i])
4:      if acc > higher_acc  then
5:        higher_acc ← acc
6:      end if;
7:      if acc < lower_acc then
8:        lower_acc ← acc
9:      end if;
10:   end for
11: end for
12: Bsize ← higher_acc + |lower_acc|
13: Bthreshold ← |lower_acc|
```

Figure 7 - Algorithm to define the size and threshold values for the D-buffer.

Figure 6(c) presents the *acc* value when executing the algorithm detailed in Figure 7. The execution of the algorithm results in *lower_acc=-3* and *higher_acc=1*. Therefore, *Bsize* is equal to 4 and *Bthreshold* is equal to 3. The D-buffer sized according to this method enables the IP to consume the flits according to the specified rate, without data loss. The *threshold* value becomes a design parameter, included in the NI of the target IP. When the number of received flits reaches the threshold value, the IP may safely read the D-buffer.

5. RESULTS

The HERMES [11] NoC was used to evaluate the traffic modeling and the buffer sizing technique proposed in this work. The fixed network design parameters are: 8x8 mesh; credit-based flow control; 16-bit flit size; 8-flit depth internal buffers; deterministic XY routing algorithm; 2 virtual channels associated to each physical channel. The employed NoC frequency is 50MHz, corresponding to a channel rate of 800Mbps.

The experiments employ three synthetic traffic models (Table 1). Control signals are used as noise traffic, being transmitted using a complement spatial distribution by all IPs not transmitting HTTP or HDTV traffic. HTTP traffic is used to disturb the real time streaming traffic, HDTV. The HDTV traffic is characterized with temporal requirements.

Table 1 – 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)

Control signal packets are modeled with small 15-flit packets, which represent a typical size for this type of traffic. These packets are continuously generated in time, characterizing a CBR flow. The injection rate is 16 Mbps, corresponding to 2% of channel bandwidth. HTTP traffic is modeled using Pareto ON-OFF injection process, where packets are generated during the ON period at 160 Mbps. HDTV traffic is modeled using the described ON-OFF model. 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 or traced-based packets [12].

Figure 8 illustrates a first spatial traffic distribution, with two HDTV initiators (M1-M2), with 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 [9]. HDTV and HTTP flows are placed next to the middle of the network, to maximize the noise traffic influence in these flows.

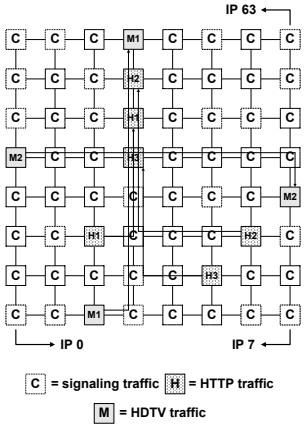

Figure 8 – Spatial traffic distribution, with HDTV M1 being disturbed by three HTTP flows.

5.1 Simulation Scenario 1: Frames With Fixed Packet Sizes

This scenario models HDTV flows with the ON-OFF traffic model, using fixed-size packets, with size varying between 500 and 1500 flits. Table 2 presents the results.

Table 2 – Simulation results for HDTV flow with fixed-size packets.

HDTV packet size	HTTP sources	HTTP packet size	jitter (HDTV)	*threshold* (HDTV)	*buffer size* (HDTV)
500	No concurrency		0	0	375
500	2	250	0.23	0	478
1000	2	500	0.33	0	944
1500	3	750	0.38	0	1427

The second line of Table 2 presents one HDTV flow without concurrent HTTP flows nor noise traffic. Since there is no concurrency, all flits arrive at the *D-buffer* with the same latency, resulting in a null jitter. The threshold value (0) means that flits can be consumed as soon as they arrive at the *D-buffer*. Two reasons explain this fact: (*i*) as all packets have the same size, the idle time between packets is the same, resulting in enough time to consume the flits; (*ii*) the consumed bandwidth is inferior to the channel bandwidth, thus NoC channels do not saturate. The HDTV application rate takes 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*. Multiplying the packet size, 500, by 0.75 (percentage of flits not consumed), the result is 375, the obtained buffer size.

Lines three to five of Table 2 present one HDTV flow competing with two HTTP flows and noise traffic. Flits arrive at the *D-buffer* with a jitter varying between 23% and 38%. The previously discussed factors introduce jitter, which is also influenced by the packet size. Even with concurrency between flows, the threshold value remains zero. Again, fixed packet sizes and application rates inferior to the channel rate explain this value. However, the concurrency induces a buffer size bigger than the relationship "packet size"/"stored flits", used when there is no concurrency.

Figure 9 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). 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 (6^{th} column, Table 2) of the second and third rows, the difference is 103, corresponding to the right shifting of the curve. These 103 extra buffer slots are required to absorb the jitter introduced by the concurrency between flows.

This first experiment demonstrated the effect of the concurrency between flows in the jitter and buffer size, even when virtual channels are employed. The curves of Figure 9 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.

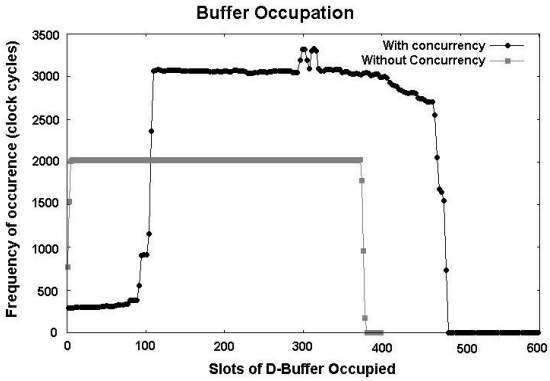

Figure 9 – D-buffer occupation using HDTV with fixed packet sizes, 500-flit packets HDTV flow without (gray) and with (black) concurrency.

5.2 Simulation Scenario 2: Frames Sizes from Traffic Traces

The second experiment models the HDTV flows with the ON-OFF traffic model, employing packets with variable size, obtained from traffic traces [12]. Traces have an average packet size equal to 1500 flits, varying from a minimum of 15 flits and a maximum of 9066 flits. Table 3 presents the simulation results.

Table 3 – Simulation results for HDTV flow with packets with variable size.

HTTP sources	HTTP packet size	jitter (HDTV)	*threshold* (HDTV)	*higher_acc* (HDTV)	*buffer size* (HDTV)
No concurrency		0	4043	640	4683
2	600	0.25	4045	640	4685
2	2000	0.43	3436	1251	4687
3	750	0.4	4053	651	4692
3	1000	0.45	4036	651	4687

The second line of Table 3 presents an experiment with one HDTV flow without concurrent flows. The jitter between packets is zero, as in the previous scenario. Note the *threshold* parameter,

which is now 4043. The threshold is greater than zero, since real application data traces contain different packet sizes. Figure 10 shows packets arriving at the *D-buffer*, being consumed by the application, considering threshold value equal to zero. The interval between packets is proportional to the packet size. As packet 2 is longer than packet 1, the interval between these two packets is longer than the required time to consume packet 1, causing an interval gap between packets. 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 flits to be stored in the D-buffer before data consumption starts. This value is high because a significant variation in packet length is present in the trace files.

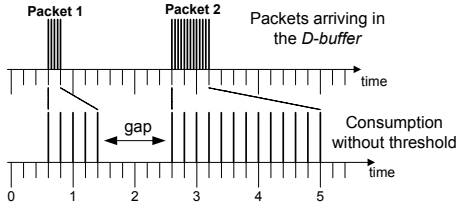

Figure 10 – Data consumption without the threshold parameter, causing starvation.

The column *higher_acc* corresponds to the amount of flits that must be stored to avoid buffer overflow (Figure 7), as in the first scenario. The buffer size value corresponds to the addition of the threshold and *higher_acc* parameters. The third and fourth lines of Table 3 show the trace-based HDTV with the concurrency of two HTTP flows and traffic noise. These experiments display an import jitter, varying from 25% to 43%. It is important to observe that the *higher_acc* value increases with the packet size of the disturbing flows (600 flits in the third line and 2000 in the fourth). The fifth and sixth lines show an experiment with 4 concurrent flows, 3 HTTP and one HDTV. These experiments show similar results, with an important jitter (40% to 45%).

Figure 11 presents the buffer use for a flow having variable packet sizes. As shown in Figure 9, flows with fixed packet sizes use the buffer in a uniform way. Figure 11 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.

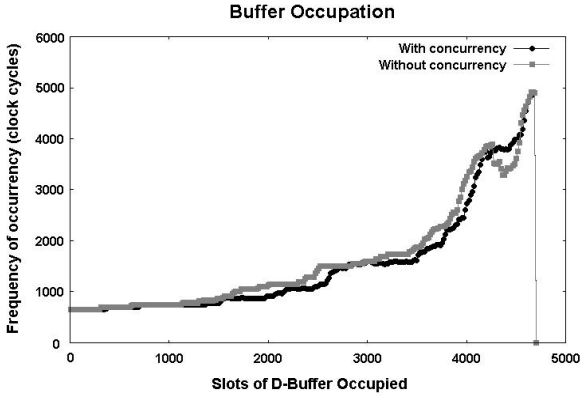

Figure 11 – D-buffer using HDTV with variable packet sizes, HDTV flow without (gray) and with (black) concurrency of 750-flit HTTP packets.

6. CONCLUSIONS AND FUTURE WORK

This work presented a method to define the size of decoupling buffers (*D-Buffers*), attached to the NoC output ports, to minimize the effect of jitter occurrence on a given traffic. Two parameters characterize decoupling buffers: size and threshold. The use of decoupling buffers is necessary for applications that process data periodically in time, in contrast with aperiodic data reception from NoC. The obtained results show that modeling data traffic with fixed frames sizes leads to underestimating buffer occupation. The use of trace based traffic models allows the estimation of buffers considering the inherent variability of streaming data, which makes necessary a higher level of threshold, compared to estimations based on synthetic traffic models.

Another parameter evaluated during simulation was buffer occupation during simulation. Results indicated that during most of the simulation run, the buffer occupation is in average 70%. Results indicated that the buffer occupation was near 100% during most of simulation run, caused by larger HDTV packets. Thus, depending on application requirements (e.g. tolerance on data losses), it is possible to use only part of the obtained size without significant effect on application behavior.

Future work includes developing a technique to estimate the *recv* array contents, based on statistical properties of the incoming traffic. This can be useful, considering that the method presented in this work estimates buffer dimensioning based in all data that arrives on the target IP, being time-consuming.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] Benini, L.; De Micheli,G. "Networks on chips: a new SoC paradigm". IEEE Comp., v.35(1), 2002, pp. 70-78.

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

[3] Hu, J. et al. “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), 2006, pp. 2919-2933.

[4] Frost, V.; Melamed, B. “Traffic Modeling for Telecommunications Networks”. IEEE Communications Magazine, 32(3), 1994, pp. 70-81.

[5] Chandra, V. et al. “An interconnect channel design methodology for high performance integrated circuits”. In: DATE´04, pp. 1138-1143.

[6] Coenen, M. et al. “A Buffer-Sizing Algorithm for Networks on Chip using TDMA and Credit-Based End-to-End Flow Control.”. In: CODES+ISSS´06, pp. 130-135.

[7] Tedesco, L. et al. “Application driven traffic modeling for NoCs”. In: SBCCI´06, pp. 62-67.

[8] Hu, J.; Marculescu, R. “Dyad – Smart routing for networks on chip”. In: DAC’04, pp. 260-263.

[9] Tedesco, L. et al. “Traffic Generation and Performance Evaluation for Mesh-based NoCs ”. In: SBCCI´05, pp. 184-189.

[10] Guerrier, P.; Greiner, A. “A Generic Architecture for On-Chip Packet-Switched Interconnections”. In: DATE´00, pp. 250-256.

[11] Moraes, F. et al. “Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration, the VLSI Journal, 38(1), 2004, pp. 69-93.

[12] Volisz, A. et al. “MPEG-4 and H.263 Video Traces for Network Performance Evaluation”. http://www-tkn.ee.tu-berlin.de/research/trace/trace.html. Captured in 2007.