Traffic Generation and Performance Evaluation for Mesh-based NoCs

Leonel Tedesco ltedesco@inf.pucrs.br
Aline Mello alinev@inf.pucrs.br
Diego Garibotti dgaribotti@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 - Prédio 30 / Bloco 4 - 90619-900 - Porto Alegre – RS – BRASIL

ABSTRACT

The designer of a system on a chip (SoC) that connects IP cores through a network on chip (NoC) needs methods to support application performance evaluation. Two key aspects these methods have to address are the generation and evaluation of network traffic. Traffic generation allows injecting packets in the network according to application constraint specifications such as transmission rate and end-to-end latency. Performance evaluation helps in computing latency and throughput at network channels/interfaces, as well as to identify congestion and hot-spots. This paper reviews related works in traffic generation and performance evaluation for mesh topology NoCs, and proposes general methods for both aspects. Three parameters are used here to define traffic generation: packet spatial distribution, packet injection rate and packet size. Two types of methods to evaluate performance in NoCs are discussed: (i) external evaluation, a common strategy found in related works, where the network is considered as a black box and traffic results are obtained only from the external network interfaces; (ii) internal evaluation, where performance is computed in each network channel. The paper presents the result of experiments conducted in an 8x8 mesh network, varying the routing algorithms and the number of virtual channels. The main contribution of this work is the set of methods for internal NoC evaluation, which help designers to optimize the network under different traffic scenarios.

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

Networks on Chip, Performance Evaluation, Traffic Modeling.

1. INTRODUCTION

Systems on a Chip (SoCs) require scalable communication architectures able to interconnect several dozens or hundreds of cores. Busses allow only one communication transaction at a time, all cores share the same communication bandwidth in the system and scalability is limited to a few dozen cores. Networks on Chip (NoCs) are a promising alternative to busses [1][2].

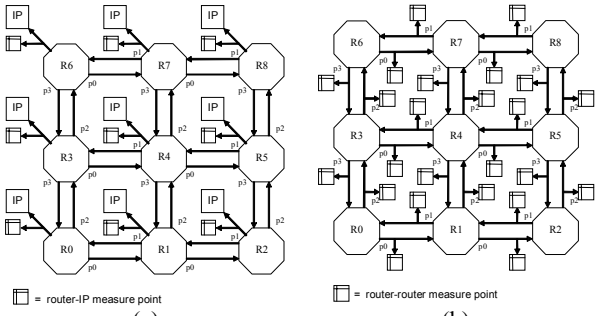
NoCs bring to the design of embedded systems concepts originated from computer networks, data communication and distributed systems. The main advantages of NoCs, when compared to busses, are communication parallelism, scalability and reusability [3][4]. However, NoC designs must consider power and silicon area constraints, in contrast to traditional network design [5].

The main steps of a NoC design are architecture specification, traffic modeling, and performance evaluation. *Architecture specification* defines network parameters, such as topology (how routers are connected), switching mode (how packets move through the routers) and routing algorithm (path taken by a packet between the source and the target nodes). *Traffic modeling* specifies the traffic parameters for each communication initiator, offering the designer the possibility to analyze, quantitatively and qualitatively, communication events taking place in the network. Modeling involves parameters such as packet spatial distribution, packet injection rates and packet length. *Performance evaluation* helps designers in computing latency and throughput values, as well as in identifying traffic congestion regions.

Related works in NoC performance evaluation [3][7][8][9][10] consider the NoC as an IP. In this situation, only the external interfaces, which communicate with the system IPs generate data for performance evaluation (Figure 1(a)). The data allows computing latency and throughput for packets, but no specific analysis of the internal behavior of NoC links is possible. This scheme is called here *external evaluation*.

This work has as two main goals. The first goal is to present a general method for NoC traffic generation. The second goal consists in proposing methods for *internal evaluation* as defined in Figure 1(b). Internal evaluation collects data at each NoC channel, computing average link bandwidth use, average time for flit transmission and resources utilization.

This paper is organized as follows. Section 1 presents the state of art in NoC traffic generation and performance evaluation. Section 2 approaches the traffic generation problem. Section 3 presents internal performance evaluation techniques, while Section 4 investigates external performance evaluation. Section 5 applies traffic generation and performance evaluation using the HERMES NoC [11]. Conclusions and directions for future work are the subject of Section 6.



(a) (b)

Figure 1 - (a) NoC external traffic data access points. (b) NoC internal traffic data access points.

1. RELATED WORK

Adriahantenaina et al. [3] compares the performance of a PI-bus w.r.t. the SPIN NoC, using external evaluation. The generated traffic is random, with each initiator generating successive packets separated by a random number of clock cycles. The results show that for a dozen of cores, the performance of SPIN NoC is superior to that of the PI-Bus. The authors show that a 4% offered load saturates the PI-Bus, while the SPIN NoC only saturates after a 28% offered load.

Bolotin et al. [6] present QNoC, a NoC with QoS features. This work defines four service classes: signaling, real-time, RD/WR and block-transfer. QNoC employs 4 virtual channels, one for each traffic class. Packets belonging to each class have different priorities, using packet preemption if a higher priority packet arrives in a router. Traffic generation adopts two spatial distributions: uniform, where all nodes have the same probability to be targets; and non-uniform, where the nodes closest to the sources have higher priority to be targets. Internal and external evaluation are used. A 3D graph with a normalized load in each link is used to help the designer to customize the NoC according to the QoS requirements, characterizing the use of internal evaluation.

Hu and Marculescu [8] propose a routing algorithm named Dyad, which combines the low latency of deterministic routing with the high throughput of adaptive routing. They compare the proposed algorithm to three other routing algorithms, namely *XY*, *OE-fixed*, and *odd-even*. Two traffic scenarios were adopted: (*i*) a 6x6 mesh with IPs generating 5-flit packets with exponential distributions, using uniform and complement traffic patterns; (*ii*) a 4x4 mesh where nine IPs are randomly picked to generate multimedia traffic, with the remaining IPs generating uniform traffic. Average latency is computed using external evaluation. According to the authors, Dyad outperforms the XY routing algorithm by 60% in terms of sustainable throughput, for complement traffic pattern.

Banerjee et al. [7] evaluate latency, throughput and power consumption of a 4x4 mesh NoC. During traffic generation targets are randomly chosen, with uniform interval between packets. External evaluation is used, considering the packets latency according to the data injection rate. To improve NoC performance the authors suggest using a greater number of virtual channels and virtual-cut-through switching mode.

Ye et al. [9] propose a *contention-look-ahead* routing algorithm. Routers analyze the status of their neighbors to decide which output port to use for a given packet. They compare this routing algorithm with two others, *hot potato* and *XY*, in a 4x4 mesh NoC. Five service classes are proposed: (*i*) *memory access request*; (*ii*) *cache coherence synchronization*; (*iii*) *data fetch*; (*iv*) *data update* and (*v*) *IO interrupt*. A set of benchmark applications is employed to generate traffic. External evaluation is used, considering parameters such as average latency and total execution time, according to different packet sizes. Using performance and power consumption data, the authors established a qualitative analysis of the packet size impact on the overall MPSoC performance.

Vellanki et al. analyze Guaranteed Throughput (GT) and Best-Effort (BE) traffic in a 4x4 mesh [10]. Again, targets are randomly chosen, with a uniform interval between packets. External evaluation enables the generation of a packet latency distribution for GT and BE traffics, according to the packet injection rates.

This review of the state of the art in traffic generation and performance evaluation reveals that most approaches are specific for a given communication architecture or for comparing two distinct architectures. Authors often use only random traffic and latency evaluation to validate a NoC design. However, the network behavior is a function not only of its architecture, but also of the application running on it. For example, some applications consists in long messages mostly (e.g. streaming video), while in others short messages dominate (e.g. sensor network controllers). Thus, it is important to have available traffic generation techniques able to reflect real traffic behaviors, and metrics to evaluate the internal performance of the NoC.

2. TRAFFIC GENERATION

Traffic modeling defines the structure of data transmission from initiators to targets. According to [12], traffic modeling is defined by 3 parameters: packet spatial distribution, packet injection rate and packet size.

The packet spatial distribution specifies the relationship between initiators and targets. Due to the similarities between parallel architectures and MPSoCs, patterns usually observed in parallel applications can be used to define the spatial distribution of packets in NoCs. Patterns current used are *bit reversal*, *perfect shuffle*, *butterfly*, *matrix transpose* and *complement*. Except for [8], which uses random and complement patterns, all other works mentioned in the previous Section employ only the random pattern. Non-uniform traffic patterns cause traffic concentrations that are closer to real applications behavior.

Figure 2 illustrates the complement traffic pattern in a 4x4 mesh network, using the XY routing algorithm. Arrows in Figure indicate the paths from source to target routers. Two different initiators simultaneously use channels in the X and Y bisections. Due to this feature, congestion in the network bisections, the complement pattern may be used to evaluate the performance gains when using virtual channels, since these allow sharing the use of a physical channel by distinct packet flows. Note that, if different routing algorithms are used, channel usage for the same traffic pattern is different.

Following the definition of the traffic spatial distribution pattern, the user chooses an injection rate and packet sizes for each IP. The injection rate is always a fraction of the maximum channel bandwidth, in bps (bits per second). Note that injection rates are associated only to links connecting an IP to its router. Also, the injection rate is an ideal value that may not be respected if the network entrance is congested.

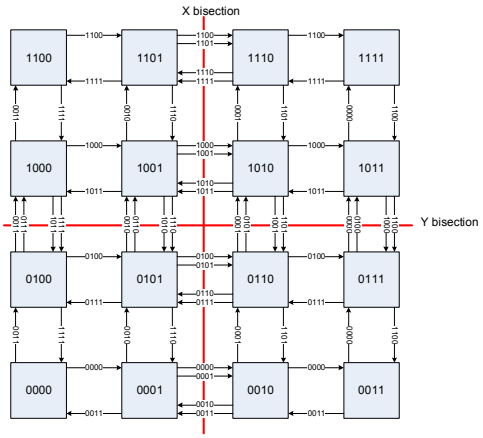


Figure 2 - Complement traffic pattern, in a 4x4 mesh network, using a XY routing algorithm.

Figure 3 illustrates two distinct injection rates. For simplification purposes, suppose 1 flit equals to 1 bit and the available channel bandwidth is equal to 10 bps. If a packet is transmitted at 10 bps, the bandwidth is completely used and represents a load of 100%, as depicted in (a). If packets are transmitted at 5 bps, only 50% of the bandwidth is used, as illustrated in Figure 3(b).

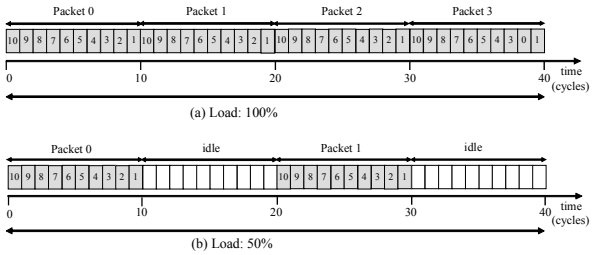


Figure 3 - Two data injection rates: (a) 100% load, (b) 50% load.

From the injection rates defined by the user, globally or for each IP, a traffic generator can compute the interval between packets. This interval, named *idle*, is computed according to Equation 1.

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

where:
- *chr*: channel transmission rate (Mbps) (channel bandwidth);
- *ipr*: IP transmission rate (Mbps) (injection rate);
- *pcksize*: packet size (number of flits);
- *ncyclesflit*: number of cycles to transmit one flit.

It is important to mention one structural parameter of the network in Equation 1: *ncyclesflit*. This parameter represents the time spent by the router to transmit one flit, and is a function of the control flow mechanism adopted by the network. For example, in handshake control flow *ncyclesflit* is equal to 2 and for credit based control flow *ncyclesflit* is equal to 1.

One important consideration is when packets are created by the IP. This information, named *pkt_ts* (packet timestamp), is inserted in each flit occupying the first flits of the payload. A global counter, named *timestamp*, is used as temporal reference and contains the number of clock cycles passed since the start of simulation. The first packet receives *pkt_ts* equal to zero. The following packets receive *pkt_ts* according to Equation 2. The value of *pkt_ts* allows latency and throughput evaluation, since packets may be delayed due to congestion in the network. The value *pkt_ts* in fact represents the earlier moment a packet may be inserted.

$$pkt\_ts = prvtmp + (pcksize * ncyckesflits) + idle \qquad (2)$$

where: *prvtmp*: timestamp the last flit of the last packet sent.

3. INTERNAL PERFORMANCE EVALUATION

Internal performance evaluation allows the designer to verify the performance at each NoC channel, according to the NoC structural parameters and traffic generation. During the NoC simulation, data is collected in all router ports (Figure 1(b)).

Identification of hot spots and critical paths created by packets with higher latency values allows the designer to optimize the NoC. Examples of structural optimizations are to insert virtual channels, to eliminate links unused by the application [6], change link width, resize buffers and change the routing algorithm. Example of functional optimization is to change the IPs placement.

3.1 Channel use and throughput

Two metrics allow evaluating the average channel use: average channel bandwidth and throughput.

Average channel bandwidth (*abw*) indicates the percentage of use for a given channel during the transmission of packets. If the network is correctly dimensioned, uniform use of all channels is expected. Network optimizations, as insertion of virtual channels or buffer sizing, may be necessary if *abw* values are unevenly distributed. Equation 3 computes *abw* for a NoC channel, by dividing the summation of the time to transmit each packet by the total time to transmit all packets. Figure 4 illustrates the use of Equation 3.

$$abw_{channelXY} = \frac{\sum_{i=1}^{npck} (tpl - tpf)_i}{tsl - tsf} \qquad (3)$$

where:
- *tpl*: timestamp of the last flit leaving an output port;
- *tpf*: timestamp of the first flit leaving an output port;
- *npck*: number of packets transmitted in an output port;
- *tsl*: timestamp of the last flit of the last packet leaving an output port of a router;
- *tsf*: timestamp of the first flit of the first packet leaving an output port of a router.

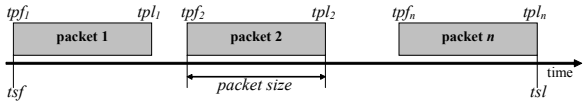


Figure 4 - Parameters used in Equation 3 to compute average channel bandwidth.

Equation 3 models the exact average channel use for store-and-forward and virtual cut-through networks, but may overestimate

the channel use in wormhole networks. This happens because in wormhole networks, once a packet is transmitted, channels are allocated for this packet until the end of its transmission. Therefore, flits can block channels due to downstream congestion in the path of the packet.

Throughput (*thr*) corresponds to the effective number of bits transmitted in a given amount of time, in bits per second. Equation 4 computes *thr* for each NoC channel, by dividing the total number of transmitted bits by the simulation time, in seconds.

$$thr_{channelXY} = \frac{npck * pcksize * flitsize}{nsimc * T} \quad (4)$$

where:

flitsize: flit width;

nsimc: total simulation cycles for sending all flits of all packets passing in the channel;

T: clock period.

Each initiator injects data into the network with a given throughput (see *ipr* in Equation 1). The throughput computation at each channel allows verifying deviations in the original rate. This evaluation is important for evaluating the QoS of data flows, due to latency and throughput constraints.

3.2 Average time for flit transmission

The average time for flit transmission (*avcpf*) corresponds to the average number of cycles to transmit one flit. Equation 5 shows how to compute *avcpf*.

$$avcpf_{channelXY} = \frac{\sum_{i=1}^{npck}\left(\frac{(tpl - tpf)_i}{pcksize}\right)}{npck} \quad (5)$$

This metric indicates packet congestion on router channels. Without congestion *avcpf* is expected to be one for credit-based control flow and two for handshake control flow. Higher *avcpf* also means channel allocation without data transmission.

3.3 Idle time between packets

IPs transmit packets uniformly distributed in time, as explained in Equation 1. It is expected that routers transmit packets with the specified temporal distribution. However, packets may experiment different delays to be transmitted due to congestion in the network.

Figure 5 shows the idle time behavior of a flow in a given router output. The graph plots in the X-axis the *idle* time between packets w.r.t. the number of idle time intervals. In this example, a complement traffic pattern was adopted in an 8x8 mesh with XY routing. All IPs send data at 320Mbps, which gives an idle interval between packets of 200 clock cycles. The graph shows idle time distribution at the north port of router 31 created by packets from router 0 targeted to router 63. The average *idle* time is indeed around 200 clock cycles, but it is possible to observe a significant spreading of values. For GT traffic, the network must provide mechanisms to avoid or reduce the spreading displayed in Figure 5.

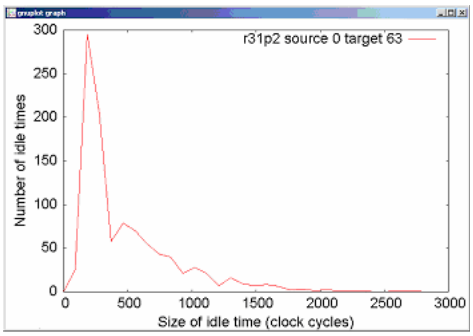


Figure 5 - Distribution idle time between packets.

4. EXTERNAL PERFORMANCE EVALUATION

External performance evaluation deals with the network as a black box, and all results are obtained from the network external interfaces only. External evaluation comprises Chaos Normal Form graphs and latency distribution analysis.

4.1 Chaos Normal Form (CNF) graphs

Chaos Normal Form graphs [13] display achieved network throughput on one graph and network latency on a second graph. In both graphs, the X-axis corresponds to normalized applied load. The normalized applied load is obtained by dividing the IP transmission rate by the channel transmission rate.

By using two graphs, network latency and throughput are shown both below and above the network saturation point. A network is said *to saturate* when increases in applied load do not result in higher throughput.

4.2 Latency distribution

This analysis is similar to the one shown in Section 3.3. A latency distribution graph plots latency values (in the X-axis) versus the number of packets (in the Y-axis). The goal is to identify the latency spreading. A small latency spreading allows predicting when packets arrive at destination nodes more easily, allowing to support GT traffic. If a large latency spreading is observed, this means packets are transmitted with little or no temporal restriction, characterizing BE traffic.

5. RESULTS

HERMES [11] is a packet-switched mesh topology NoC , whose basic elements are a router with centralized control logic and five bi-directional ports: East, West, North, South and Local. The Local port is connected to an IP and the others are connected to neighbor routers. Each port has an input buffer for temporary storage. The switching mode is wormhole. Handshake or credit-based flow control may be used. The flit size is parameterizable, and the maximum number of flits in a packet is fixed at $2^{(\text{flit size, in bits})}$. The first and second flits of a packet indicate the header information, being respectively the address of the target node, and the number of flits in the payload. The remaining portion of the packet is the data payload.

5.1 Experimental setup

Network on chip was described in RTL VHDL. The fixed network design parameters are: 8x8 routers mesh; credit-based flow control; 16-bit flit size; 8-flit buffers. For evaluation purposes two

routing algorithms (deterministic XY and partially adaptive west-first), and implementations with and without virtual channels are used. Results are obtained with complement traffic pattern, 10 to 60% normalized applied load, and 50-flit packets. Each router generated 1000 packets. Simulation execution time for each scenario was about 35 minutes.

5.2 Average channel use

Figure 6 shows the normalized average bandwidth (*abw* – Equation 3) at each NoC link (each link contains two unidirectional channels) for west-first and XY routing algorithms. The format of these graphs is similar to the ones presented in [6].

Note in **Figure 6**(a) the higher use of the links to the left of the X bisection when compared to **Figure 6**(b). The nature of the west-first routing explains this behavior, since the algorithm tries to route packets first in the west direction (left) whenever possible. Also, note in **Figure 6**(b), XY routing distributes more homogeneously the channel bandwidth use for this traffic pattern (complement).

5.3 Average time for flit transmission

From **Figure 6** it is possible to observe that the middle of the network concentrates the higher traffic density. From this, it is possible to expect contention in the routers placed in the network periphery. **Figure 7** (a) plots the average time to transmit one flit, when no virtual channels are used. Note the average CPF (cycles per flit) in the periphery links. These values indicate that these flits stay 6 clock cycles stored in buffers in average, before being sent, thus allocating bandwidth with no effective data transmission.

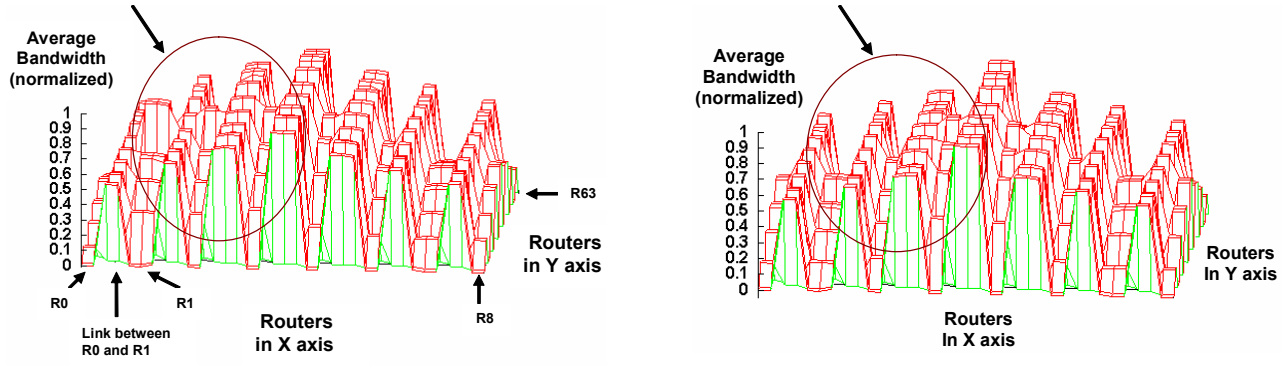
One way to reduce contention in this case is to multiplex a physical channel using virtual channels. **Figure 7**(b) plots the average time to transmit one flit, when each physical channel is multiplexed in two virtual channels. This plot demonstrates the direct benefit of virtual channels: contention reduction in wormhole networks.

5.4 CNF graphs

Figure 8 illustrates accepted traffic and average latency w.r.t. the normalized offered load. Graphs in **Figure 6** and **Figure 7** are obtained simulating the network for a given offered load. To obtain CNF graphs the user simulates the network for each plotted point. Works presented in [3][6][7][8] also use CNF graphs.

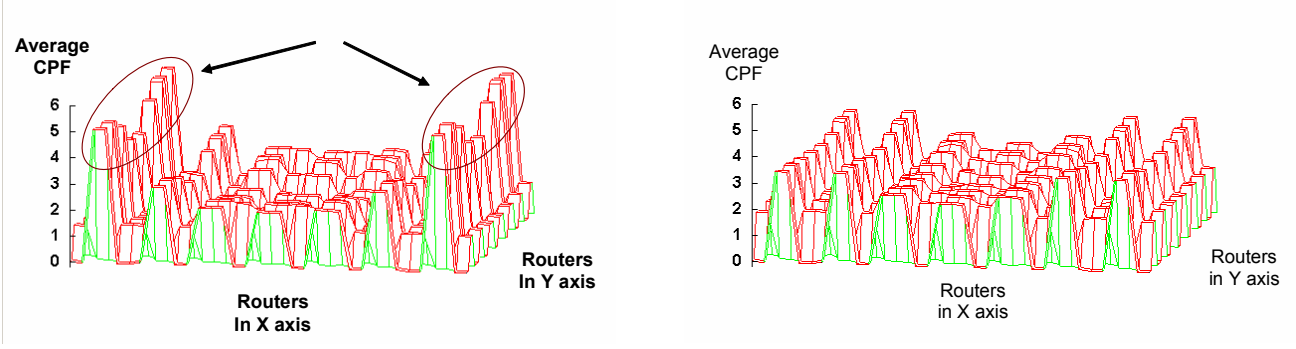
Figure 8(a) shows that virtual channels increase the maximum accept traffic and the superiority of the XY over the west-first routing algorithm for the network in use. Note in this Figure the saturation point for each curve. Saturation starts when the accept traffic is inferior to the offered load. For example, for 20% offered load, only the xy_2VC implementation is not saturated.

Up to the saturation point, Figure 8(b), the average latency is constant. Beyond the saturation point, latency increases significantly. Moving the saturation point to right improves the network performance. However, NoC design complexity and area (cost) also increase.



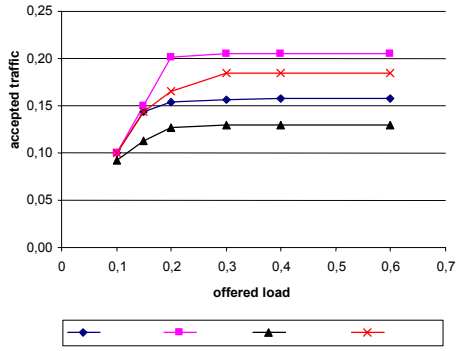
(a) (b)

Figure 6 – Average bandwidth per link for two routing algorithms (20% load).

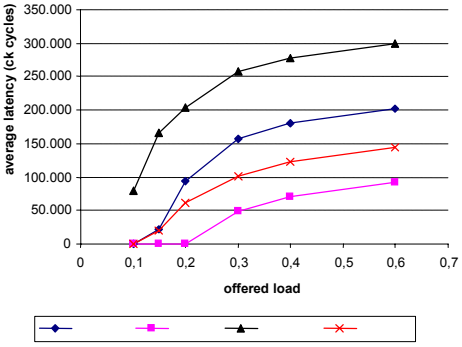


(a) (b)

Figure 7 – Average time for flit transmission (20% load).



(a) accepted traffic



(b) average latency
Figure 8 - CNF graphs.

5.5 Latency distribution analysis

Figure 9 plots the number of packets with a given latency value, with and without VCs, for 30% offered load. This plot shows the distribution of packet latency values, and the spreading of these values. When virtual channels are used (2 VC), the average packet latency is 48,964 clock cycles. When no VC is used, the average packet latency is 155,732 clock cycles. The smaller spreading observed in the VC curve allows estimating the time to transmit packets more precisely, a key factor to implement QoS in NoCs.

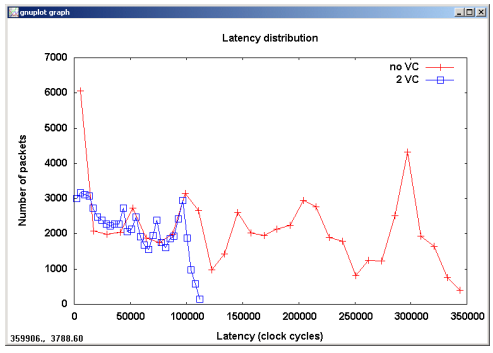


Figure 9 – Latency distribution with and without virtual channels, 30% offered load, XY routing.

Vellanki [10] presents a similar latency analysis, for BE and GT traffics. The higher priority for GT traffic guarantees a small spreading in latency values.

6. CONCLUSIONS AND FUTURE WORK

This work presented traffic generation and measuring parameters for internal and external NoC performance evaluation. The main contribution is a set of metrics for internal NoC evaluation: channel use, flit transmission time and idle time between packets. Using such metrics, designers can optimize the network under different traffic scenarios.

Future work includes statistical traffic generation and the use of real applications trace files. Statistical traffic allows injecting data varying packet length and idle time between packets. Several standards, such as JPEG, are defined with statistical models. Trace files corresponds to real applications traffic. The lack of benchmarks leads designers to use mostly synthetic traffic generators.

7. ACKNOWLEDGMENTS

Work partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, projects 550009/2003-5 (RICHA) and 307665/2003-2.

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] Guerrier. P.; Greiner. A. A generic architecture for on-chip packet-switched interconnections. *DATE´00*, 2000, pp. 250-256.

[3] Adriahantenaina, A. et al. SPIN: a Scalable. Packet Switched, On-chip Micro-Network. *DATE´03*, 2003, pp. 70-73.

[4] Dally, W.; Towles, B. Route Packets, Not Wires: On-chip Interconnection Networks. *DAC´01*, 2001, pp. 684-689.

[5] Benini, L.; De Micheli, G. Powering networks on chips: energy-efficient and reliable interconnect design for SoCs. *14th ISSS'01*, 2001, pp. 33-38.

[6] Bolotin E. et al. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, v.50(2-3), 2004, pp. 105-128.

[7] Banerjee, L. et al. A power and performance model for networks on chip architectures. *DATE´04*, 2004, pp. 1250-1255.

[8] Hu, J.; Marculescu, R. Dyad – Smart routing for networks on chip. *DAC´04*, 2004, pp. 260-263.

[9] Ye, T.; Benini, L.; De Micheli, G. Packetization and routing analysis of on-chip multiprocessor networks. *Journal of Systems Architecture*, v. 50(2-3), 2004, 81-104.

[10] Vellanki, P. et al. Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures. *Integration, the VLSI Journal*, v.38(3), Jan. 2005, pp. 353-382.

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

[12] Duato, J. et al. *Interconnection Networks*. Elsevier Science, 2002, 600 p.

[13] Dally, W.; Towles, B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004, 550 p.