Evaluation of a Non-intrusive packet delivery monitoring service on Networks-on-Chip

Gabriel Ganzer[1], Marcelo Daniel Berejuck[2]

[1]Computer Engineering Department, Politecnico di Torino, Piedmont, Italy
[2]Computer Engineering Department, Federal University of Santa Catarina, Ararangua, Brazil

***Abstract*— *We present the design and evaluation of a non-intrusive packet delivery monitoring service on a Network-on-Chip (NoC) that focus real-time systems. Recent works show that using adaptive routing or optimization techniques are solutions to improve its latency. These strategies usually need to know the traffic behavior previously to make changes. A monitoring service is indicated as a solution to this issue, but since silicon consumption is a restriction in these projects, most of them use routers or other NoC's resources to performer such task. Our design is based on a strategy that does not interfere with the NoC operation to collect and to evaluate traffic information.*

***Keywords*— System-on-chip, Network-on-Chip, Computational Observability.**

I. INTRODUCTION

The Embedded Systems field has used Systems-on-Chip (SoC) with multiple heterogeneous processing units also named cores as means to ensure performance and temporal guarantees to real-time systems. It has become consolidated that a Network-on-Chip (NoC) [1] should perform the communication between those processing units. Generally, those systems are designed to handle flows from different types of tasks at the same time, not only the ones of real-time origin, and a NoC must allow guarantees on bandwidth and latency for each flow.

An approach to achieve Quality-of-Service (QoS) is the use of circuit switching, whose resources are not shared until the end of packet transmission; thereby, real-time guarantees are easily achieved. HERMES [2], QnoC [3], SoCBUS [4][5], SoCIN [6][7], and Xpipes [8][9] are examples of NoCs based on that technique. Time Division Multiplexing (TDM) is another method of circuit switching employed at Æhereal [10][11]. In this technique, the time domain is divided into several recurrent time slots of fixed length, one for each channel. TDM implements virtual circuit switching, which may imply better resource utilization. A third technique is to estimate the network traffic and scale the project by its greater restriction. That is the case of RTSNoC , this paper object of study, a network that aims real-time systems and provides a predictable worst-case latency (WCL) at design time, in which there is no resources reservation.

Critical constraints occur sporadically in SoC communication, which consequently could result in resource depletion in NoCs that use the worst-case latency prediction technique. Thus, non-critical flows also called best-effort (BE) have no QoS guarantees, which results in an average improvement of their latency. Some recent works have proposed the use of adaptive techniques in NoCs as an alternative to providing such level of performance for both real-time and best-effort flows. Such techniques rely on a database that contains information about the current network traffic state provided by a monitoring mechanism.

We noticed that NoCs' researches that explore some monitoring system adopt, most of the times, intrusive approaches or router's resources that result in a performance decrease or a non-relevant improvement. This approach can be seen in [12], an online monitoring and adaptive routing for aging mitigation in NoCs that uses its routers for data evaluation, which proved to be an issue in their service that could be improved. In [13] is presented a NoC that implements a pseudo adaptive XY routing. Their service uses routers for both traffic data collection and evaluation, which by their conclusion increased the latency at some points since routers were busy with those tasks instead of performing the routing ones indeed.

In this context, this paper introduces a non-intrusive packet delivery monitoring service for real-time NoCs. The proposed service would be located in a separate layer, above routing. In contrast with those works presented before, this approach does not use NoC's resources or routers for such task, and it has its specialized components for traffic data collection and evaluation. Silicon consumption may be the reason for previous papers do not explore this kind of approach, since most NoCs have routers able to do a maximum of five interconnections, which is not the RTSNoC's case whose one single router can deal with five up to eight interconnections resulting in a low silicon consumption rate. Thus, a monitoring service could provide information for an adaptive routing or any other optimization technique.

The remainder of this paper has the following organization: in Section II introduces the network concept describing the internal structure of the communications channels and router architecture proposed to build the NoC. Section III introduces the non-intrusive monitoring service itself discussing its architecture and internal components. Section IV presents the experimental results obtained by simulating communication patterns of standard interconnection networks in an RTSNoC combined with the monitoring service environment, and Section V finalizes the paper with our conclusion.

II. RTSNoC ARCHITECTURE

The RTSNoC network is a latency predictive network for use in real-time systems [1]. Its focus is embedded systems in which there are more real-time applications than best-effort. This NoC is based on the flit-interleaving technique to guarantee throughput and quality-of-service, where every competing flow has their packets interleaved flit-by-flit.

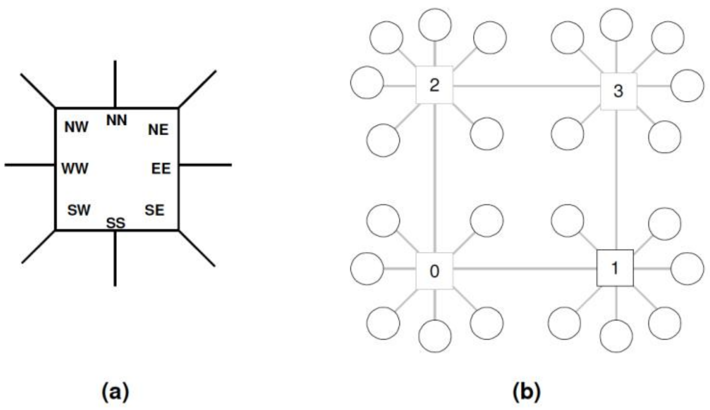


Fig.1: RTSNoC topology: (a) router with eight ports; and (b) example of a network with four routers and twenty-four processing cores.

Routers in the RTSNoC are conceived to be connected in a 2-D orthogonal topology. Each router can be configured at design-time to have different communication channels quantities, from five up to eight as shown in Fig. 1-a. The communication channels are named with cardinal points and can be connected either to one core or to another router. Fig. 2-b depicts a network with four routers and twenty-four processing elements connected in it. Each router interconnection point provides two unidirectional channels, one for input and one for output. Both channels transport packets and synchronization signals.

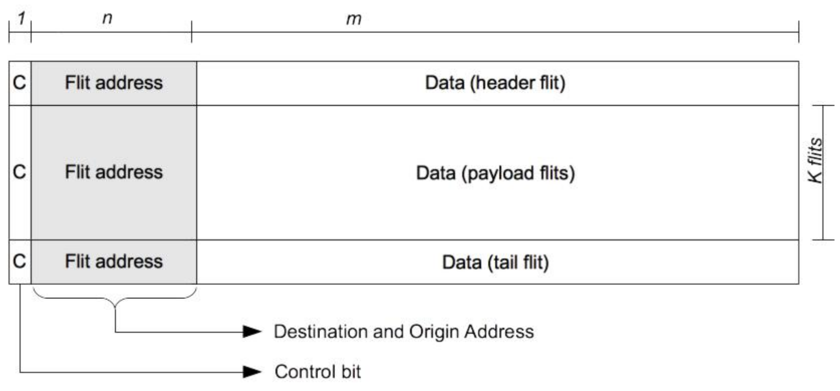


Fig.2: RTSNoC's packet format. The size of X/Y fields depends on the network size.

From the point of view of its routers, the network's packet is an arbitrary-length sequence of payload flits preceded by a header flit and followed by a tail flit. To perform the interleaving technique, RTSNoC demands extra bits on its flits, as depicted in Fig. 2. When flows compete to be routed, an internal arbiter gives priority to flits coming from farther routers. Buffers are located only at the output interface, an approach that aims to minimize the router's silicon consumption due to interleaving technique. RTSNoC ensures all flits that compose a package are delivered in the same order that they are injected into the network.

During initialization, every channel receives a different level of priority. The highest priorities are assigned to NN, SS, EE, and WW channels since they could be used in routers interconnection. In addition, these channels may send more than one flit to each grant signal. The number of flits that a channel can send to each grant depends on the number of requests occurred at the same time on previous routers in that path. Any flit has its routing request answered if it has the highest priority or if there are no other requests to the arbiter. When the request is answered, the channel receives a lower priority level and can only send other flits if there is no other one waiting to be routed.

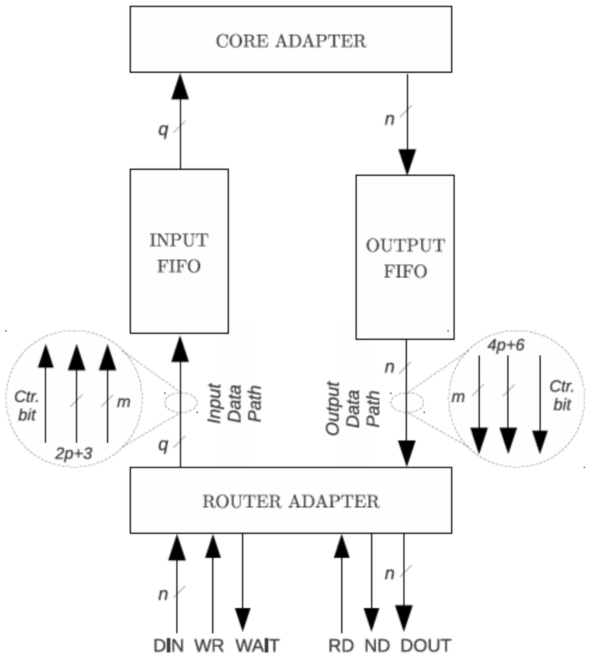


Fig.3: RTSNoC's network interface internal structure.

RTSNoC has its own Network Interface (NI) to be used between a core and router when those operate in different clock domains, as shown in Fig. 3. The NI is composed of four main components: an input FIFO, an output FIFO, a core adapter, and a router adapter. The designer at design-time establishes the size of the queues performed by FIFOs, and the network throughput is a direct function of this parameter.

The RTSNoC's project was firstly conceived only in VHDL. In order to facilitate the exploration of the project by decreasing the compilation time but still maintaining the cycle-level accuracy as well as to ensure the possibility of synthesis already provided by VHDL the RTSNoC's router and NI were modeled in SystemC, a standard language for performance evaluation that is able to provide those characteristics to the project .

III. NON-INTRUSIVE MONITORING SYSTEM

The main purpose of this work was to introduce into the RTSNoC's project a system able to observe the network traffic behavior and verify the packet delivery at run-time, without reducing the performance. This following section would discuss the design of two components used in a non-intrusive monitoring system: sniffer (S) and network manager (NM).

III.1 Sniffer

The SoC's core can be classified into three categories: emitters, receptors, and hybrids, i.e., can both send and receive data. When dealing with a service that will monitor the packet delivery in a NoC is intuitive to say that its target will be a receptor or hybrid core. The placement of a Sniffer into the RTSNoC is explicit in Fig. 4. This device captures the signals of the input channel of the core and output of the

NI, an approach that aims to guarantee accuracy in the measurement of parameters.

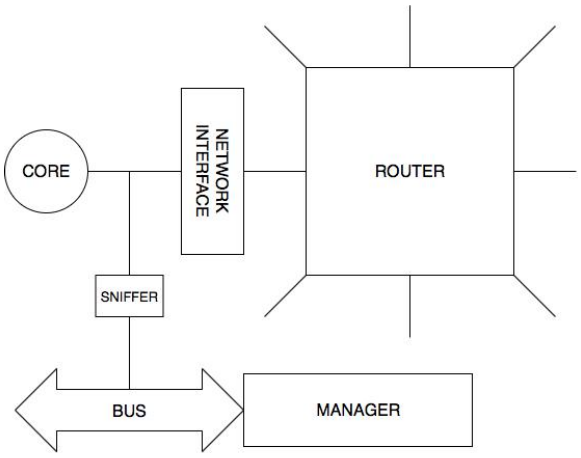


Fig.4: Placement of a sniffer in a standard NoC.

A sniffer consists of five components: extractor, demultiplexer, flit-counters, timers, registers, and a multiplexer, shown in Fig. 5. When the core sends a new request to the NI, the extractor also receives this signal and starts to "listen" the data channel and reads from flits passing by it their origin address and also their control bit. If it recognizes as a control flit, the extractor will also check whether this is a header or tail flit on the data field. Else, S will skip this check. Internally, the origin address becomes the selection signal of a demultiplexer and one of the register inputs. Each flow needs to be handled separately, i.e.; the sniffer must take into account that flits with different origin addresses can be interleaved even if the core has not received its tail. Therefore, the total outputs of the demultiplexer and consequently inputs into the multiplexer, as well as the number of required timers and registers are directly proportional to the total of valid origin addresses. The upper bound of valid addresses is given by the size in bits of its field shown as n in Fig. 2. Cores that are exclusively emitters are considered not valid addresses.

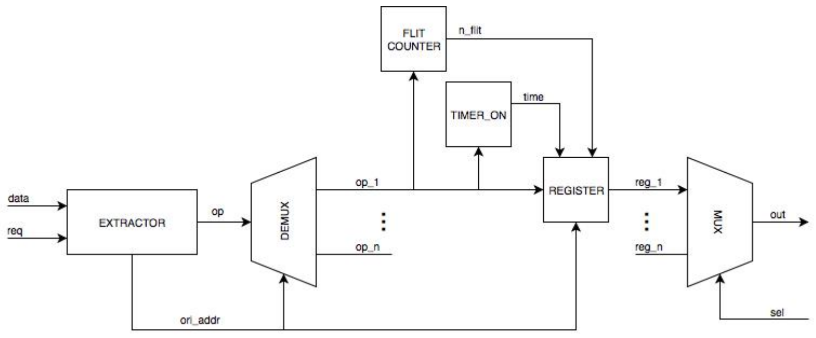


Fig.5: Internal components of a sniffer.

The values read in the request signal, in the flit control bit, and in the byte stuffing field combined forms the extractor basis to infer what signal would be assigned to the demultiplexer inputs. This signal characterizes the operation to be performed by the flit-counter, timer, and register that belongs to a particular flow. Table 1 lists the states of each of these components by the input op: **0b00** is assigned when the request signal is deactivated, regardless of what is read from the data field; **0b01** is assigned when the request signal is active and the control bit is off; **0b10** is assigned when the request signal is active, the control bit is on and the byte stuffing filed characterizes a header; finally, **0b11** is the value assigned to op when the request signal is active, the control bit is on and the byte stuffing filed characterizes a tail.

Table 1: Sniffer's internal components states.

Op	Flit-counter	Timer	Register
0b00	Idle	Increment	Idle
0b01	Increment	Increment	Idle
0b10	Reset	Reset	Idle
0b11	Register	Register	Save

III.2 Network Manager

The current monitoring systems use the NoC's resources to transport and evaluate the information obtained. This paper aims a solution that would exempt this task from routers by inserting those components responsible for the monitoring service in a layer above routing. The Sniffer, presented in the previous subsection, is responsible for observing the NoC's traffic behavior and by feeding a second component called Network Manager, with that collected information.

An NM communicates with sniffers via dedicated buses with an on-demand protocol. This approach is intended to exclude from routers' communication channels the task of transporting this information, which may infer on routing behavior. The communication topology between sniffer and manager can be either centralized when all sniffers communicate with a single NM or decentralized, where every router have their own manager. Fig. 6 illustrates a version of RTSNoC that implements a centralize topology.

It is important to highlight that in the particular case of the centralized topology that uses a single communication bus, the system complexity increases proportionally to the network's size, i.e., this topology is indicated only for small networks with few cores. In the decentralized topology, each manager is responsible for the sniffers of a single router. Both topologies can also be combined according to the application's needs and form several other topologies that will not be discussed in this paper.

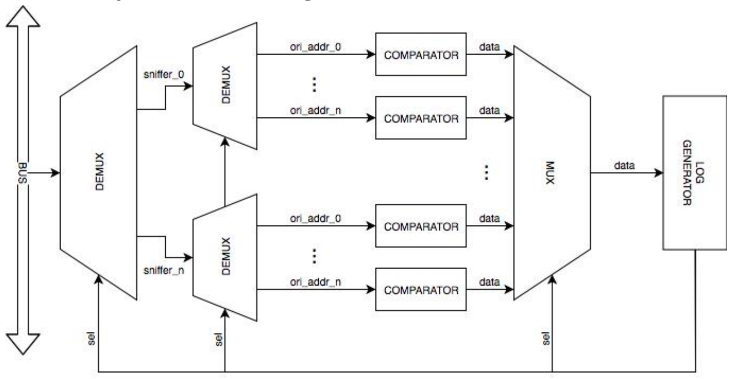


Fig.6: Internal components of a network manager.

The internal components of an NM are depicted in Fig. 7 composed basically of four different structures: comparators, demultiplexers, a log generator and a multiplexer. In that figure, the main demultiplexer is connected directly to the NM's communication bus acting as a switch to separate the data coming from each sniffer. This demultiplexer is only necessary when the communication topology is not decentralized. The second level of demultiplexers causes the sniffer data to be switched according to their source address. Again, the insertion or not of these demultiplexers, as well as the quantity of outputs is parameterizable according to the application needs. The data passes through a comparator before being forwarded to the log generator, to avoid repeating information in homogeneous flows. Finally, the log generator is responsible for receiving the data and storing it. This latter component can be either on-chip or off-chip.

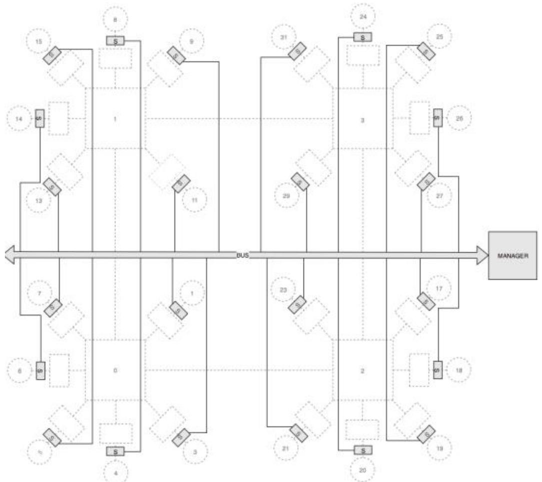


Fig.7: Centralized topology.

IV. EXPERIMENTAL RESULTS

This section describes the monitoring service evaluation under traffic patterns established by [15]. The RTSNoC version used in these experiments is the same as demonstrated in Fig. 6. That version is composed of four routers connected in orthogonal mesh able to perform up to eight interconnections with centralized monitoring topology and all twenty-four hybrid cores, i.e., they can send and to receive data. Other parameters were specified in Tab. 2.

Table 2: RTSNoC parameters.

Parameter	**Specification**
Clock frequency	1 GHz
Network size	2x2
Core numbers	24
Payload size	32 bits
Routing algorithm	Static XY
Arbitration	Flit-interleaving
FIFO's depth	32 positions
Counters resolution	11 bits
Timers resolution	17 bits

The following subsections describe general aspects about the traffic generators used to simulate real-time applications, and the experiments themselves, performed in two phases: the first one with a constant injection rate model and the second one with variable injection rate.

IV.1 Traffic generation

The experiments were performed with the aid of a traffic generator. This component can send packets with fixed or variable size and with an interval between packets also fixed or variable. The generation rate can be constant or ruled by probability functions such as Normal, Exponential or Pareto on/off. The communication pattern can also be characterized by specifying the spatial distribution of traffic generators into those with a single destination node and those with multiple destinations throughout the simulation.

Communication patterns with a single destination are based in parallel numerical algorithms such as bit reversal, perfect shuffle, butterfly, matrix transpose, and complement. In these patterns, the destination node for the messages generated by a given node is always the same. However, only at the complement pattern, every node would have a destination that is not itself, thereby, since the experiments aim to evaluate a packet-delivery monitoring service the other patterns are not considered.

There are three spatial patterns with multiple destinations: uniform, non-uniform, and local. In the uniform pattern, all nodes have the same probability of being a destination, and the number of packets sent is the same to everyone. In the non-uniform pattern, the probability of a node sending packets to another node decreases with the distance between them and neighboring nodes exchange one maximum number of packets. Finally, in the local pattern, all nearby nodes are equally likely to be a destination, and they send the same number of packets. However, non-adjacent ones have zero probability. All of these patterns are used in the experiments.

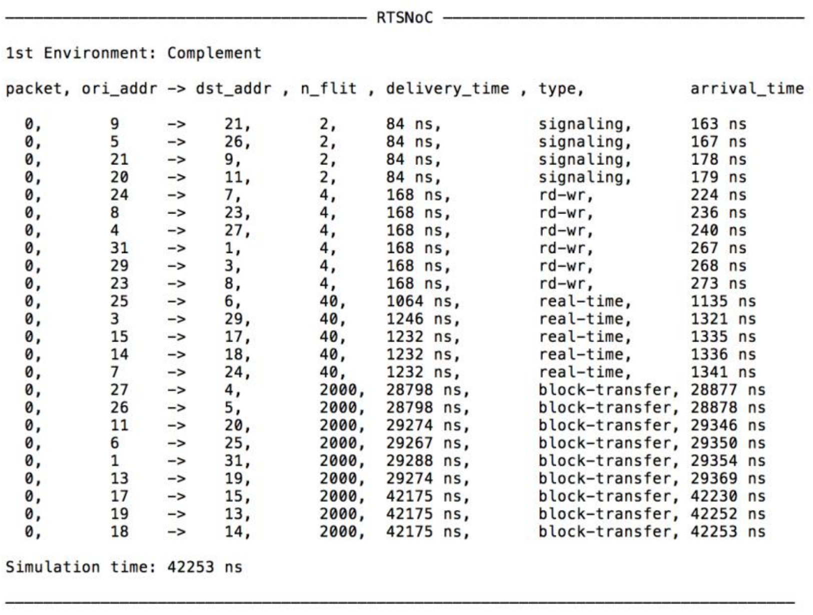
In NoCs the parameter that defines a packet length is the number of flits that form these packages. In [3] the QNoC traffic is classified into four groups based on the average number of flits of the packets, as shown in Tab. 3. Thus, the same classification was implemented on the RTSNoC's traffic generators.

Table 3: QNOC flow's classification.

Flow	**Average packet length [flits]**
Signaling	2
Real-Time	20
RD/WR	4
Block transfer	2000

IV.2 Constant model

The first evaluation scenario aims to identify the RTSNoC's behavior under those spatial distributions. The log file generated by the NM is depicted in Fig. 8 and follows the CSV pattern to facilitate data handling. The first column of that file refers to the packet identifier, followed by its flow origin and destination addresses. The third column refers to the number of payloads flits in that packet, followed by the delivery-time in nanoseconds, its type, and its arrival-time also in nanoseconds.



```
---------------------------------------- RTSNoC ----------------------------------------

1st Environment: Complement

packet, ori_addr -> dst_addr , n_flit , delivery_time , type,          arrival_time

  0,       9     ->   21,       2,     84 ns,           signaling,      163 ns
  0,       5     ->   26,       2,     84 ns,           signaling,      167 ns
  0,       21    ->   9,        2,     84 ns,           signaling,      178 ns
  0,       20    ->   11,       2,     84 ns,           signaling,      179 ns
  0,       24    ->   7,        4,     168 ns,          rd-wr,          224 ns
  0,       8     ->   23,       4,     168 ns,          rd-wr,          236 ns
  0,       4     ->   27,       4,     168 ns,          rd-wr,          240 ns
  0,       31    ->   1,        4,     168 ns,          rd-wr,          267 ns
  0,       29    ->   3,        4,     168 ns,          rd-wr,          268 ns
  0,       23    ->   8,        4,     168 ns,          rd-wr,          273 ns
  0,       25    ->   6,        40,    1064 ns,         real-time,      1135 ns
  0,       3     ->   29,       40,    1246 ns,         real-time,      1321 ns
  0,       15    ->   17,       40,    1232 ns,         real-time,      1335 ns
  0,       14    ->   18,       40,    1232 ns,         real-time,      1336 ns
  0,       7     ->   24,       40,    1232 ns,         real-time,      1341 ns
  0,       27    ->   4,        2000,  28798 ns,        block-transfer, 28877 ns
  0,       26    ->   5,        2000,  28798 ns,        block-transfer, 28878 ns
  0,       11    ->   20,       2000,  29274 ns,        block-transfer, 29346 ns
  0,       6     ->   25,       2000,  29267 ns,        block-transfer, 29350 ns
  0,       1     ->   31,       2000,  29288 ns,        block-transfer, 29354 ns
  0,       13    ->   19,       2000,  29274 ns,        block-transfer, 29369 ns
  0,       17    ->   15,       2000,  42175 ns,        block-transfer, 42230 ns
  0,       19    ->   13,       2000,  42175 ns,        block-transfer, 42252 ns
  0,       18    ->   14,       2000,  42175 ns,        block-transfer, 42253 ns

Simulation time: 42253 ns

-----------------------------------------------------------------------------------------
```

Fig.8: Log file generated by NM.

In Fig. 9-a is shown how many cycles are used by RTSNoC to switch a single flit from one router channel to another before embedding the monitoring service. In contrast, Fig. 9-b presents the waves observed after that service being included. The router used the same amount of cycles to do the same operation mentioned before, which proves that our service does not increase the NoC latency whatsoever.

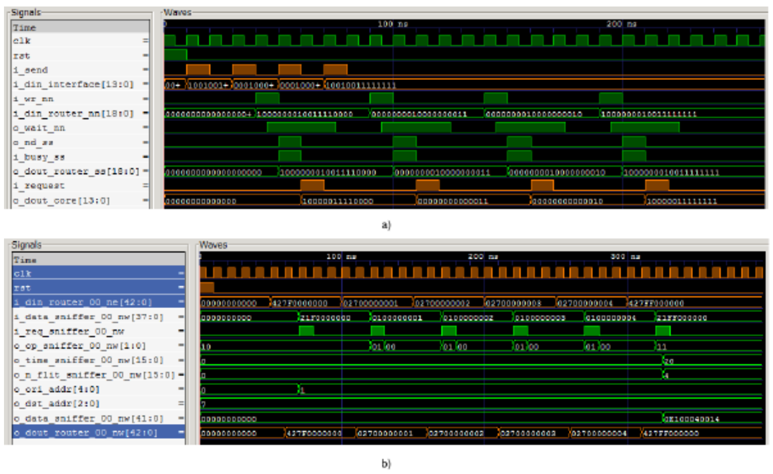


Fig.9: Waves observed on a single router before and after embedding the monitoring service.

It should be noted that the NM records the information in the log file in arrival order. As expected, due to the flits interleaving technique, smaller packets coming from critical flows were delivered earlier and in a shorter time than the other ones. In addition to the information generated by sniffers, the distribution pattern evaluated and the total simulation-time are printed in this file.

The RTSNoC has latency predictability, which is calculated by its WCL equation. Therefore, before the tests, the theoretical values were obtained for each one of the flow types. A packet can go through up to three routers, i.e., three hops. Equation (1) describes the amount of cycles a header flit needs to traverse, for example, routers 0 → 1 → 3. In this equation, N represents the number of streams competing for the same resources, equal to eight in a worst-case scenario.

$$L_{header}(i) = \sum_{j=1}^{h} 2N_j = (2 x 8) + (2 x 8) + (2 x 8) = 48 \qquad (1)$$

Equation (2) describes the number of cycles that payload flits of a block-transfer packet need to cross the same path in a worst-case scenario. The number of packets competing for the same router resources is given by k in that equation.

$$L_{payload}(i) = 2k(f-1) = 2x8(2002-1) = 32016 \qquad (2)$$

In ideal conditions where destination cores read those flits waiting at the NI immediately after their arrival being signalized without overfilling memory buffers, B in (3) is null. Otherwise, it has been found during experiments that B depends on the number of flits f, the FIFO's depth p and K, which refers to the clock cycles required for writing into the interface and for a core to read the next flit.

$$B = 2k(f-p) = 2x8x(2000-32) = 31488 \qquad (3)$$

Finally, (4) describes the total WCL for a block-transfer flow. The same thought was used to calculate the theoretical value of other flows. These values refer only to the NoC used in this experiment and might vary according to the topology used, traffic generators and different configurations. The theoretical value for the Best-Case Latency (BCL) is obtained when two farther cores communicate without having any competition.

$$WCL_{packet}(i) = \sum_{j=1}^{h} 2N_j + 2k(f-1) + 2B \qquad (4)$$

The Injection Rate (IR) used in the experiments is shown in (5). This rate relates the number of flits injected by the traffic generators according to their type, i.e., a rate equal to 100% means that all twenty-four cores are block-transfer generators, the flow that demands the most NoC resources.

$$IR = [(2000x_1 + 40x_2 + 4x_3 + 2x_4)/(24x2000)]x100 \qquad (5)$$

The average delivery-time obtained with the experiments is illustrated in Fig. 10. The first thing to be observed in this graph is its linearity, an expected behavior of a NoC that uses the flits interleaving technique, unlike others that tend to present exponential curves when injection rate is higher than 80%.

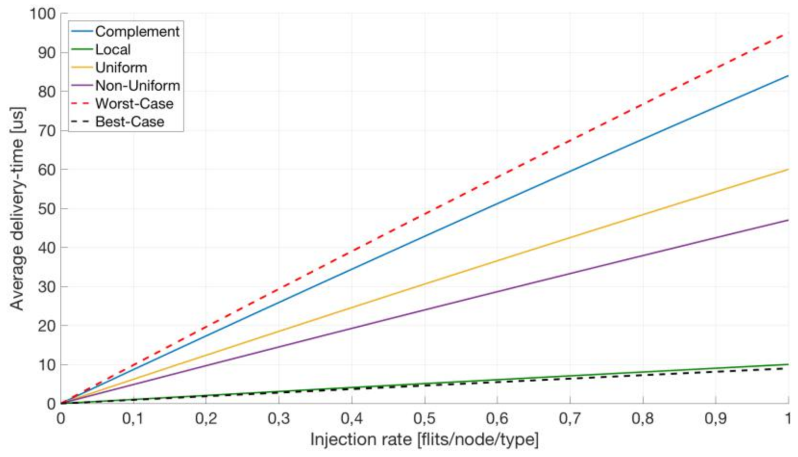


Fig.10: Delivery-time obtained for each spatial distribution.

These results could be compared with Fig. 11 presented by [7], which relates the average latency obtained with SoCIN [6] using the same patterns. This graph shows its curves starting to be exponential at even lower rates. However, the traffic meters used on SoCIN were not embedded in its network design and their architecture was not specified. Other related work discussed in Section I did not detail their project specifications, such as clock frequency, interfacing, or evaluated their NoCs under the same traffic patterns which makes it difficult to compare their results with those obtained by our monitoring service.

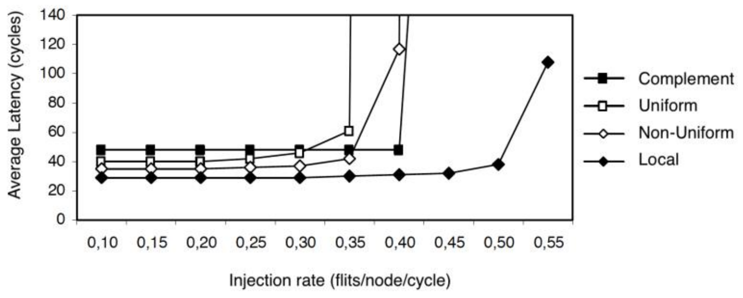


Fig.11: Average latency registered on SoCIN [7].

The spatial distribution that presented the best results was Local since in this pattern there is no communication between adjacent routers. The Complement distribution presented performance close to the WCL theoretical curve. It is important to notice that the Non-Uniform distribution has greater locality than Uniform, i.e., cores make more exchanges with its neighbors than with those cores connected into farther routers. It explains why the pattern of Non-Uniform distribution had better results than the uniform.

The linear programming branch focus on solving either maximizing or minimizing problems restrained by an objective function that satisfies a certain number of constraints. Graphically, the region delimited by WCL and BCL thresholds is called feasible and due to its linearity optimization methods could be applied to the results to approximate them into an optimum value. Re-configurable computing could also be used in conjunction with linear programming to operate the cores redistribution.

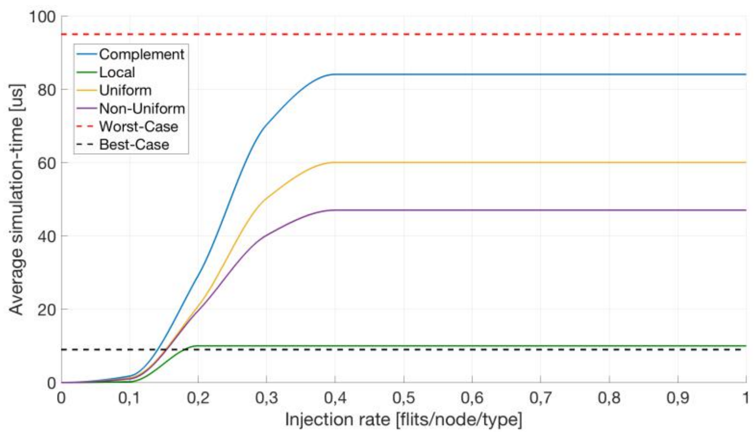


Fig.12: Simulation-time obtained for each spatial distribution.

The constant model of traffic generation allows the total simulation-time measurement, as shown in Fig. 12. Note that curves of all patterns converge to a constant value from a given injection rate. Unlike Best-Effort NoCs where results are exponential with increasing injection rate, it is expected that a NoC with flits interleaving attempts to soften the latency and converge the results into a constant value, as proven by the simulation-time graph. Also as expected, the Local distribution demanded shorter simulation-time and converged at lower rates than the others.

IV.3 ON-OFF model

The second evaluation scenario aims to evaluate if the same properties verified in the constant model of traffic generation are observable in more dynamic environments. The ON-OFF Model was chosen as a representation of real multimedia systems by varying packets' length throughout the simulation. Packets were randomly generated as shown in Fig. 13, by varying t_{packet} and t_{slack} which respectively refer to its length and to the interval between two packets.

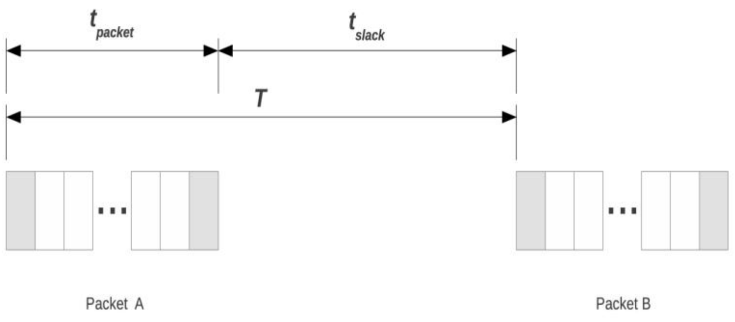


Fig.13: Random packet generation.

The graph illustrated in Fig. 14 relates the packets' length generated and the average delivery-time obtained with the Complement distribution. Its results show that in fact, the RTSNoC latency is linear and proportional to the packet length since the average delivery-time curve of those packets reaches peaks similar to the histogram of their length.

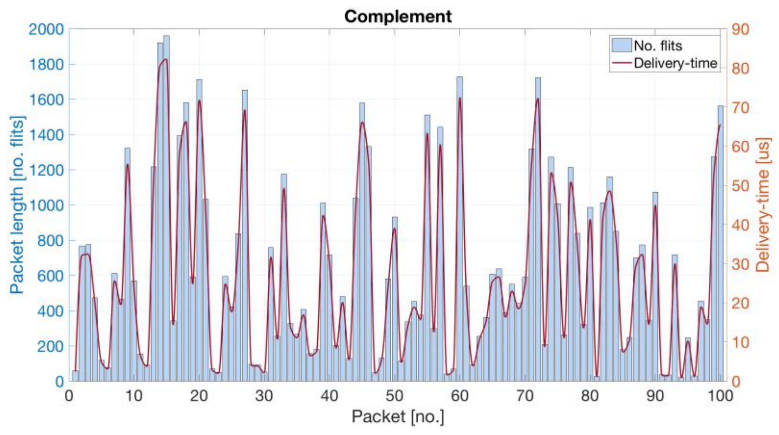


Fig.14: Results obtained with variable packets in the Complement spatial distribution.

Results obtained with the Local distribution are related in Fig. 15 in the same way as the distribution presented before. As expected, this pattern had a shorter average delivery-time than the Complement distribution.

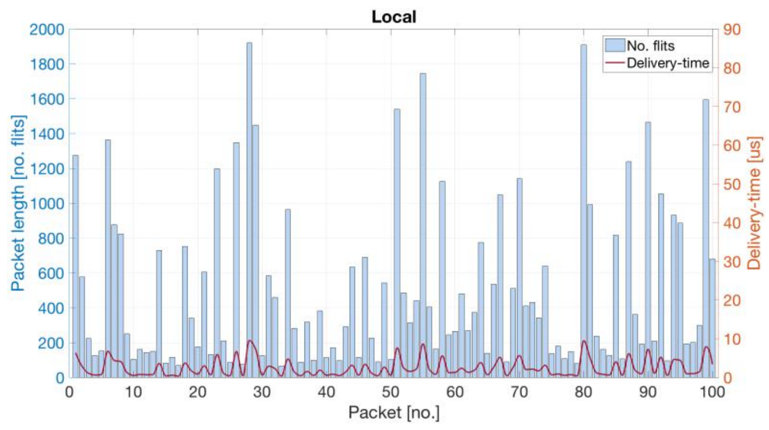


Fig.15: Results obtained with variable packets in the Local spatial distribution.

Respectively, Fig. 16 and Fig. 17 depict the results obtained with the Uniform and Non-Uniform distributions that presented an average performance on constant traffic experiments. The average delivery-time curve had not had the same peaks as the packet length histogram at certain times in the Non-Uniform distribution. This result demonstrates effectively its behavior, in which the probability of a core being destination varies during execution with times when the locality is stronger, which may cause the delivery-time drop.

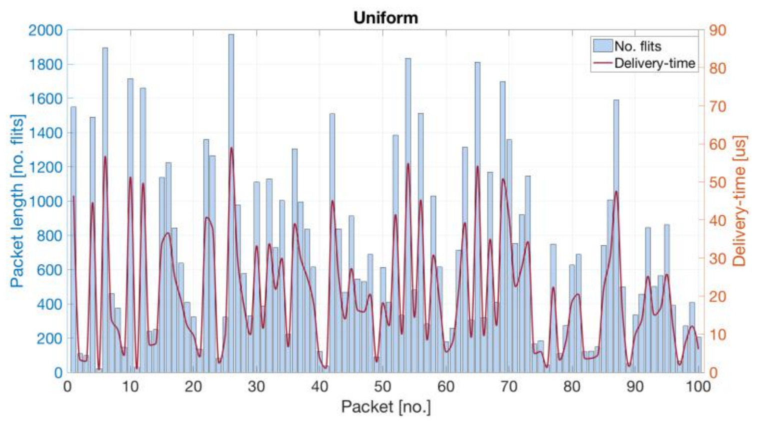


Fig.16: Results obtained with variable packets in the Uniform spatial distribution.

Randomly generate packets means that there is no correlation between subsequent samples. One way to determine whether the samples from a data-set are correlated is by their correlation coefficient analysis [16]. It measures the intensity of the relationship between the samples of a data-set and may vary between -1 and +1. When zero or near zero the samples have no relation, which means that a coefficient near one or minus one would characterize perfectly correlated samples. Tab. 4 shows the coefficient obtained for each distribution compared to a sinusoidal signal that has a high correlation. It can be said that the four data-sets have no relation, i.e., they are random.

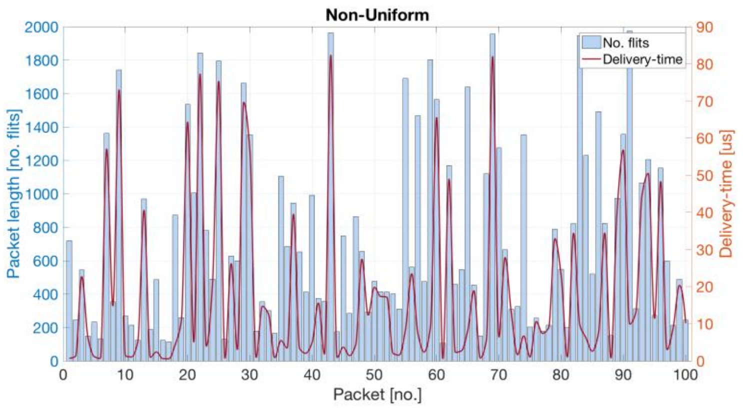


Fig.17: Results obtained with variable packets in the Non-Uniform spatial distribution.

A stochastic process is a sequence of random variables indexed by some element T, i.e., it is nothing more than a collection of random variables describing the behavior of some process over time [16]. In a random traffic scenario, the adaptability could be done statistically, through the stochastic processes control.

Table 4: Correlation Coefficient.

Signal	Coefficient	Correlation
f(x) = cos(x)	0.995	High
Fig. 14	-0.042	Low
Fig. 15	0.081	Low
Fig. 16	-0.164	Low
Fig. 17	0.025	Low

Two characteristics would allow an algorithm to control a stochastic process: its periodicity in both regular and discrete time; the fact that it evolves from one state to another depending only on its last state, which might cause that after a long execution period the results of a particular state do not change. Therefore, the monitoring service presented in this paper could be used as a means to verify this last property in random traffic for an adaptive technique.

V. CONCLUSION

This paper presented a completely non-intrusive packet delivery monitoring service that uses its resources to transport, store and evaluate the traffic information obtained without interfering on routers operation. This service was embedded into the RTSNoC's project, a network-on-chip with WCL predictability that aims real-time systems which proved to guarantee a low delivery time for flows composed by small packets.

Experiments based on traffic generation models not only validated the proposed approach for monitoring a NoC traffic but also showed that in fact the cores spatial distribution and their packet length are metrics of more significant influence on RTSNoC latency increasing. In addition, the log generated by the Network Manager also demonstrated that the flits interleaving algorithm adopted in arbitration guarantees QoS.

RTSNoC is an efficient alternative of interconnection between processing elements in applications that present a large number of small packet flows. However, systems whose adopt this NoC architecture but have their flows mostly characterized as best-effort, e.g., multimedia applications, will have an average latency for all of them. The monitoring service presented in this paper could be used as a way to provide the necessary information needed by optimization methods, re-configurable computing techniques, load redistribution algorithms, and stochastic process control, all of which aim to improve the average latency of best-effort flows.

Lastly, it is important to highlight that by virtue of our main goal all components used in our experiments and the RTSNoC itself were implemented in SystemC due to its enforcement on performance exploration. Measurement of silicon area consumption with this particular hardware description language it is not only fairly laborious but it is also not efficient. Thus, possibilities of future work for our paper include the project conversion into another language, such as VHDL, which easily allows quantifying the overhead imposed by our monitoring service in terms of extra area needed. Furthermore, the mitigation in practical settings and application of those improvement techniques described before are also prospects of future work.

REFERENCES

[1] M. D. Berejuck & A. A. Fröhlich (2016). "Evaluation of a connectionless technique for system-on-chip interconnection". Journal of Circuits, Systems and Computers 25(10).

[2] F. Moraes, N. Calazans, A. Mello, L. Möller, & L. Ost (2004). "Hermes: an infrastructure for low area overhead packet-switching networks on chip". INTEGRATION, the VLSI journal, 38(1), pp. 69–93.

[3] E. Bolotin, I. Cidon, R. Ginosar, & A. Kolodny (2004). "Qnoc: Qos architecture and design process for network on chip". Journal of systems architecture, 50(2), pp. 105–128, 2004.

[4] D. Wiklund & D. Liu (2003). "Socbus: switched network on chip for hard real time embedded systems". Proceedings International Parallel and Distributed Symposium, pp. 8 pp.–.

[5] S. Sathe, D. Wiklund, & D. Liu (2004). "Design of a guaranteed throughput router for on-chip networks". Proceedings 2004 International Symposium on System-on-Chip 2004, pp. 25–28.

[6] C. A. Zeferino & A. A. Susin (2003). "Socin: a parametric and scalable network-on-chip". Proceedings 16th Symposium on Integrated Circuits and Systems Design, SBCCI 2003, pp. 169–174.

[7] C. A. Zeferino, J. V. Bruch, T. F. Pereira, M. E. Kreutz, & A. A. Susin (2007). "Avaliação de desempenho de rede-em-chip modelada em systemc". Proceedings of the 27rd Congress of Brazilian Computer Society-WPerformance, 2007, pp. 559–578.

[8] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, & L. Benini (2003). "Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor socs". Proceedings 21st International Conference on Computer Design, Oct 2003, pp. 536–539.

[9] D. Bertozzi & L. Benini (2004). "Xpipes: a network-on-chip architecture for gigascale systems-on-chip". IEEE Circuits and Systems Magazine,4(2), pp. 18–31.

[10] K. Goossens, J. Dielissen, & A. Radulescu (2005). "Aethereal network on chip: concepts, architectures, and implementations". IEEE Design Test of Computers, 22(5), pp. 414–421.

[11] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, & J. V. Meerbergen (2005). "An event-based monitoring service for networks on chip". ACM Transactions on Design Automation of Electronic Systems (TODAES), 10(4), pp. 702–723.

[12] Z. Ghaderi, A. Alqahtani, & N. Bagherzadeh (2017). "Online monitoring and adaptive routing for aging mitigation in nocs". Proceedings Design, Automation Test in Europe Conference Exhibition (DATE), March 2017, pp. 67–72.

[13] M. Dehyadgari, M. Nickray, A. Afzali-kusha, & Z. Navabi (2005). "Evaluation of pseudo adaptive xy routing using an object-oriented model for noc". Proceedings 2005 International Conference on Microelectronics, Dec 2005.

[14] G. Martin (2003). "Systemc: from language to applications, from tools to methodologies". Proceedings 16th Symposium on Integrated Circuits and Systems Design, SBCCI 2003, pp. 3–9.

[15] J. Duato, S. Yalamanchili, & L. M. Ni (2003). Interconnection networks: an engineering approach. Morgan Kaufmann.

[16] G. James, D. Witten, T. Hastie, & R. Tibshirani (2013). An introduction to statistical learning. Springer, vol. 112.