RTSNoC: a predictive Network-on-Chip for real-time applications

Marcelo D. Berejuck , Antônio A. Fröhlich, Tiago R. Mück
Federal University of Santa Catarina
Florianópolis, Brazil
{berejuck,guto,tiago}@lisha.ufsc.br

Hiren D. Patel
University of Waterloo
Waterloo, Canada
hdpatel@uwaterloo.ca

***Abstract*—Network-on-Chips (NoCs) are becoming the main communication infrastructure for large Systems-on-Chips (SoCs). In the scope of real-time applications, determining the latency and jitter of on-chip communication is a requirement for fixing the worst case execution time (WCET) of the system. This paper addresses this issue and presents RTSNoC. RTSNoC uses a connectionless technique in which it is possible to predict the maximum latency of any data flow. The chosen approach is to add routing information to all flits. An arbitration algorithm then leverages on this extra information to dynamically schedule the flits, preventing communication channel contention.**

***Index Terms*—Network-on-Chip, Real-time system, Field-programmable gate array.**

I. Introduction

Network-on-Chips (NoCs) are being increasingly used in order to meet *System-on-Chips* (SoCs) needs in terms of scalability, reusability, and to minimize electrical issues [1]. In the scope of real-time applications, a NoC must provide means to predict the maximum communication latency in order to allow the definition of the *worst case execution time* (WCET) of the system. Some NoCs address this issue by focusing on *best effort* (BE) mechanisms based on connection-oriented approaches or *time division multiplexing* (TDM) techniques [2], [3], [4], [5]. However, the use of connection-oriented techniques usually demands more silicon resources to create virtual channels. The NoC must be oversized at design time in order to support a wide range of network loads while keeping a predictable latency.

In this paper we introduce the *Real-Time Service Network-on-Chip* (RTSNoC). RTSNoC uses a connectionless technique that allows the determination of the network WCET. An RTSNoC network works as a regular 2-D mesh in which each router can connect up to eight processing cores. The chosen approach for RTSNoC is to add extra information to all flits. This information is then used by a priority-based dynamic round-robin algorithm implemented at the routers to schedule routing requests. The implemented algorithm prevents channel contention and keeps a uniform latency for all requests, while yielding a fixed maximum latency regardless of the network load.

The remaining of this paper is organized as follows: Section II presents a discussion about network implementations that exploit connection-oriented techniques; Section III presents RTSNoC architecture and its WCET analysis, along with a comparison between RTSNoC and a traditional TDM-based NoC; Section IV evaluates the impact of the proposed mechanisms in terms of silicon area, latency, and jitter; Section V presents the deployment of RTSNoC in a FPGA-based *Private Automatic Branch eXchange* (PABX) SoC; and Section VI closes the paper with our conclusions.

II. Related Work

Several NoCs that offer latency guarantees have been proposed. One of those networks is AEthereal [2]. It employs TDM techniques to create dedicated channels that provide contention-free routing and guaranteed latency bounds. On this model, the bandwidth of each link is split into a predefined number of time slots that are defined at design-time according to the traffic expected in the network. dAElite [6] uses a similar approach. The network uses a distributed routing mechanism in which each router contains a slot table to store the TDM schedule. A network that also uses contention-free routing is TTNoC [7]. According its authors, this network claims to offer more freedom than a fixed, periodic TDM table. The network also supports multicast and broadcast operation. These networks provide tight latency guarantees, however, they must be designed to closely match the network load required by the application. Different applications may require a costly redesign of the network, which hinders the flexibility and scalability of the system.

Nostrum [8] uses a different approach and does not require a fixed TDM schedule. The TDM period is linked to the length of looping connections. Multicast is also supported by adding more receiver nodes to a closed loop. Nostrum also offers BE communication using deflection routing. The disadvantage of Nostrum is that routing paths, and consequently multicast node sets, also must be decided at design time.

Some networks provide explicit *Quality-of-Service* (QoS) mechanisms. QNoC [3] splits flows into different categories, assigning different priorities for each one. Other networks, like SoCIN [4] and Hermes [5], were originally designed as BE networks and latter extended with QoS mechanisms. Hermes introduced a virtual channel mechanism, similar to the one proposed by [2]. SoCIN incorporated a packet aging mechanism in which a packet receives an initial priority level when injected into the network. Increments in the priority level will occur according its permanence in the network [9].

On a different approach [10], the authors claim that it is difficult to estimate the communication delay of NoC-based systems due to their complexity, making implementation of real-time systems on NoCs virtually impossible. They proposed a method for communication modeling and synthesis of dynamically reconfigurable NoC-based systems that enable contention-aware scheduling. The network model is defined at system-level, and the method to prevent against contention is an extension of the edge scheduling approach used in distributed systems [11].

III. THE RTSNoC ARCHITECTURE

The basic building block of an RTSNoC network is the router. The RTSNoC router has eight interconnection points, as shown in Figure 1. In order to facilitate their identification, the eight points receive the names of cardinal direction points from the compass: north (NN), northeast (NE), east (EE), southeast (SE), south (SS), southwest (SW), west (WW), and northwest (NW).

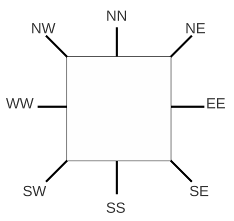

Figure 1. RTSNoC router. Each corner is a point of connection to the network.

Each router communication point provides two opposite unidirectional channels, as shown in Figure 2. The size of each channel may be configured by the user, according to the application needs. The signal bus called DIN and DOUT refers to input and output data bus, respectively, while the signal RD and WR are strobes used to read and write data. The signals WAIT and ND are used for flow control. WAIT inform that the core must wait before writing new data in the router input channel, while the signal ND informs the destination that a new packet is available.

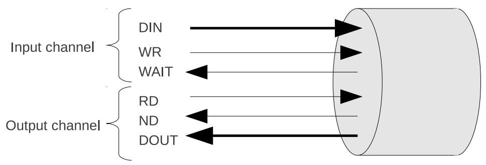

Figure 2. Communication channel signals

Communication channels may be connected to processing cores or to other routers in order to build a 2D regular mesh network. Figure 3 illustrates a generic system implemented using the RTSNoC. Four routers are used to establish a 2x2 2-D regular mesh network that can interconnect up to 24 cores.

Figure 4 shows the internal router structure. Each communication channel has an input interface, an output interface, and a flow controller. The input interface has a register that may store a flit. When a core needs to send a packet through the network, it writes the packet in this register. Then, the flow controller checks the packet header for the destination and

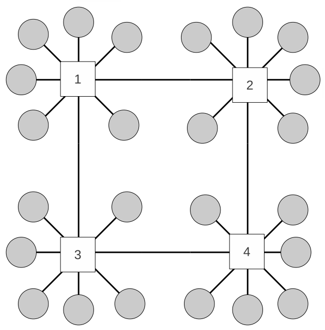

Figure 3. A regular mesh NoC with 4 routers interconnecting and 24 cores

verifies if the destination is available to receive a new packet. At the same time, the flow controller informs the arbiter about a new routing request.

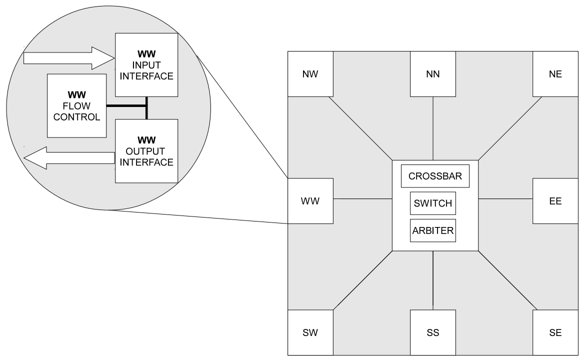

Figure 4. Block diagram showing the internal structure of the router

The *arbiter* implements a dynamic scheduling algorithm, as shown in Figure 5. Each channel receives a different priority level when the system starts. The highest priorities are given to the channels NN, SS, EE, and WW, since they are used to interconnect other routers in a 2-D regular mesh network. Any core has its routing request attended if it has the highest priority, or if there are no other requests in the arbiter. Once the request is attended, the channel that requested the sending of a packet receives the lowest routing priority level and may only send other packet if there is no other packet waiting for routing. Once the channel that has the routing priority is chosen, the arbiter sends a command to a logical block called *switch*, informing which routing has to be executed.

To perform the routing algorithm, NoCs use a minimum set of control data containing information such as the source and destination address of a packet. Most NoCs use packets composed by several flits. Some of these flits are chosen to carry routing information or to inform the end of packet, and the remaining consists in the actual payload. The network proposed in this document uses a single flit to store both payload and the routing information. That is, each set of payload bits of a data flow has appended its source and destination addresses. This approach has an initial impact on silicon area since it requires a wider data bus for each channel. However, it allows the individual scheduling of each packet

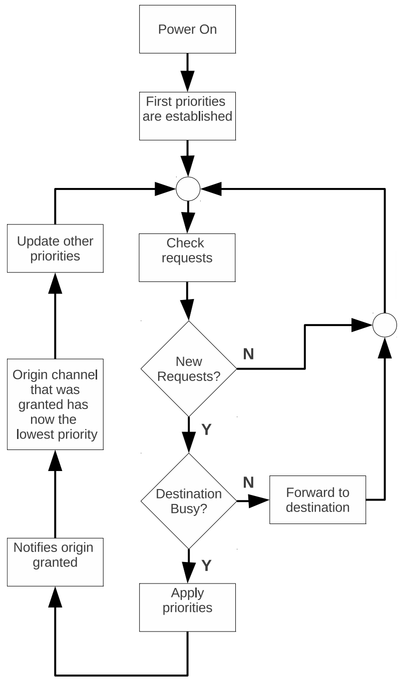

Figure 5. Flowchart showing the dynamic scheduling algorithm implemented in RTSNoC

and prevents network contention. Figure 6 shows the format of these packets with one flit.

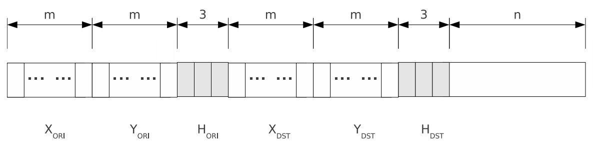

Figure 6. RTSNoC packet format. The size of X/Y fields depends on the topology.

Two fields called X_{ORI} and Y_{ORI} inform the X and Y coordinates of the network router where the packet was generated (i.e. the origin address of the packet). The same happens to the destination router that have the fields called X_{DST} and Y_{DST} to indicate the destination coordinates. The field H_{ORI} refers to the local port address inside the router where the packet has been sent. Similarly, H_{DST} is the information of the destination core. The last field shown in Figure 6 is the payload.

A. Static WCET analysis

In an RTSNoC network composed by several routers, as shown in Figure 3, the worst case scenario for a transmission request is defined as follows:

1) the request has the lowest priority at that moment and competes with N requests in the same router, for the same output channel;
2) when the request reaches the next router, it finds the same situation described in 1).

In this scenario, the maximum latency in a generic RTSNoC is defined by the following equations:

$$L_{rou} = 6 * N_{req} * C_{clk} \tag{1}$$

$$L_{max} = \sum_{i=1}^{N_{rou}} (L_{rou})_i \tag{2}$$

where L_{rou} is the maximum router latency, N_{req} is the number of input channels requesting for the same output channels, and C_{clk} is the clock period. These values are then multiplied by a constant that represent the number of cycles required for the routing algorithm. Finally, with N_{rou} routers between the source and destination, L_{max} gives the maximum latency of the network. Due to functional properties of RTSNoC routers, there is no need to include in Equation 1 information about payload size.

In the realm of real-time theory, it is common to use the *processor utilization factor* to verify if a set of periodic tasks can be scheduled:

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \tag{3}$$

where C_i is the processor time of a task τ_i and T_i is its period. The fraction of processor time required for the execution of a set of N tasks is given by U. Considering a system in which a task must exchange data with another nodes, Equation 3 can be redefined to include the network latency:

$$U = \sum_{i=1}^{N} \frac{C_i + \sum_{j=1}^{M} Q_j * L_j + \sum_{k=1}^{O} Q_k * L_k}{T_i} \tag{4}$$

where Q_j and Q_k are the amount of packets to be transmitted and received, respectively, between the node τ_i is running on and the remote nodes $_j$ and $_k$. L_j and L_k are the maximum latency given by Equation 2. For simplicity, it is assumed that the processor time of nodes $_j$ and $_k$ is null.

B. RTSNoC X contention-based TDM

In order to highlight the difference between RTSNoC and the previous NoCs that use contention-based TDM approaches, we describe in more details the SoCIN-TDM network [4], and the impact of its mechanism in terms of WCET. The SoCIN-TDM network implements TDM links that provides four channels between two routers. The cores connected to local ports may choose which channel will transfer its packets. Bits in the SoCIN header flit are used to do this choice.

SoCIN-TDM has the same problem discussed in Section II: the TDM schema must be redefined at design time to match different network loads. The TDM schedule is not fully parameterized, thus, SoCIN-TDM allows communication without contention for a network in which the number of hops is less than or equal to four. SoCIN accepts two routing algorithms: XY and West-First. Figure 7 shows a SoCIN-TDM network that uses the XY routing algorithm. If the cores A, B, C, D, E and F send packets to the core connected to router *(4,2)*, only four of them will receive TDM channels. In Figure 7, the cores A, B, C and D receives TDM channels, while the cores E and F must wait indefinitely, thus no bounds for WCET can be defined. The contention problem may be solved by redesigning the network with more TDM channels, however, this approach expends more silicon resources and increase the total system design time.

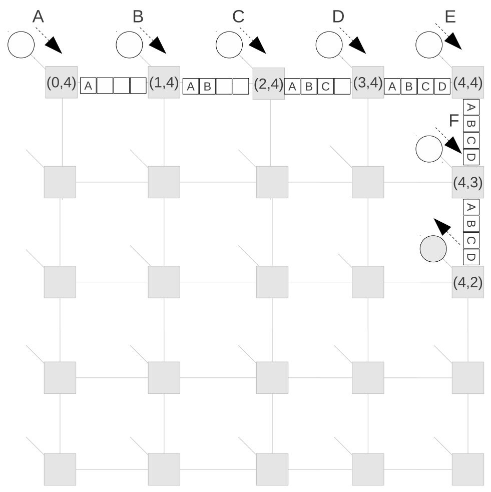

Figure 7. Example of a SoCIN-TDM network with contention. The links between routers are shared between four TDM channels

IV. Performance evaluation

In this section we evaluate RTSNoC in terms of silicon consumption, latency, and jitter. The obtained results are compared with the ones of the SoCIN-TDM network [4] in order to assess how RTSNoC performs in relation to a classical contention-based TDM NoC. The implementation and validation of RTSNoC network was performed using Xilinx's ISE version 13.1 targeting FPGAs from the Virtex 6 family.

A. Area

To evaluate the resource consumption in the scope of a complete SoC, we have chosen as reference the SoC proposed by [12]. It is a high performance 4G modem used in mobile telecommunications and implements a *Multi-Carrier Code Division Multiple Access* (MC-CDMA) system. Figure 8 presents this application scenario.

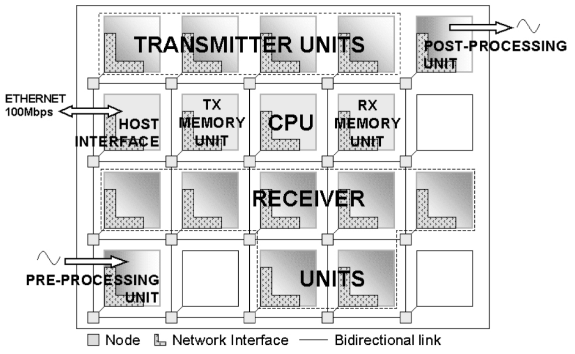

Figure 8. The MC-CDMA implemented using a NoC with a regular 2-D mesh topology [12]

Figure 9 summarizes the results. Considering the area of a single router, the RTSNoC shows an apparent high resource consumption. This is due to the number of cores supported by each router. While RTSNoC supports up to eight cores, SoCIN supports only a single core. Also, the adoption of packets with several control bits yields a slight increase in silicon area, since channels will need more communication lanes.

For the SoC evaluation, the system was implemented using seventeen different cores. In the SoCIN-TDM network twenty routers were necessary to implement the system, since the network requires a regular mesh topology and each router supports a single processing element. On the other hand, the RTSNoC requires only three routers. RTSNoC may be considered a lower silicon cost alternative to implement real SoCs, when it is compared to the SoCIN-TDM network. The use of communication channels to connect more than one processing element compensates the increase of silicon area due to a bigger number of bits used in the flit definition.

NoC	FPGA resources		
	SLICEs	LUTs	FFs
SoCIN-TDM (router)	402	772	372
RTSNoC (router)	807	429	1131
SoCIN-TDM (SoC)	8040	15440	7440
RTSNoC (SoC)	2421	1287	3393

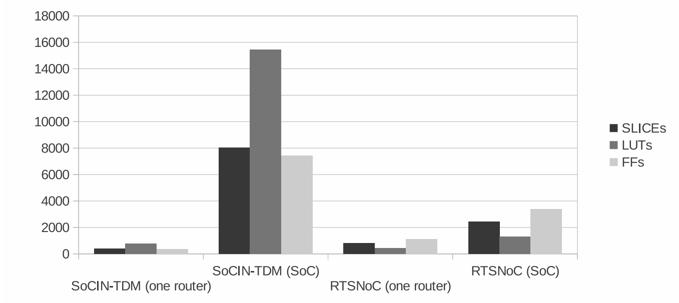

Figure 9. FPGA resource consumption for a single router and SoC based on the implementation of a 4G modem SoC

B. Latency and jitter

The performance of a NoC is usually given by its latency. However, another important criterion for real-time applications is jitter. With small jitter it is possible to establish lower bounds on the WCET of the system. To evaluate these metrics, four networks, composed by routers from SoCIN-TDM network and RTSNoC network were built and split in two scenarios, as shown in Figure 10. *Traffic generators* (TG) cores generate random amounts of packets at random time periods. All packets are sent to a *traffic measurement* (TM) core which measures the latency and jitter for each TG.

Thirty consecutive measurements of latency were done for different fractions of the total traffic capacity (20, 40, 60, 80 and 100%). Figure 11 shows the results. The SoCIN-TDM network keeps the latency constant for the scenario I since it was designed to offer constant latency (the number of hops is less than the number of TDM channels for that network

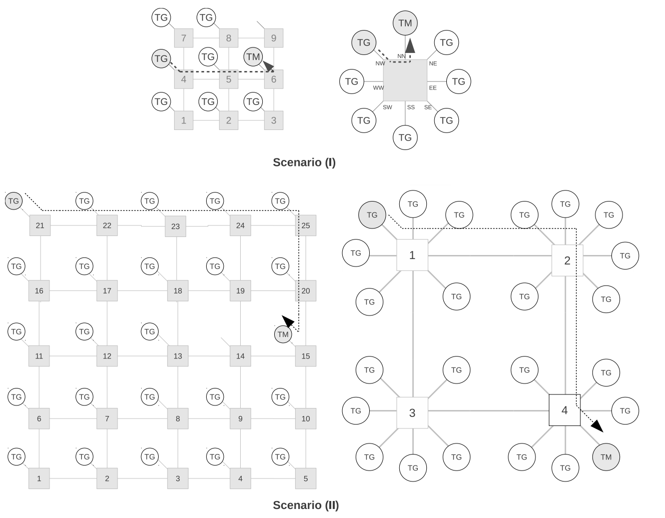

Figure 10. Networks used for latency and jitter evaluation: (A) SoCIN-TDM with 7 cores; (B) RTSNoC with 7 cores, (C) SoCIN with 23 cores, and (D) RTSNoC with 23 cores.

size). When the network is increased to receive new cores as shown in scenario II, the limited number of available TDM channel causes contention, resulting in a significant increase in latency. The RTSNoC, on the other hand, keeps the latency under control even at 100% of network utilization. RTSNoC performs a dynamic scheduling of the flows that are competing for some communication channel, meaning that all flows may have access to all channels over time.

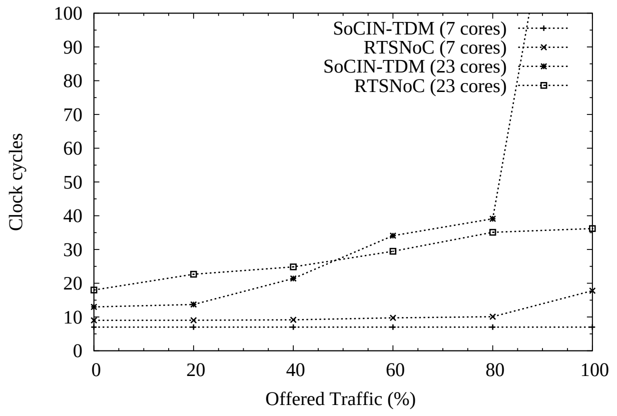

Figure 11. Average latency *vs.* offered traffic for SoCIN-TDM and RTSNoC networks.

Figure 12 shows the average jitter values. As expected for scenario I, the SoCIN-TDM network has a constant jitter due to the availability of enough TDM channels. When the NoC connects more cores in scenario II, the jitter grow significantly. On the other hand, the jitter for RTSNoC is high under low traffic and tends to decrease as the traffic increases. In RTSNoC, with low traffic the probability of a flow being randomly preempted increases, thus increasing the jitter. In the extreme case of 100% of traffic, however, all flows are competing for the same resources full-time, keeping the scheduling policy constant. For over 80% of traffic, RTSNoC has the same null jitter presented by SoCIN-TDM in scenario I.

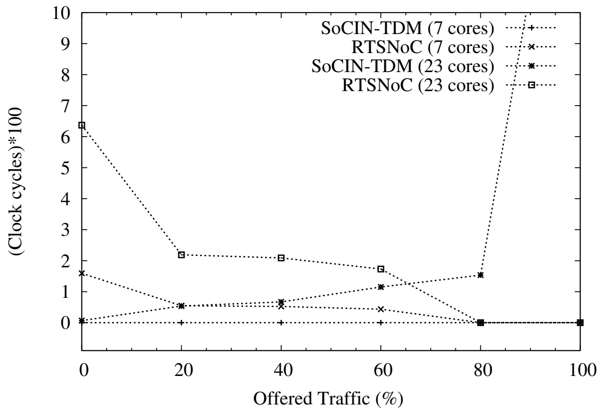

Figure 12. Jitter variance *vs.* offered traffic for SoCIN-TDM and RTSNoC networks

V. Case study

In order to evaluate the performance of RTSNoC in an industrial application with real-time constraints, we have used it as the base infrastructure of a *Private Automatic Branch eXchange* (PABX) SoC.

The PABX SoC shown in Figure 13 targets FPGA-based telecommunication systems and consists of an RTSNoC router interconnecting some telecommunications cores, providing the following minimum features to implement a digital PABX: a 425 Hz generator; a DTMF generator; a digital switch that provide conference service for voice calls; a DTMF detector; a softcore processor that works as the system controller; an interface to E1 link; and finally an interface to provide access to the PABX subscribers.

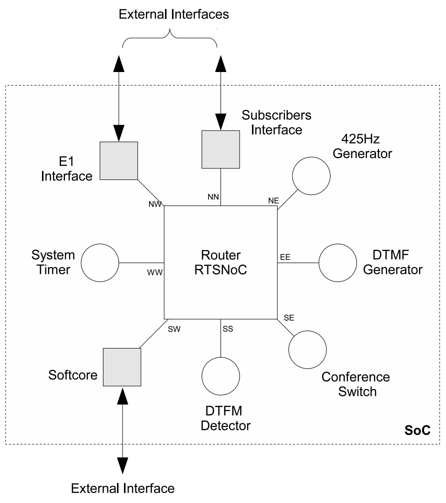

Figure 13. Block diagram showing the internal structure of the PABX SoC implemented using RTSNoC. Squares are used to represent cores with external interfaces.

The core *System Timer* generates a packet every 125 μs

which is sent to all other cores connected to the router. This packet is an indication of a synchronism pulse to the system and, due to network traffic conditions, its arrival may present jitter. However, the jitter is always between known and tolerable limits for the PABX system, since the adoption of the packet structure ensures that no single flow will retain the communication channels indefinitely.

A. Results

The SoC was synthesized using the Xilinx's ISE version 13.1 targeting an XC4VFX100-12FF1152 FPGA. The PABX system was analyzed in its maximum operation load. For this scenario, thirty two telephone calls were generated at *E1 link* and sent to *Subscribers interface*. The incoming calls were randomly generated and have short and random duration. For each incoming call, the PABX cores exchange messages related to call identification, DTMF signaling generation and detection, and call forward.

The RTSNoC has been clocked at 100 MHz, providing a maximum throughput of 133,312 $Mbps$, while the cores generate data at about 16 $Mbps$. Despite being a relatively low traffic in respect to the total network capacity, message exchanges occurs in bursts soon after the identification a synchronism message from the clock System (*System Timer* on WW port). Thus, there is intense competition for resources during a short period, creating jitter for the synchronism packets. Several measures were performed to evaluate the jitter in the synchronism packets.

Figure 14 shows the latency variation of synchronism packets received at port NW. All measured latency was between 317 ns and 288 ns, with an average value of 302 ns, matching the 420 ns WCET that can be obtained from Equation 2.

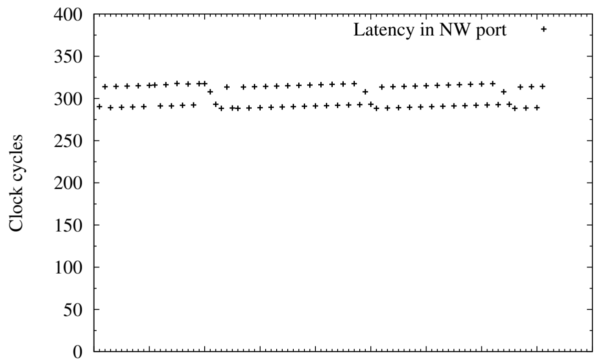

Figure 14. Latency of synchronism packets at port NW

VI. Conclusion

This paper presents a model of a Network-on-Chip with regular 2-D mesh topology. The router proposed in this work can connect up to eight cores or be connected to other routers to build a regular mesh network. We have proposed a dynamic priority-based algorithm that keeps uniform and known network latency for any network load. This is possible since all flits include routing information, which allows the preemption of any data flow and avoids network contention.

The experimental results have shown an apparent high jitter, however, in all evaluated scenarios, the latency was within the bounds defined in our WCET analysis. We have also presented the use of RTSNoC as the interconnect solution for a PABX SoC. The latency introduced by the NoC showed a well-defined behavior, allowing the creation of local logic clocks in a relatively straightforward way. Furthermore, RTSNoC showed significant differences in terms of silicon area when compared with contention-based a TDM NoC. RTSNoC has a high relative area cost when used to connect a small number of cores, however, the relative area is significantly reduced when RTSNoC is used to build large SoCs. This confirms RTSNoC as a low cost and scalable solution for the interconnection of complex SoCs for real-time applications.

VII. Acknowledgments

We would like to thank the authors of SoCIN network for providing the source code of their NoC, and Intelbras S.A. for providing the PABX hardware platform used in our evaluation.

References

[1] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[2] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Des. Test*, vol. 22, no. 5, pp. 414 – 421, Sep. 2005.

[3] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, pp. 105–128, 2004.

[4] C. A. Zeferino and A. A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip," in *Proc of the 16th Symposium on Integrated circuits and Systems design*, 2003, p. 169.

[5] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integr. VLSI J.*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[6] R. Stefan, A. Molnos, and K. Goossens, "dAElite: A TDM NoC Supporting QoS, Multicast, and Fast Connection Set-up," *IEEE Trans. on Computers*, May 2012.

[7] C. Paukovits and H. Kopetz, "Concepts of switching in the Time-Triggered Network-on-Chip," in *Proc. of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2008, pp. 120 – 129.

[8] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip," in *Proc. of the conference on Design, automation and test in Europe*, 2004, pp. 20 890–.

[9] M. D. Berejuck and C. A. Zeferino, "Adding mechanisms for QoS to a network-on-chip," in *Proc. of the 22nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes*, 2009, pp. 25:1 – 25:6.

[10] M. Tagel, P. Ellervee, T. Hollstein, and G. Jervan, "Contention aware scheduling for NoC-based real-time systems," in *NORCHIP, 2011*, nov. 2011, pp. 1 –4.

[11] O. Sinnen and L. Sousa, "Communication contention in task scheduling," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 503 – 515, june 2005.

[12] R. Lemaire, F. Clermidy, Y. Durand, D. Lattard, and A. Jerraya, "Performance evaluation of a NoC-based design for MC-CDMA telecommunications using NS-2," in *Proc. of the 16th IEEE International Workshop on Rapid System Prototyping*, june 2005, pp. 24 – 30.