On-Chip Networks for Mixed-Criticality Systems

Polydoros Petrakis*, Mohammed Abuteir†, Miltos D. Grammatikakis*, Kyprianos Papadimitriou*, Roman Obermaisser†, Zaher Owda†, Antonis Papagrigoriou*, Michael Soulie§, and Marcello Coppola§

*Technological Educational Institute of Crete, GR
†University of Siegen, DE
§STMicroelectronics, FR
*contact: {mdgramma,kpapadim}@cs.teicrete.gr

***Abstract*—We propose the integration of a network-on-chip-based MPSoC in mixed-criticality systems, i.e. systems running applications with different criticality levels in terms of completing their execution within predefined time limits. An MPSoC contains tiles that can be either CPUs or memories, and we connect them with an instance of a customizable point-to-point interconnect from STMicroelectronics called STNoC. We explore whether the on-chip network capacity is sufficient for meeting the deadlines of external high critical workloads, and at the same time for serving less critical workloads that are generated internally. To evaluate the on-chip network we vary its configuration parameters, such as the link-width, and the Quality-of-Service (QoS), in specific the number (1 or 2) and type (high or low priority) of virtual channels (VCs), and the relative priority of packets from different flows sharing the same VC.**

I. Introduction

With the advent of multicore processors residing on the same chip (MPSoC) several efforts exist towards integrating systems with different levels of importance, safety and dependability, known as mixed-criticality systems integration [1], [2]. Certification of such systems is challenging because concurrently executed applications with different criticality levels can block each other when accessing shared resources [3]. Furthermore, a few recent efforts started studying on-chip networks to explicitly support mixed-criticality systems [4], [5], [6]. Some propose new on-chip networks, and others augment existing ones with certain features. Previous studies to this direction researched the extend to which guaranteed-performance and best-effort traffic can be served by a NoC [7]. The authors of [8] discussed adaptation of NoCs to real-time requirements, and showed that for soft real-time systems the number of missed deadlines can be reduced by utilizing a routing mechanism based on message priorities. We complement these works by searching the set of parameters of a certain NoC that suit well for mixed-criticality systems.

We consider that external hard real-time messages enter the MPSoC through a gateway, which in-turn forwards them to the tiles, while at the same time less critical workload is produced by independent processes running in the on-chip CPUs. Real-time tasks are transferred in the form of Time-Triggered Ethernet (TTE) messages [9] entering the MPSoC, in which tiles are interconnected via the Spidergon STNoC [10]. By definition, a mixed-criticality system should not sacrifice lower criticality applications for whatever purpose [11], and it is desirable to serve them at competitive soft real-time rates that can be specified at design time by a service-level agreement. Our main contributions are:

- we reveal the cases showing saturation of the NoC and noticeable variations in packet delay;
- we identify a proper set of NoC parameters by varying link-width and QoS that balances cost with performance;
- a worst case timing analysis of the transfer of packets throughout the NoC.

In the next Section we provide some preliminaries, and in Section III we present the experimental setup. Section IV discusses the application mapping and scenarios, and the NoC parameters we varied. Section V has our results and findings.

II. Preliminaries

This Section has clarifications on the fields related with our work: we briefly describe real-time systems, then time-triggered communication technology, and finally we give details of the Spidergon on-chip communication network.

A. Real-Time Systems

Real-time tasks are distinguished in four categories depending on their arrival pattern and deadlines. If meeting a given task's deadline, i.e. time point a task must have been completed, is critical for the system operation, the deadline is considered to be hard. If it is desirable to meet the deadline but in some cases missing it can be afforded, the deadline is termed soft. Tasks with regular arrival times are called periodic. On the other hand, tasks with irregular arrival times are called aperiodic, and typically have soft deadlines. Also, there exist aperiodic tasks that have hard deadlines, termed sporadic tasks.

B. Time-Triggered Ethernet

Time-Triggered Ethernet [9] or TTE emerges as one of the dominant technologies for integrating high critical and low critical traffic into a homogeneous backbone, and stands as a suitable replacement of bus-systems in mixed-criticality systems. It offers three classes of traffic: static time-triggered (TT) traffic that has the highest priority, and dynamic traffic, which is subdivided into Rate Constrained (RC) traffic that has bounded end-to-end latencies, and Best-Effort (BE) traffic for which no timing guarantees are given. TTE networks are composed of one or more clusters connected via gateways, and each cluster groups end-nodes and network-switches interconnected with full duplex physical links. Data are transmitted

in the form of frames; messages are transferred in the frame payload. TTE frames follow the previous naming convention: TT frames are transmitted based on offline computed schedules and are periodic, RC frames have a minimum inter-arrival time and are sporadic, and BE frames are aperiodic.

The TTE network is partitioned into communication channels with a predefined link bandwidth and scheduling time, using the concept of virtual links (VL). Virtual links are tree structures with one sender and one or multiple receivers that allow to emulate point-to-point connectivity. In terms of implementation, this is done using a virtual link identifier (VLID) transferred into the frame. The VLID is used to differentiate between the different traffic classes, by a table lookup in a device. Network-switches route packets based on their VLID, and it is possible that different switches change the traffic class of a frame, e.g. a frame sent as TT to a switch, can then be tagged as RC and forwarded accordingly [9].

Further details on TTE technology are out of our scope, as we use it as the external source only for feeding the on-chip network with real-time messages, and we could use other technologies such as CAN bus or FlexRay [12].

C. Spidergon on-chip Communication

Spidergon STNoC is a ring-based topology comprising three main building ingredients; network interface (NI) that functions as access point to the interconnect onto which initiators and targets are directly connected; simple non-programmable router; and physical link. STNoC relies on network operation, thus it provides programmable services such as changing the routing information and Quality-of-Service (QoS). It can be customized according to given specifications to efficiently interconnect MPSoC components, i.e. CPUs, memories, and peripherals. From the network point of view, each component is a node (called also tile), and information - instructions and data - is transmitted across the network in packets. Each packet carries a header information that allows the receiver to reconstruct the original message. Typically, packets sent from a certain tile follow always the same path. This is determined with a source-based static routing policy, which functions based on the information contained in the packet header.

Prior to transmitting a packet, this is broken down into smaller units called flits that in turn are released into the network. A flit carries the maximum amount of data that can traverse the physical link per transaction. Thus, for a given packet size, the amount of flits comprising each packet depends on the link-width. During customization, along with the topology, the NoC designer tunes-up these parameters, i.e. link-width and packet size. For example, for an STNoC with 16 $Bytes$ link-width, if packet size is 72 $Bytes$, e.g. 64 for payload and 8 for header, this is split up in 5 $flits$.

Each router is equipped with buffer(s) that store temporarily the flits, prior to forwarding them to the next router. A buffer implements essentially the virtual channel (VC) on which the packets are transferred. Routers can be configured with up to 2 buffers each, thus 2 VCs, each one with a different priority, i.e. high- or low- priority. This comes at the expense of additional hardware cost, but it allows to simultaneously store packets from different flows onto the same router, which accounts for congestion reduction. Different traffic policies can be set depending on the number of available VCs; this relates to the Quality-of-Service of STNoC.

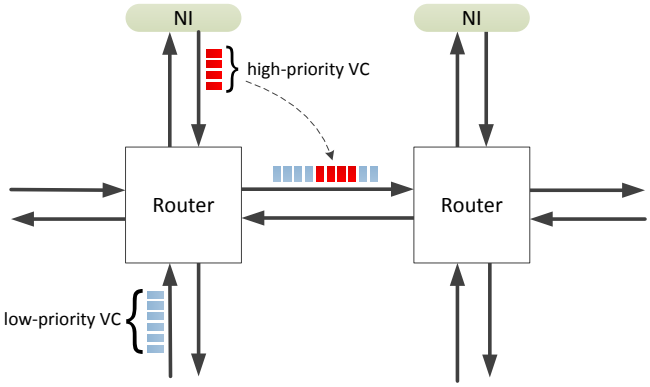

Fig. 1: Packets from different VCs transferred in an interleaving manner. Packets using the high-priority VC are served immediately, which can result in preempting packets being transferred on the low-priority VC.

Figure 1 shows that the presence of high priority VC allows preempting packets transferred on the low priority VC, by permitting packet interleaving. We consider this feature as important for delivering hard deadline packets, as their transfer time on the STNoC can be pre-calculated; ideally it is equal to the STNoC latency. Another traffic policy is that of assigning priorities for the different flows sharing the same VC. This can be done with the QoS Fair Bandwidth Allocation (FBA), a low cost arbitration algorithm that distributes the NI target bandwidth among different initiators during peak request time. This algorithm is enabled at the garnet router and allows to manage packets from different flows that compete for the same output port of the router.

Recapitulating, the different STNoC configurations offer a wide range of choices, but they are related to cost. On-chip networks with 8 $Bytes$ link-widths cost much less than 32 $Bytes$, and 1 VC costs less than 2 VCs, thus selecting wisely the parameters prevents from increasing unnecessarily the hardware cost. This can be huge, considering that existing SoC products using STNoC such as the STiH312 Cannes Ultra platform interconnect up to 148 tiles [13].

III. Experimental Setup

This Section overviews the setup on which we evaluated the NoC in serving mixed-criticality systems.

A. System Model

Three TTE nodes communicate over a TTE network via a switch. One of the TEE nodes is the STNoC that interconnects CPU and memory tiles. TTE messages sent by other nodes are forwarded by the switch, then enter the STNoC through an on-chip gateway, converted through a protocol into STNoC packets, and subsequently relayed to the on-chip tiles. We modeled this system within an environment that mixes commercial and academic frameworks. We use a simulation framework for the

TTE published in [14] to generate the TTE messages, which then feed an extended version of gem5 simulator implementing the STNoC-based MPSoC [15]. The frameworks communicate via files, in which actions are annotated with time-stamps and other related parameters, i.e. ID/type of message, message size, VLID, target tile, arrival time, and deadline.

B. Spidergon STNoC Node

We target the MPSoC of Figure 3 that interconnects four CPUs and four memories via a Spidergon STNoC. The clock frequency of the on-chip network is $1GHz$, thus once a packet transmission begins, each flit is injected from the NI every $1ns$. With regard to the delay added by each component participating in a complete transaction: the latency of the protocol at the initiator for converting an incoming message into an STNoC packet, either from the gateway or a CPU, is 5 $clock\ cycles$; the NI that breaks the packet into flits adds a latency of 1 $clock\ cycle$, and the garnet router along with the physical link add a total latency of 2 $clock\ cycles$. Therefore, if for instance a packet passes via two routers, 10 $cycles$ are needed in total to deliver the head flit at the NI of the target. The remaining flits are transferred in a pipeline manner, thus 1 $more\ cycle\ per\ flit$ is needed for delivering the entire packet to the NI. Packet routing is static, source-based, following the across-first strategy, and we implemented it in the gem5 routing tables.

TTE messages are received at the gateway that is implemented in one of the on-chip CPUs. The other CPUs invoke individual processes with soft deadlines, producing packets with a fixed injection rate. The capability of the on-chip network to serve all requests depends on its bandwidth capacity. Here we study its capability in serving the TTE workload, with and without enabling concurrently other CPU processes. An analysis for different packet injection rates, similar to the one we present in [16], would expose additional extreme cases. In present work, we focus on the effect of STNoC parameters only, and consider a fixed injection rate specified by a service-level agreement. Furthermore, we do not examine the overhead added by any other on-chip resource, and hence our measurements do not include the time overhead from CPU computation, gateway processing, and memory access. We should notice that in order to generate packets we used the network tester primitive provided by gem5 [17]. We connected one network tester per STNoC node acting as initiator, instead of using a CPU model from a library.

C. Hard Deadline Workload Flow and its Representation

TTE messages carry both write and read memory requests. Once a TTE packet reaches the NI of a memory it is immediately consumed, the NI responds within a single cycle, and sends a new packet back to the gateway. This packet is then converted into Ethernet format and relayed to the TTE network. We recall that we do not measure the memory access time; we focus only on the NoC transfer overhead.

The TTE traffic is represented with a DAG that carries information about the dependencies between the virtual links (VL) and the relative deadline requirements. Figure 2 depicts part of the example we used. It resembles a high critical application from the healthcare domain consisting of both periodic and sporadic tasks, all characterized by deadlines; in a previous work this application was deployed onto a Time Triggered platform [18]. Further details on the DAG in terms of which task is periodic and which task is sporadic are out of the scope of this paper. The DAG nodes represent cluster-level TTE communication actions each assigned with a VLID, and the edges represent dependencies between the TTE communication actions. An edge means that the input from a source node, i.e. incoming virtual link, must be processed in order to provide output for a target node, i.e. outgoing virtual link. Processing is carried out by the STNoC-based MPSoC. For example, VL1 and VL2 are incoming virtual links that should be processed on-chip in order to feed the outgoing VL3; once the on-chip processing completes and the result is sent back to the TTE, VL3 becomes the incoming virtual link. In turn, after the on-chip processing of VL3 completes, it provides input to the VL5, and so on. The labels on edges correspond to the relative deadline between the reception of an input message from the incoming virtual link, and the transmission of an output message to the outgoing virtual link. This deadline refers to the maximum time that should be spent at chip-level, i.e. the time-window within which on-chip actions should complete. These are: the gateway processing for handling the incoming/outgoing messages, the transfer on NoC from the gateway to memory and back, and the memory access. Here we concentrate on the NoC transfer only to examine whether its overhead results in missing deadlines.

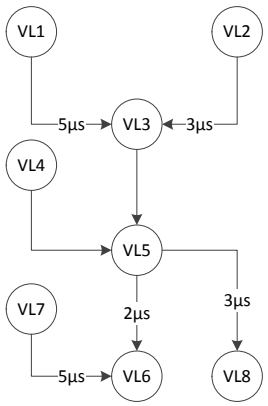

Fig. 2: DAG with dependencies between virtual links for on-chip processing, and on-chip deadlines.

DAG nodes without a label on their output edges correspond to read-only messages. The maximum permitted time specified by a deadline includes the time for completing read actions of such subsequent nodes. For instance, in Figure 2, VL3 output edge is not labeled; the smallest deadline of a previous node, i.e. VL2, specifies the maximum permitted time for completing the processing of both VL2 and VL3.

IV. Experimental Cases and System Parameters

The MPSoC we studied is shown in Figure 3; CPU $\{0\}$ implements the gateway between TTE and STNoC.

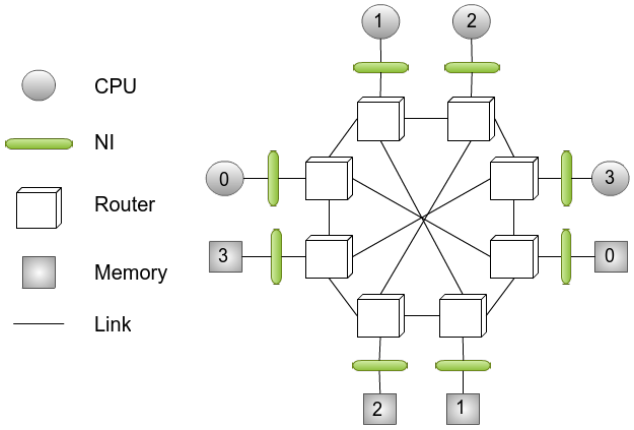

Fig. 3: 8-tile Spidergon STNoC. Gateway connection for incoming/outgoing TTE messages is implemented in CPU{0}.

A. Application Mapping and Traffic Direction

The TTE messages entering the gateway induce write and read requests to the memory tiles $\{0\}$, $\{2\}$ and $\{3\}$ of Figure 3. In particular, a write packet is sent to a memory, and once arrived at its NI, it responds within a single cycle by sending another packet back to the gateway. In our mapping scenario we made a convention with regard to the memories to which the different TTE message types are sent; sporadic messages are forwarded to different memories than periodic messages, therefore the respective transfer times vary depending on the number of intermediate routers in each path. The injection rate of requests by the gateway is low; in a total of 10 $million\ clock\ cycles$ of simulation time the gateway releases 250 $requests$. This is driven by the rate of TTE messages arriving at the gateway. The other on-chip CPUs invoke processes injecting requests to any on-chip memory in a random way, and they are of write-type only. They are less critical processes produced by real-time applications with soft deadlines, such as continuous media, and real-time performance monitoring, for which the required service rate is specified by a service-level agreement.

B. Parameter Selection and Scenarios

Apart from the NoC link-width, we varied its QoS parameters such as the number of VCs, and the relative priority of packets transferred on the same VC. We measured the NoC delay at flit- and packet- level, and for some cases the corresponding power consumption. We tested the following scenarios to cover a sufficient range of cases that exhibited critical situations:

#1 STNoC serves the TTE workload only, using one VC;
#2 on-chip CPU processes generate concurrently memory requests. TTE workload is transferred on the high-priority VC0, CPU workloads on the low-priority VC1;
#3 all workloads share the same VC, without setting priorities (FBA in the routers is off); and
#4 all workloads share the same VC, and we examine the effect from setting different packet priorities (FBA is on).

The gateway receives TTE messages of varying sizes, and converts them into network packets of fixed size prior to transmitting them to STNoC. TTE messages do not exceed the amount of 100 $Bytes$, hence we selected a payload of 128 $Bytes$ to fit the largest TTE message. The packet header size is 8 $Bytes$. Prior to injecting a packet to the STNoC, this is broken down into flits. The router's buffer size was set equal to the number of flits comprising a packet. On the other hand, we set large queues at the NIs in order to avoid potential overflows when new data arrive; although this is unrealistic, it prohibited packet dropping, and thus allowed us to find the time for serving a packet in the worst case and also assess saturation cases, by measuring the time a packet remains in the NI queue prior to releasing it into the STNoC. In a real system the NI queue is relatively small, and packet dropping is handled by either controlling the rate of data entering the queue depending on its status, or, with retransmission of dropped packets.

The packet injection rate from each CPU was set equal to 0.05 $packets\ per\ clock\ cycle$, i.e. 1 $packet\ every$ 20 $clock\ cycles$. This is driven by the rate of CPU messages arriving at the protocol prior to NI. In [16] we showed that for a similar setup with 8 nodes and link-width equal to 8 $Bytes$, saturation occurs above 0.035 $packets\ per\ clock\ cycle\ per\ CPU$. We expected here that the given rate, i.e. 0.05, will cause NoC saturation.

V. Performance Evaluation

We performed experiments in the framework published in [15] that allows measuring the end-to-end delivery times and the power consumption of routers and links. We measure two types of delay; the time a packet is waiting in the NI queue before it is split up into flits and released to the network (called NI queue delay), and then, after the head flit leaves the NI queue, the time elapsed for traversing the network until it reaches the destination (called network traversal delay). This way we measure the time for transferring the entire packet. We also measure the time for serving a TTE message after converting it into STNoC packet format; this is composed of the time for a TTE packet to reach the NI of the target memory, and the time for the response packet to be delivered back to the gateway. We refer to this as TTE message or TTE packet round-trip time. We ran each simulation for 10 $million\ clock\ cycles$, and we repeated the experiments to validate the results. For the traffic rates we reported earlier, gem5 simulation ran fast, carrying out $140,000\ cycles/sec$ on an Intel i5-3470 clocked at $3.2GHz$.

A. Varying the Link-Width

Table I has results for scenario #1 and shows that the network traversal delay and the TTE message round-trip time decrease when link-width increases. This is due to the bigger amount of data transferred per flit and the smaller amount of flits per packet, i.e. when link-width is 8 $Bytes$, the TTE message is broken down in 17 $flits$, while for 16 and 32 $Bytes$ it is broken down in 9 and 5 $flits$ respectively. We observe that when link-width is 32 $Bytes$, the TTE message round-trip time drops considerably, down to $24-28$ $clock\ cycles$ (cc). However, the wide link-width increases router's total power, mainly due to its clock; we

observe that when link-width doubles, clock power doubles as well. We also notice that TTE message round-trip time varies in each case, i.e. between $48-52cc$, $32-36cc$, and $24-28cc$. This is due to the application mapping we reported earlier; for different message types, periodic or sporadic, the corresponding packets follow different routing paths, which in turn results in different transfer times, e.g. reaching memory $\{0\}$ from gateway requires fewer hops than for memory $\{2\}$.

TABLE I: Effect from link-width (lw) variation on NI queue delay, network traversal delay, TTE packet round-trip time, and power consumption. Values are average and runs concern scenario #1, i.e. TTE workload only.

Network metric	lw=8 B	lw=16 B	lw=32 B
NI queue delay	1.555 cc	1.555 cc	1.555 cc
Network traversal delay (flit)	13.444 cc	9.444 cc	7.444 cc
TTE packet round-trip time	48-52 cc	32-36 cc	24-28 cc
Router total power	0.261 W	0.380 W	0.616 W
Router clock power	0.095 W	0.191 W	0.383 W

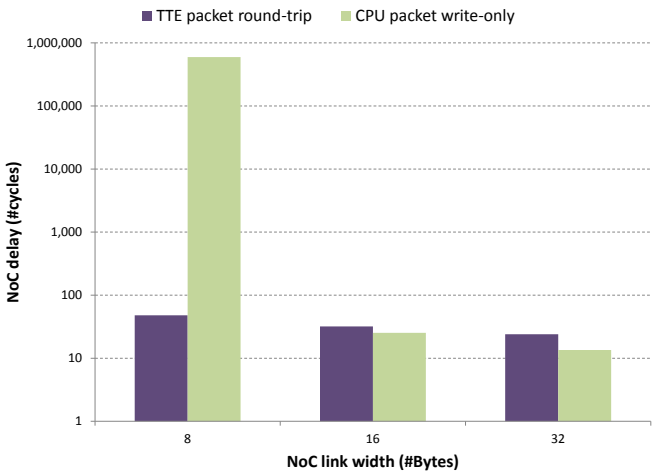

Fig. 4: Average NoC delay (logarithmic scale) for different link-widths, for scenario #2 in which TTE and CPU packets are transferred on separate VCs.

Figure 4 depicts the total NoC delay per packet, i.e. sum of NI queue delay and network traversal delay for all its flits, as an average from all CPUs, and the average TTE packet round-trip time, for scenario #2. We obtain the best results when link-width is 32 $Bytes$, but this comes at the expense of power. We also observe that when link-width is 8 $Bytes$, the CPU workload causes saturation of the NoC, leading to enormous delays. In fact, we noticed that NoC begins to saturate early; in the first $10,000$ $clock\ cycles$ of simulation the average NoC delay per CPU packet went up to 724 $clock\ cycles$. Saturation depends on the bandwidth capacity of NoC and emanates from its incapability to accept new packets arriving with the given rate, as it is busy transferring other packets. On the other hand, when link-width is 32 $Bytes$, the average NoC delay drops down to 25 $clock\ cycles$. Finally, for all link-widths the NoC satisfies the requirements of TTE workload as round-trip time reaches up to 52 $clock\ cycles$ ($52ns$), which is much smaller than any deadline in Figure 2.

B. One or Two Virtual Channels?

To explore the variation in packet delivery time, we analyzed each packet individually. We did this initially for the high critical workload. We configured the NoC with the narrowest link-width, and performed tests for all scenarios. Figure 5 shows the TTE packet round-trip time over the simulation time window in which we observed the largest variations. Each of the scenarios #1 and #2 is characterized by minor variations in the TTE packet round-trip time, i.e. $48-52$ $clock\ cycles$. Therefore, the dedicated high-priority VC for the hard deadline workload keeps the time for transferring the corresponding packets within predictable levels. The small variation is due to that packets corresponding to periodic messages follow a different path from the sporadic ones; this was noticed also in the previous subsection. On the other hand, in scenarios #3 (1 VC, FBA is disabled) and #4 (1 VC, FBA is enabled) we observe considerable variations, i.e. TTE round-trip time varies between $48-143$ $clock\ cycles$ and $48-119$ $clock\ cycles$ respectively. It appears that a shared VC affects the TTE round-trip time significantly. Moreover, in scenario #3 in which FBA is disabled, we observed the longest delay in the round-trip. Below we analyze further these time variations, along with a worst case timing analysis, for both high critical and low critical workloads.

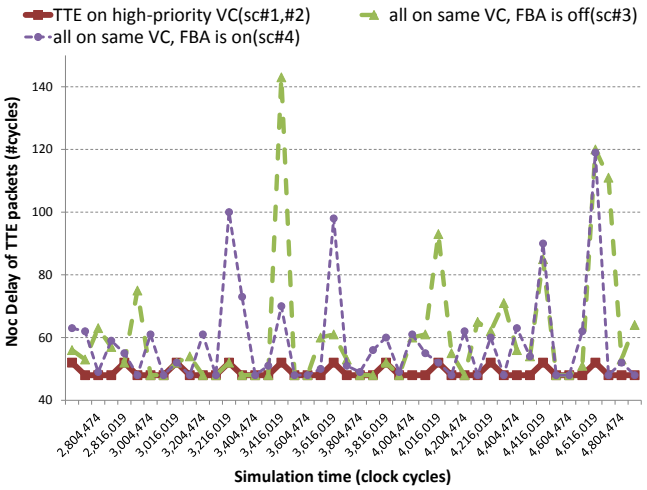

Fig. 5: Variation of NoC round-trip time for the TTE packets over simulation running time, for scenarios #1-#4. Link-width is 8 Bytes.

C. Worst Case Transfer Time of NoC

There are two main methods to determine worst case timing: static analysis and measurement-based. Our setup allows for performing the second one; we analyzed the worst case times for transferring packets throughout the on-chip network, which we term $WCTT_{noc}$. As it is not based on mean times, we

TABLE II: $WCTT_{noc}$ analysis for the TTE packet round-trip time, and for the time per CPU packet for reaching a memory. Values are in NoC clock cycles.

	link-width=8 Bytes				link-width=16 Bytes			
scnr.	TT_{nocTTE}	$WCTT_{nocTTE}$	TT_{nocCPU}	$WCTT_{nocCPU}$	TT_{nocTTE}	$WCTT_{nocTTE}$	TT_{nocCPU}	$WCTT_{nocCPU}$
#1	48-52cc	52cc	n/a	n/a	32-36cc	36cc	n/a	n/a
#2	48-52cc	52cc	saturation	-	32-36cc	36cc	15-269cc	269cc
#3	48-143cc	143cc	saturation	-	32-64cc	64cc	15-270cc	270cc
#4	48-119cc	119cc	saturation	-	32-62cc	62cc	15-270cc	270cc

examined each packet of each workload. Table II has the results for all scenarios and two link-widths. We observe again that a dedicated high-priority VC maintains time-predictability in the transfer of hard deadline packets. We reported this in the previous subsection for link-width equal to 8 $Bytes$, and here we demonstrate that it holds also when link-width is 16 $Bytes$. The same tendency with the previous subsection holds when 1 VC is shared amongst the workloads; but for link-width equal to 16 $Bytes$, the TTE round-trip time experiences smaller variations, and NoC is not saturated. Soft deadline packets can be served, given that the corresponding flows can tolerate NoC transfer times up to 270 $clock\ cycles$ for a packet.

We then measured the piece-wise delays for releasing and transmitting a packet. We did this for the soft deadline packets, which have longer transfer times in Table II, and we found that in the worst cases the largest portion of the time is consumed in the NI queue. When link-width is 16 $Bytes$, 255 out of the 270 $clock\ cycles$ of the $WCTT_{nocCPU}$ are consumed in the NI queue of the initiator. This is due to that while CPU messages arrive at the NI protocol, the NoC cannot accept them immediately as it is busy serving other packets. This problem propagates back to the NI of initiator, and prevents it from breaking down the new packet in flits immediately. The packet remains in the NI queue of the initiator, and new packets arriving from the CPU are stacked in it waiting for the NoC to become uncongested.

D. Discussion

Our analysis revealed that the NoC with link-width equal to 16 $Bytes$ serves the requirements of all workloads. The narrowest link leads to NoC saturation, and a wider link increases router's clock power consumption. Future study includes the energy consumption per workload, or, per session opened by an initiator for completing one job. We demonstrated that setting up routers with 1 VC allows for meeting all requirements without missing any deadline, neither degrading highly the NoC transfer rate, i.e. max transfer time is 270 cc, which accounts for resource savings. We found that FBA improved slightly the transfer time of hard deadline packets, but a dedicated high-priority VC performs better by maintaining time-predictability. The above can be taken into account when considering the trade-offs prior to configuring the NoC.

ACKNOWLEDGMENT

This research was funded by the EU FP7/2007-2013, under grant agreement no. 610640, DREAMS project.

REFERENCES

[1] http://www.mixedcriticalityforum.org/projects/.
[2] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. de Dinechin, "The Shift to Multicores in Real-Time and Safety-Critical Systems," in *International Conference on CODES+ISSS*, October 2015, pp. 220–229.
[3] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele, "Scheduling of Mixed-Criticality Applications on Resource-Sharing Multicore Systems," in *ACM/IEEE International Conference on Embedded Software (EMSOFT)*, September 2013, pp. 1–15.
[4] S. Tobuschat, P. Axer, R. Ernst, and J. Diemer, "IDAMC: A NoC for Mixed Criticality Systems," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2013, pp. 149–156.
[5] A. Burns, J. Harbin, and L. S. Indrusiak, "A Wormhole NoC Protocol for Mixed Criticality Systems," in *IEEE Real-Time Systems Symposium (RTSS)*, December 2014, pp. 184–195.
[6] G. Giannopoulou, N. Stoimenov, P. Huang, L. Thiele, and B. D. de Dinechin, "Mixed-criticality Scheduling on Cluster-based Manycores with Shared Communication and Storage Resources," *Real-Time Systems, The International Journal of Time-Critical Computing Systems*, vol. 59, no. 1, pp. 1–51, 2015.
[7] K. Goossens and A. Hansson, "The Aethereal Network on Chip after Ten Years: Goals, Evolution, Lessons, and Future," in *ACM/IEEE Design Automation Conference (DAC)*, June 2010, pp. 306–311.
[8] E. de Faria Corrêa, E. W. Basso, G. R. Wilke, F. R. Wagner, and L. Carro, "The Implications or Real-Time Behavior in Networks-on-Chip Architectures," in *IFIP Conference on Distributed and Parallel Embedded Systems (DIPES)*, August 2004, pp. 307–316.
[9] R. Obermaisser, *Time-Triggered Communication*. Taylor & Francis, 2011.
[10] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. Taylor & Francis, 2008.
[11] B. B. Brandenburg and J. H. Anderson, "Integrating Hard/Soft Real-Time Tasks and Best-Effort Jobs on Multiprocessors," in *Euromicro Conference on Real-Time Systems (ECRTS)*, November 2007, pp. 61–70.
[12] S. Chakraborty, M. Lukasiewycz, C. Buckl, S. A. Fahmy, N. Chang, S. Park, Y. Kim, P. Leteinturier, and H. Adlkofer, "Embedded Systems and Software Challenges in Electric Vehicles," in *Design Automation and Test in Europe (DATE)*, March 2012, pp. 424–429.
[13] *Cannes Ultra - ARM-based, UHD Multimedia Connected Client-Box Platform*, STMicroelectronics, December 2013.
[14] M. Abuteir and R. Obermaisser, "Simulation Environment for Time-Triggered Ethernet," in *IEEE International Conference on Industrial Informatics (INDIN)*, July 2013, pp. 642–648.
[15] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, O. Tomoutzoglou, G. Tsamis, and M. Coppola, "Security in MPSoCs: A NoC Firewall and an Evaluation Framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pp. 1344–1357, August 2015.
[16] K. Papadimitriou, P. Petrakis, M. D. Grammatikakis, and M. Coppola, "Security Enhancements for Building Saturation-free, Low-power NoC-based MPSoCs," in *IEEE International Conference on Communications and Network Security (CNS)*, September 2015, pp. 594–600.
[17] http://www.m5sim.org/Publications.
[18] Z. Owda, M. Abuteir, R. Obermaisser, and H. Dakheel, "Predictable and Reliable Time Triggered Platform for Ambient Assisted Living," in *IEEE International Symposium on Medical Information and Communication Technology (ISMICT)*, April 2014, pp. 1–5.