InterNoC: Unified Deterministic Communication For Distributed NoC-based Many-Core

Eleftherios Kyriakakis
Technical University of Denmark
Kgs. Lyngby, Denmark
elky@dtu.dk

Jens Sparsø
Technical University of Denmark
Kgs. Lyngby, Denmark
jspa@dtu.dk

Martin Schoeberl
Technical University of Denmark
Kgs. Lyngby, Denmark
masca@dtu.dk

ABSTRACT

Network-on-Chip is a popular paradigm for scalable many-core communication. There is a trend in modern system-on-chip to integrate more functionality. This combined with recent research for network-on-chip in the aerospace industry, gives room for design space exploration in new architectural paradigms for distributed and real-time many-core communication. In this paper, we present InterNoC, a deterministic communication scheme for distributed network-on-chip many-core that allows for unified IP-based time-triggered communication. It is hypothesized that such an architecture will efficiently minimize communication complexity in distributed many-core systems as well as provide hard-bounded end-to-end latency guarantees. We extend the real-time multi-core platform T-CREST by introducing a time-triggered NoC-based switching mechanism combined with a NoC packet to Ethernet frame traffic controller. The proposed architecture will be evaluated in an experimental InterNoC network that implements a 36-core real-time system distributed over four FPGA devices.

1. INTRODUCTION

The shift to chip multi-processors (CMPs) in real-time systems is inevitable as emerging technologies in the field of industrial automation and industrial internet of things continue to increase the demand for performance on the edge of the network while maintaining strict requirements in low power consumption [12]. To meet these requirements, the number of cores in modern many-core system-on-chip continues to scale and thus the requirements on resources and bandwidth of traditional buses become demanding. Network on-chip (NoC) have proven to be a viable solution for scalable many-core on-chip interconnection as they allow for efficient all-to-all communication [4].

Most research in the field of NoC focuses on improving the functional characteristics of the architecture as well as analyzing the application mapping strategies to such many-core systems. Over the years, few works have been presented that discuss the use of NoC for inter-chip communication, particularly in the field of real-time systems. In this paper, we envision a prototype network architecture that allows for a unified deterministic communication scheme for distributed real-time many-core systems. It is hypothesized, that such an architecture will allow real-time many-core applications to deploy tasks on cores in a locality agnostic way. Essentially, a task will execute the same code regardless if it needs to communicate with other tasks found on- or off-chip. We identify two possible ways to extend a NoC over physically distributed CMPs: 1) the local NoC communication (i.e. flits) gets encapsulated in an inter-chip communication protocol or 2) the Ethernet network protocols are pulled into the local NoC communication, abstracting away any on- and off-chip communication details from the application. In this paper, we focus on investigating the implementation of the second option as it abstracts away from the application any details about the internal architecture and routing of the NoC and allows to integrate heterogeneous systems.

The contribution of this paper is a first exploration of the idea of distributed CMP real-time systems that aims to provide seamless and scalable inter-chip time-triggered communication using the Internet Protocol (IP). It is based on the time-division multiplexed NoC architecture Argo, the time-predictable processor Patmos and an Ethernet controller extension. The communication is realized by implementing a new software abstraction layer between NoC packets and Ethernet IP frames. Bounded latency guarantees and contention-free communication is achieved by enforcing a network-wide time-triggered schedule.

The rest of the paper is organized in 7 sections: Section 2 provides the user with a background of the related work in the field of NoC-based inter-chip communication. Section 3 provides the reader with a background on the fundamental components the proposed communication is built on. Section 4 presents the proposed architecture and describes its integration with the T-CREST platform. Section 5 provides a preliminary evaluation of the feasibility and characteristics of the proposed architecture. Section 6 describes the future scope of this project and the immediate research focus points. Finally, Section 7 concludes the paper.

2. RELATED WORK

Inter-chip communication has been previously explored in the context of distributing the processing resources of a NoC and the following works present different techniques for implementing an off-chip bridge for NoC system-on-chip [9, 11, 20]. In this work, we implement Ethernet as the physical link for the inter-chip NoC communication as it allows for greater interoperability that can take advantage of different industrial Ethernet technologies. It also facilitates for software development as applications can communicate using a single protocol regardless if they are on the same chip or distributed over multiple devices.

More recently, in [1] the authors propose a mixed avionics full-duplex Ethernet (AFDX) and NoC communication between many-core applications. The proposed communication model is motivated by real-time use-cases in the field of avionics and health monitoring the authors investigated different application mapping strategies concerning the effects of contention on the worst-case traversal time of communication flows. Moreover, in [2] the authors propose a function mapping strategy in mixed AFDX/NoC communication that aims to minimize flow contentions and thus provide bounded jitter.

Another related distributed real-time system is presented in [19]. The authors propose and implement the Mock Turtle architecture, which is based on 32-bit RISC-V processor cores. On-chip communication is provided by a shared memory mechanism while off-chip communication is implemented using the white-rabbit network [16]. As shared memory communication does not scale well and with a cache coherence protocol is hardly time predictable, we use a NoC for on-chip communication.

Finally, in contrast to previous works the proposed architecture does not suffer from contention as it is based on a network-wide time-triggered schedule and a time-division multiplexing (TDM) NoC design. The software driver handling the time-triggered communication is implemented on the time-predictable Patmos processor allowing for worst-case execution time (WCET) analysis. The building blocks of this architecture allow for a bounded end-to-end communication latency that can be statically calculated as shown in Section 5.

3. BACKGROUND

This section aims to briefly introduce the existing architecture components, namely the processor and the NoC architecture, which the proposed InterNoC design uses for the implementation. Both of them are part of the open-source research platform T-CREST [13], which is an FPGA-based many-core platform that has been developed for on-going research in real-time applications.

3.1 Argo NoC

Argo [17] is a packet-switched and source-routed NoC that uses TDM to guarantee bandwidth and latency. It allows for deterministic unicast message-passing communication over virtual channels, which is implemented using dedicated direct-memory access (DMA) controllers for each source-end of every virtual channel. The DMAs are integrated with a TDM mechanism in the network-interface (NI), eliminating the need for buffering and flow control. According to the TDM schedule, each virtual channel gets a time-slot during which it can start sending a message. These time-slots are assigned according to a statically scheduled period.

The TDM mechanism is implemented in each Argo NI as TDM counter. This TDM counter indexes a schedule table that in turn points to an entry in the DMA table containing the respective counters and pointers for a DMA access. The NI exposes two interfaces to the processor core, a configuration interface and a data interface. The data interface allows the processor to exchange data with the NI's dual-port scratch-pad memory. The configuration provides access to both the schedule table as well as the DMA controller.

It is worth noting, that Argo supports on-the-fly schedule reconfiguration. This can be done with zero delays experienced by any on-going messages in a virtual channel communication as described in [17], but using this mechanism is outside the scope of this work.

3.2 Patmos

Patmos [15] is a time-predictable WCET-optimized, dual-issue RISC processor that has been designed with a focus on WCET analysis. It uses special WCET-optimized instruction and data caches along with private scratchpad memories for instructions and data. It is supported by an LLVM-based [10] compiler, also optimized for WCET and by the WCET analysis tool *platin* [6].

The tool *platin* performs static analysis to compute the WCET of a certain code segment by using the information generated and preserved during compilation to determine a control flow graph. Together with low-level timing information of the processor architecture, it can calculate a safe WCET bound of the analyzed code segment.

4. ARCHITECTURE

The architecture of InterNoC is building on top of the statically scheduled TDM Argo NoC (see Section 3.1), but the communication concept can be applied to any NoC architecture as long as it can guarantee bandwidth allocation. Moreover, the NoC cores are implemented using the Patmos processor, allowing for time-predictable and statically WCET analyzable execution of software tasks. This combination allows the proposed architecture to have a fully deterministic communication scheme.

The proposed architecture defines two types of NoC cores, (1) a *computation core* and (2) a *gateway core* that is interfaced with an Ethernet controller and is responsible for handling incoming and outgoing traffic. This is illustrated in Figure 1, where *core 0* is assigned the role of the *gateway core*.

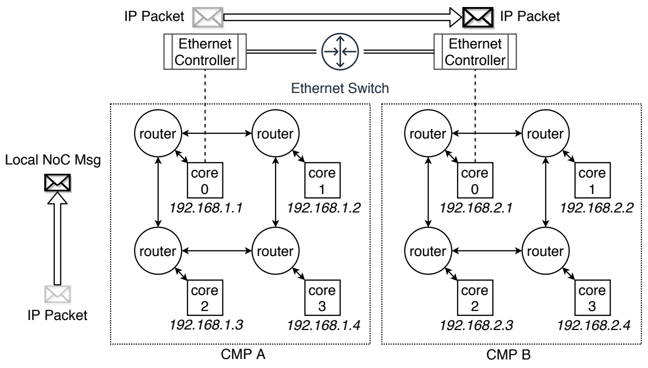

Figure 1: Local and distributed NoC communication concept over IP

When an application wants to transmit a message it constructs an IP packet and passes the data to the driver. Consecutively, the driver checks if the subnet of the requested IP address is on- or off-chip. If the destination is on-chip it uses the respective NoC virtual channel and writes the IP packet to the corresponding DMA to reach the destination *computation core*. If the destination subnet IP address is off-chip, the driver uses the respective NoC virtual channel pointing to the *gateway core* and writes the IP packet to the respective buffer.

Each *gateway core* executes two periodic tasks. The first task is responsible for collecting incoming IP packets from the on-chip *computing cores*, encapsulating them into Ethernet frames and forwarding them at specific time-slots using the Ethernet controller. The second task is responsible for polling the Ethernet controller for incoming IP packets and depending on the destination IP address creating the NoC channel and transmitting the incoming IP packet to the respective NoC node.

Due to the difference in time granularity between the local NoC schedule and the Ethernet transmission, each *gateway core* performs a store-and-forward scheme and maintains separate buffers per NoC channel for the incoming NoC packets, which it needs for to reconstructs the respective IP packets. For example, if a NoC packet can transfer a maximum of two words of data (64 bits) per NoC TDM slot, then the on-chip transmission of a maximum size Ethernet frame (1518 bytes) requires 190 NoC TDM slots to complete. This is further discussed in Section 5.

All the components of the presented architecture are developed as open-source and are hosted under the T-CREST project GitHub repository `https://github.com/t-crest/`.

5. EVALUATION

This section provides an initial evaluation of the scalability and feasibility of the presented architecture. The proposed communication is composed of two types of traffic:

1. Intra-CMP, referring to the on-chip NoC communication
2. Inter-CMP, referring to the off-chip communication that processing cores can access via Ethernet.

A full evaluation of the intra-CMP communication for Argo has been already presented in [14, 18]. In this evaluation, we try correlate the results with the proposed IP-NoC abstraction layer's added overhead and the additional inter-chip communication scheme.

Figure 2 presents the constraints for the inter-chip time-triggered communication that define the schedule period (length). These are the total number CMPs, the number of cores per CMP and the number of frames available for transmission/reception by each core.

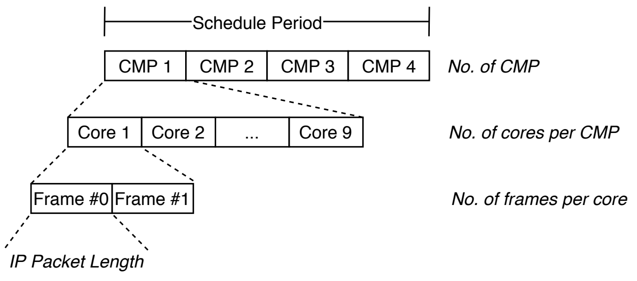

Figure 2: Constraints defining the inter-CMP schedule period

The evaluation is motivated by three industrial Ethernet traffic classes, defined in [5], that specify the schedule period requirements. The available bandwidth for inter-CMP communication is assumed to be 100 Mbps, as this is commonly used in real-time communication protocols such as the TTEthernet [8]. Table 1 presents the number of frames per schedule period that can be exchanged using the inter-chip links, by taking into account two types of frames, 1) full Ethernet frames of 1518 bytes and 2) minimum Ethernet frames of 64 bytes.

Table 1: Available frames versus inter-CMP schedule period

Class	Schedule Period (ms)	No. of Frames	
		Full frames	Min. frames
1	100	833	19531
2	10	83	1953
3	0.25 – 1	2 – 8	48 – 195

In most industrial use-cases the period of real-time traffic constraints the frame size. Assuming a hard real-time schedule period of 1 ms that aims to cover an all-to-all communication schedule, with min. frame size of 64 bytes (6 bytes of IP payload). This means that M CMPs containing N cores share a bandwidth of 195 frames or 1.17 MBps that each carries 6 bytes of IP payload. For example, a system composed of $M = 4$ then each CMP has 44 time-slots available for inter-CMP communication. This can be seen as a trade-off between the number of cores per CMP and the assigned bandwidth to each CMP core. The application designer is responsible for minimizing the off-chip communication flows. The subject of task mapping strategy is outside the scope of this work.

The worst-case traversal time (WCTT) of the proposed architecture is end-to-end bounded since the Argo NoC operates on a TDM schedule and the planned inter-CMP communication is time-triggered. Thus, the WCTT of a packet in the presented architecture can be statically expressed by $WCTT_{e2e}$ in Equation 1. The $WCTT_{e2e}$ is composed of three parts, the worst-case traversal time of the NoC as $WCTT_{noc}$, the worst-case traversal time of the Ethernet link as $WCTT_{eth}$ and the worst-case execution time $WCET_{driver}$ of the abstraction layer that performs the store-forward and the translation between NoC and IP packets.

The WCTT of the NoC is presented in Equation 2. $WCWT_{noc}$ is the worst-case waiting time that a NoC packet can experience and is equal to the period of the NoC TDM schedule, $Size_{IP}$ is the size of the transmitted/received IP packet, B_{noc} is the bandwidth of the NoC channel, L_{noc} is the end-to-end latency of the NoC and T_{noc} is the NoC schedule period.

Finally, the worst-case traversal time over Ethernet is presented in Equation 3. The $WCWT_{eth}$ is the worst-case waiting time an Ethernet frame can experience before starting the transmission, $Size_{eth}$ is the size of the Ethernet frame, B_{eth} is the available bandwidth of the Ethernet connection and L_{switch} is the Ethernet switch latency.

$$WCTT_{e2e} = WCTT_{eth} + 2 * (WCTT_{noc} + WCET_{driver}) \qquad (1)$$

$$WCTT_{noc} = WCWT_{noc} + \frac{Size_{IP}}{B_{noc}} * T_{noc} + L_{noc} \qquad (2)$$

$$WCTT_{eth} = WCWT_{eth} + 2 * \frac{Size_{eth}}{B_{eth}} + L_{switch} \qquad (3)$$

Following is a short example of an experimental setup composed of four CMPs with nine cores each communicating over a single Ethernet switch. Assuming a bi-torus topology of the NoC we can safely assume that the worst-case latency L_{noc} is equal to one TDM NoC period. For a 3x3 Argo NoC the schedule period is 10 clock cycles [14]. Furthermore, we assume that the architecture is implemented on FPGA technology with 80 MHz clock. To calculate the $WCTT_{e2e}$ between two CMPs, first, we calculate the WCTT for Min. size IP frame, for each NoC, to be $WCTT_{noc} = 1.25\ \mu s$. Secondly, we calculate the WCTT for Ethernet, assuming $B_{eth} = 100 Mbps$ and $L_{switch} = 3.9\ \mu s$ [7]. According to Table 1 the system can support a schedule period of 0.25 ms, where each core is allowed to transmit one frame per period, thus $WCTT_{eth} = 260.24\ \mu s$. Finally, we calculate $WCTT_{e2e} = 266.64\ \mu s + WCET_{driver}$. In future works, a WCET analysis of the abstraction layer driver will allow for the exact calculation of the end-to-end WCTT.

6. DISCUSSION AND FUTURE WORK

The proposed architecture is hypothesized to allow building Internet applications on-top of a deterministic IP-NoC abstraction layer that efficiently connects distributed real-time CMPs.

We plan to evaluate the communication on an experimental setup that implements a 36-core architecture distributed over four physically separate devices, similar to what was visualized in Figure 2. The system is implemented in four Altera Cyclone IV FPGA devices [3] that each implements a 3x3 NoC and communicate over Ethernet using a standard off-the-shelf switch. Such a system would be otherwise impossible to fit on a single FPGA device, something that emphasizes the importance of this communication scheme for FPGA-based many-core real-time systems. We plan to implement the experimental setup on a TTEthernet network and integrate the off-chip time-triggered communication with the TTEthernet schedule. We will investigate distributed systems schemes such as client-server, publish-subscribe and remote process call built on top-of-the presented communication scheme.

Finally, since the abstraction layer implements IP, it allows for the physical communication link to be extended to other technologies such as wireless. However, this requires different mechanisms to guarantee determinism which are outside the scope of this work.

7. CONCLUSION

In this paper, we proposed a new paradigm of unified communication for NoC-based CMPs. The proposed architecture allows to minimize communication complexity and improve real-time systems scalability by extending the TDM Argo NoC using an IP Ethernet frame to NoC packets abstraction layer and a time-triggered off-chip communication scheme. The presented communication scheme extends the Argo message-passing library and the developed software API is statically WCET analyzable as it is implemented on the time-predictable processor Patmos.

A first evaluation of the scalability and design constraints of the proposed communication was presented. It revealed that an application mapping strategy must aim to minimize off-chip communication links, if hard real-time cycle time needs to be guaranteed. Furthermore, it was shown that the end-to-end communication delay can be guaranteed in a statically analyzable way.

8. ACKNOWLEDGEMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation.

9. REFERENCES

[1] L. Abdallah, J. Ermont, J.-L. Scharbarg, and C. Fraboul. Towards a mixed noc/afdx architecture for avionics applications. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, pages 1–10. IEEE, 2017.

[2] L. Abdallah, J. Ermont, J.-L. Scharbarg, and C. Fraboul. Reducing afdx jitter in a mixed noc/afdx architecture. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–4. IEEE, 2018.

[3] ALTERA. *Cyclone IV FPGA Device Family Overview*, March 2016.

[4] L. Benini and G. De Micheli. Networks on chips: A new soc paradigm. *computer*, 35(1):70–78, 2002.

[5] P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann, and J. Schacht. Survey on real-time communication via ethernet in industrial automation environments. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8. IEEE, 2014.

[6] S. Hepp, B. Huber, J. Knoop, D. Prokesch, and P. P. Puschner. The platin tool kit - the T-CREST approach for compiler and WCET integration. In *Proceedings 18th Kolloquium Programmiersprachen und Grundlagen der Programmierung, KPS 2015, Pörtschach, Austria, October 5-7, 2015*, 2015.

[7] Hewlett-Packard Development Company, L.P. *ProCurve Switch 1700 Series*, March 2007.

[8] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (tte) design. In *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pages 22–33. IEEE, 2005.

[9] E. Kyriakakis, K. Ngo, and J. Öberg. Implementation of a fault-tolerant, globally-asynchronous-locally-synchronous, inter-chip noc communication bridge on fpgas. In *2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pages 1–6. IEEE, 2017.

[10] C. Lattner and V. S. Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *International Symposium on Code Generation and Optimization (CGO'04)*, pages 75–88. IEEE Computer Society, 2004.

[11] W. H. Minhass, J. Öberg, and I. Sander. Implementation of a scalable, globally plesiochronous locally synchronous, off-chip noc communication protocol. In *2009 NORCHIP*, pages 1–5. IEEE, 2009.

[12] 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 *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*, pages 220–229. IEEE Press, 2015.

[13] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, C. Silva, J. Sparsø, and A. Tocchi. T-CREST: Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 61(9):449–471, 2015.

[14] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 152–160. IEEE, 2012.

[15] M. Schoeberl, W. Puffitsch, S. Hepp, B. Huber, and D. Prokesch. Patmos: A time-predictable microprocessor. *Real-Time Systems*, 54(2):389–423, Apr 2018.

[16] J. Serrano, M. Lipinski, T. Wlostowski, E. Gousiou, E. van der Bij, M. Cattin, and G. Daniluk. The white rabbit project. 2013.

[17] R. B. Sørensen, L. Pezzarossa, M. Schoeberl, and J. Sparsø. A resource-efficient network interface supporting low latency reconfiguration of virtual circuits in time-division multiplexing networks-on-chip. *Journal of Systems Architecture*, 74(Supplement C):1–13, 2017.

[18] J. Sparsø, E. Kasapaki, and M. Schoeberl. An area-efficient network interface for a tdm-based network-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1044–1047. EDA Consortium, 2013.

[19] T. Wlostowski, F. Vaga, and J. Serrano. Developing distributed hard-real time software systems using fpgas and soft cores. 2015.

[20] Y. Yin and S. Chen. Design and implementation of a inter-chip bridge in a multi-core soc. In *2009 4th International Conference on Design & Technology of Integrated Systems in Nanoscal Era*, pages 102–106. IEEE, 2009.