CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Conference Paper

Partitioning the Network-on-Chip to Enable Virtualization on Many-Core Processors

Matthias Becker
Dakshina Dasari
Vincent Nélis*
Moris Behnam
Thomas Nolte

*CISTER Research Center

CISTER-TR-150608

2015/07/07

Partitioning the Network-on-Chip to Enable Virtualization on Many-Core Processors

Leandro Becker, Dakshina Dasari, Vincent Nélis*, Moris Behnam, Thomas Nolte

*CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: dandi@isep.ipp.pt, nelis@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

In this paper, we highlight some key problems in NoC based architectures that must be addressed before the deployment of real-time applications onto these platforms becomes possible. A paradigm shift from function centric to data and communication centric approaches is required. Combining hardware and software based ﬂow-regulation seems to be the only way to ensure that NoCs go beyond the best-effort service and address the requirements of diverse applications.

Partitioning the Network-on-Chip to Enable Virtualization on Many-Core Processors

Matthias Becker*, Dakshina Dasari†, Vincent Nélis‡, Moris Behnam*, Thomas Nolte*
*MRTC / Mälardalen University, Sweden
{matthias.becker, moris.behnam, thomas.nolte}@mdh.se
† Research and Technology Centre, Robert Bosch, India
dakshina.dasari@in.bosch.com
‡CISTER/INESC-TEC, ISEP, Portugal
nelis@isep.ipp.pt

I. Introduction

Technological advances have increased the transistor density, thereby ushering in multi- and more recently many-core systems, distinguished by the presence of hundreds of cores on a single chip. For such a platform, the Network-on-Chip (NoC) has emerged as a scalable and efficient interconnect fabric to realize the communication across an ever increasing number of processor cores, memories, and specialized IP blocks both on- and off-chip. This shift in the interconnect mechanism has overcome the performance barriers of traditional shared buses that do not scale well and cross-bars that consume a lot of chip resources. Given the computational capabilities that this platform offers, several applications, with possibly different criticality/safety requirements, can be consolidated on one platform. System designers have been considering various mechanisms to harness these new platforms; one among them being virtualization. Virtualization techniques [1] are commonly used to partition one platform into several, independent, Virtual Machines (VM), with each partition given its own set of physical resources. While virtualization techniques are mature in single core systems, the clear differences in the processor architectures of many-cores compared to their precedents requires a different approach: (i) The large number of simple cores diminishes the need of partitioning on core level, i.e. a subset of cores will be allocated to one partition/application. (ii) The bandwidth on the different links of the NoC, on the other hand, must be partitioned.

a) Service Guarantees on the NoC: The NoC implemented in today's many-core processors generally provides services on a best-effort basis. With the need for guaranteed services, traffic regulation is proposed for flows with real-time constraints. In order to decrease the hardware complexity, traffic shaping is not implemented within the NoC routers. It is rather implemented in the source nodes, and thus limits the injection of messages into the NoC. As an example, Kalray's MPPA 256 processor provides a hardware based traffic shaper which allows to bound communication times using the (σ, ρ) network calculus [2]. Munk et al. present a software based traffic shaper [3] to allow for guaranteed communication services on many-core processors without hardware support.

In general, such a traffic shaper is configured with the parameters, T_w (the time window) and N_{max} (max. packets). According to [2], the translation to the (σ, ρ) parameters is done by $\rho = N_{max}/T_w$ and $\sigma = N_{max}(1 - N_{max}/T_w)$, as shown in Fig. 1. Once a message is scheduled for transmission, the traffic shaper checks if the number of packets transmitted during the last T_w cycles plus the number of packets of the scheduled message is less than N_{max}. If this is the case the message is sent. Otherwise the traffic shaper holds the message until the condition is true.

II. Challenges

In this section, we discuss the challenges in deploying applications onto the flow regulated NoC described above. From the application point of view, the challenge for the hypervisor is to configure the network flow limits across the NoC in such a way that all the application requirements are met, especially so when there is diversity in the requirements (ranging from soft or hard timing constraints) and also with varying traffic injection patterns. Though the ability to configure the flow regulation parameters is a useful feature, setting inappropriate values can have a negative impact on the performance. On the extremities, it could either lead to over-provisioning of the bandwidth, resulting in an under-utilized platform or it could lead to under-provisioning, resulting in severe degradation of the application performance (and possibly a violation of the timing constraints). Thus a network-wide default setting of the flow-limits may not satisfy the requirements of all the hosted applications. Additionally, it needs to be considered that flow regulation is done only at the source level. Flows, however, might have different paths, periods, and message sizes.

Designing the network virtualization layer to implement the above can be complex: Firstly it has to provide the basic functionality which is to abstract the physical network (links, routers, buffers, routing algorithms, etc.) and move it into the virtualization software layer, making it totally transparent to the user. Secondly, it may be necessary to provide the required resource reservations on the network for applications with higher priority, either by reserving fixed links and associated buffers on the network (like virtual circuits), by setting flow regulations as described above, or configuring rules in the routers to enforce this. All this can be challenging if limited configuration options are provided or it is required to implement it dynamically in the case of changing traffic loads. Additionally, in order to provide spatial isolation to safeguard some critical applications hosted

The work presented in this paper was partially supported by the Swedish Knowledge Foundation via the research project PREMISE and by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER Research Centre); by FCT/MEC and the EU ARTEMIS JU within project ARTEMIS/0001/2013 - JU grant nr. 621429 (EMC2); by the North Portugal Regional Operational Programme (ON.2 O Novo Norte) under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by National Funds through FCT/MEC (Portuguese Foundation for Science and Technology), within project NORTE-07-0124-FEDER-000063 (BEST-CASE, New Frontiers).

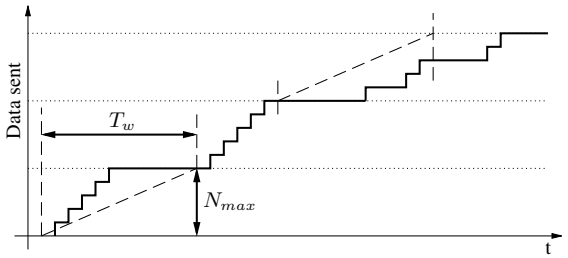
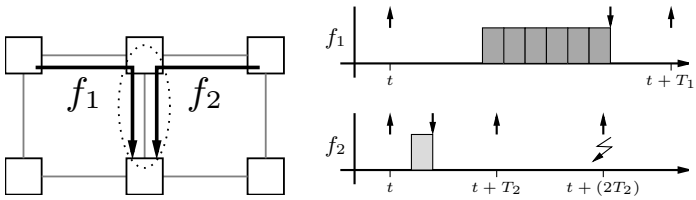


Fig. 1. Limiting the injection rate on the source nodes.



(a) Paths of two flows on a NoC as result of task mapping.

(b) Possible traffic pattern on the highlighted channel.

Fig. 2. Example of unschedulable messages as result of the task mapping.

on particular (safe) guest zones, the hypervisor may also have to restrict traffic from insecure sources from reaching the safe zones. Given the context above, in the next section we enlist specific open problems that must be addressed.

III. Discussion and Open Problems

Having traffic regulation on the source nodes enables guaranteed services for the messages on the NoC. However, as discussed above, the selection of the flow regulation parameters is not trivial. One solution could be to configure individual routers with different flow limits, but this can be pretty challenging considering the scale and it also clearly needs a fixed application mapping as a pre-requisite, in order to arrive at an optimal configuration. An uninformed mapping on the other hand may make it infeasible for the hypervisor to arrive at a network-wide optimal solution. This is for example the case in Fig. 2. The task placement leads to the message paths of the periodic messages, f_1 and f_2, with implicit deadlines (Fig 2(a)). Fig. 2(b) depicts an exemplary worst-case scenario on the channel which is shared by f_1 and f_2. Since the period T_2 is less than the time it takes to transmit a whole message of f_1 across one channel, the configuration is not schedulable. Because the best effort NoCs do not allow preemptions, traffic regulation can not be used to arrive at a schedulable system.

Software based traffic shaping at flow level, in tandem with the traffic shaping of the outgoing link could be applied in order to guarantee safe operation of each flow on the NoC. This way, the traffic shaping at flow level is used to limit the bandwidth of the flow, i.e. protect other parts of the system against malfunctioning. The second level can then be used to shape the outgoing traffic. Having such a configuration has many similarities to hierarchical scheduling. The period T_w of a traffic shaper could for example be selected similar to the techniques presented by Shin in [4].

As described above there are certain open problems which need to be addressed. Applications in the context of this paper can be modeled as a set of periodic tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task can have precedence constraints, data dependencies, and core affinity, i.e. a task can be required to execute on a certain core. We describe a task thus by the tuple $\tau = \{C_i, T_i, D_i\}$, where C_i is the Worst Case Execution Time (WCET), T_i is the task period, and D_i the deadline. Additionally, let us define the set of cores by $\pi = \{\pi_1, \pi_2, \ldots \pi_m\}$. We now define a mapping $\mathcal{M}$ from the tasks to cores given by $\mathcal{M} : \Gamma \mapsto \pi$. We also define a network wide flow configuration $\mathcal{W} : (\sigma, \rho)$ at each of the routers in the set $\{\mathcal{R}\}$ given by $\mathcal{W} \mapsto \mathcal{R}$.

1) Given a network with a fixed flow-limit configuration $\mathcal{W} \mapsto \mathcal{R}$ and a mapping $\mathcal{M} : \Gamma \mapsto \pi$, with given traffic characteristics and timing requirements, determine if all the requirements can be satisfied.
2) Given a mapping $\mathcal{M} : \Gamma \mapsto \pi$, find a suitable flow-limit configuration $\mathcal{W} \mapsto \mathcal{R}$ so that the task requirements are met. An ILP based solution is provided in [2]. A low complexity solution, however, is still an open problem. A solution needs be scalable as well in order to cope with the large platforms expected in the near future.
3) Given a default flow-limit configuration $\mathcal{W} \mapsto \mathcal{R}$, find a suitable mapping $\mathcal{M} : \Gamma \mapsto \pi$ such that all the requirements of the application are met. This is the inverse case of 2). Such a scenario is important in dynamic systems. Applications need to be added during runtime, e.g. other applications should not be influenced. Additionally a reconfiguration of all traffic regulators during runtime is connected with increased overheard.
4) Given no flow-limit configuration, find a suitable task mapping $\mathcal{M} : \Gamma \mapsto \pi$ such that all the requirements of the application are met. This is the traditional problem of distributed systems. The regular NoC architecture and other characteristics of the platform can be exploited when focusing on many-cores. The problem complexity increases given the small and typically distributed memory close to the compute cores and the large delays when off-chip memory is accessed.

In all the above problems the additional constraints could be (i) presence of interdependent applications (ii) presence of mixed criticality applications (iii) fixed sized buffer lengths (iv) the need to form isolated islands to minimize interference to critical applications, (v) presence of heterogeneous cores.

In this paper, we highlighted some key problems in NoC based architectures that must be addressed before the deployment of real-time applications onto these platforms becomes possible. A paradigm shift from function centric to data and communication centric approaches is required. Combining hardware and software based flow-regulation seems to be the only way to ensure that NoCs go beyond the best-effort service and address the requirements of diverse applications.

References

[1] G. Heiser, "The role of virtualization in embedded systems," in *Proceedings of the 1st Workshop on Isolation and Integration in Embedded Systems (IIES)*, 2008, pp. 11–16.

[2] B. D. de Dinechin, Y. Durand, D. van Amstel, and A. Ghiti, "Guaranteed services of the NoC of a manycore processor," in *Proceedings of the International Workshop on Network on Chip Architectures (NoCArc'14)*, 2014, pp. 11–16.

[3] P. Munk, M. Freier, J. Richling, and J.-J. Chen, "Dynamic guaranteed service communication on best-effort networks-on-chip," in *23rdEuromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2015, pp. 353–360.

[4] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS)*, 2003, pp. 2–13.