Providing Flexible and Reliable on-Chip Network Communication with Real-Time Constraints

Eberle A. Rambo, and Rolf Ernst
Institute of Computer and Network Engineering
TU Braunschweig, Germany
{rambo|ernst}@ida.ing.tu-bs.de

***Abstract*— The same technology downscaling that enables Multiprocessor Systems-on-Chip (MPSoCs) has increased susceptibility to soft errors, giving rise to the so-called unreliable hardware. In this paper, we discuss the design of a central component of such architectures, the Network-on-Chip (NoC), and how to provide reliable on chip communication for real-time mixed-critical systems, applications with different requirements must be served, regarding e.g. time and reliability. We also discuss formal timing analyses for resilient MPSoCs architectures.**

I. Introduction

Technology scaling increases the hardware susceptibility to soft errors, giving rise to the so-called unreliable hardware. Soft errors are caused by alpha particles from package decay and energetic neutrons from electromagnetic radiation, which are abstracted as bit flips. System design for current and future technologies have to cope with soft errors [1] in order to provide the required reliability levels on each abstraction layer. This further complicates the task of providing response time guarantees, mandatory in the domain of real-time systems.

Techniques to overcome hardware induced errors in software execution can profit from the abundance of cores available in Multiprocessor Systems-on-Chip (MPSoCs) to increase reliability. For instance, software-based fault-tolerance approaches [2], [3] provide reliable execution on a higher level of abstraction without incurring overhead in hardware. Such fault-tolerance approaches assume the correct operation of some key components, both in software and hardware, denominated Reliable Computing Base (RCB) [4].

Figure 1 shows an example of an RCB in a software-based fault-tolerance solution with two types of protection service, replicated execution and checkpointing & rollback. For these approaches to be applicable and, at the same time, financially attractive, the RCB must be as small as possible and must be provided with the lowest overhead. We focus on the hardware part of the RCB, which consists of the inter-core communication, realized by a Network-on-Chip (NoC). Fault-tolerance solutions succeed as long as the communication with the applications works and is reliable. Losing contact with the application instances in a non-predictable manner would mean a system failure. Therefore, the NoC must be highly available and reliable in the presence of soft errors.

In this paper, we discuss the requirements for a resilient Network-on-Chip (NoC) design for use in mixed-critical real-time systems. In this domain, an essential aspect is guaranteeing that the system responds in time, as specified. These guarantees are provided by formal timing analyses, which calculate worst-case response time bounds for the tasks in the system. Since the NoC is a central component and also a heavily shared resource in the system, it plays an important role when providing response time guarantees. Therefore, the NoC must be predictable even under the occurrence of soft errors. Moreover, it has to allow tight worst-case bounds, which rules out many of the available solutions.

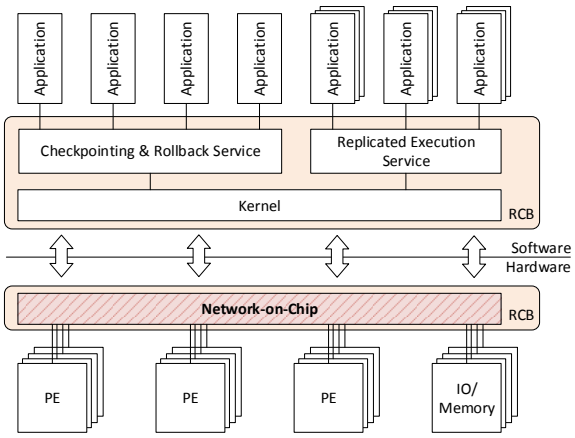

Fig. 1. Example of a Reliable Computing Base (RCB) of a software-based fault-tolerance solution for multiprocessor systems.

II. Reliability vs. Predictability

Many fault-tolerance approaches for increasing the reliability of NoCs have been proposed. The survey in [5] gives a good overview of the existing relevant work. Generally, retransmission protocols, such as Automatic Repeat reQuest (ARQ) [6], are used to correct corrupt data, which is detected using Error-Detecting Codes (EDCs), such as Cyclic Redundancy Check (CRC) [7]. The error detection and correction can be performed on an end-to-end basis, where the traffic is checked at the network interfaces, or on an hop-to-hop basis, where the traffic is also checked at each router. Usually the latter has been preferred. Hybrid schemes between the two have also been researched. Most of the approaches target packet-switched networks. Although faster wormhole-switched NoCs,

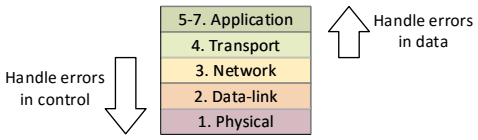

Fig. 2. OSI network model. Errors affecting the control of the network should be handled in the lower network layers. Errors affecting data should be addressed in the upper layers.

where packets are subdivided in Flow Control Units (flits), have also been considered, albeit in the context of general purpose computing.

Despite having been extensively researched, the operation of the router itself under errors has been overlooked. An FMEA analysis of a NoC implementation was performed in [8], [9]. The thorough analysis aims at preparing the NoC for use in real-time safety-critical systems, a domain that requires identifying all possible errors and their effects, which corresponds, in this case, to soft errors and their effects on the NoC routers and links.

The analysis identified several cases where soft errors (transient faults) result in static effects. A static effect caused by soft error means that, for instance, random blocking scenarios could occur in the NoC. The effect of blocking propagates backwards in the network in the form of backpressure and the NoC would only recover with a reset of the affected router(s) or even with a reset of the whole network, both with non-negligible impact on the system performance. Static effects are commonly associated with permanent faults due to similar impacts on the system. If not differentiated at design time, the occurrence of a transient fault with static effects will trigger the recovery mechanism for permanent faults (if implemented) or require a system reset.

Recovery mechanisms for permanent faults consist either of redundancy at design time (Modular Redundancy), mode change or dynamic solutions. One example of dynamic solution is deflective routing, a type of dynamic routing where the traffic is locally redirected to a neighbor router bypassing an unavailable or faulty router. Although well suited for general purpose systems, this class of techniques cannot be applied to real-time systems due to local decision making (dynamic), which drastically impairs the predictability and controllability of the system. Modular redundancy of hardware components (e.g. dual or triple – DMR or TMR) and mode change (or reconfiguration) are expensive features only implemented in systems requiring very high levels of reliability and an extended life. Moreover, those mechanisms have long recovery times, assuming that permanent faults are a rare occurrence. Handling a number of transient faults with these approaches leads to a very long response times in case of errors.

Transient faults should only cause transient effects. We argue that this must be handled in the lower layers of the NoC stack, illustrated in Figure 2. For the sake of reducing area overhead, we discard the use of fault-masking at the routers. In case of errors, corrupt packets are dropped e.g. because the routing data is corrupt and cannot be trusted anymore. This results in a highly available but lossy NoC under soft errors, providing a service comparable to the Internet layer in the TCP/IP protocol stack, where the layer does not guarantee packet delivery but guarantees the routing data integrity [6].

Additionally, the lower layers must provide sufficient independence between communication streams required in a mixed-critical context, which requires that errors do not propagate through different criticalities. This must be provided in the form of fault containment in the lower layers (up to the network layer). As a result of the fault containment, packets are dropped before they are able to affect the traffic from other tasks/criticalities.

On the top layers, transport protocols can provide reliable data transmission. It involves guaranteed packet delivery and guaranteed data integrity services. For instance by using retransmission protocols based on ARQ and CRC as checksum. However, this requires the NoC to be available (i.e. no static effects), otherwise retransmission protocols are ineffective [9]. Moreover, the transport protocol can be easily selected and configured by software in the network interface, according to the tasks' timing constraints and reliability requirements.

III. Integrated Timing Analysis under Soft Errors

Another essential aspect in the real-time domain is guaranteeing that the system responds in time. This is provided by formal timing analyses, which calculate worst-case response time bounds for the tasks in the system [10]. A simplified illustration is shown in Figure 3. The tasks, which implement functionalities of the system, are mapped to processing resources, the Processing Elements (PEs) in a multi-core. The tasks communicate with other tasks or access resources through the NoC. The analysis must model all relevant aspects of the system in order to provide bounds on the response time of those tasks. The figure abstracts the existence of shared-resources and off-chip communication.

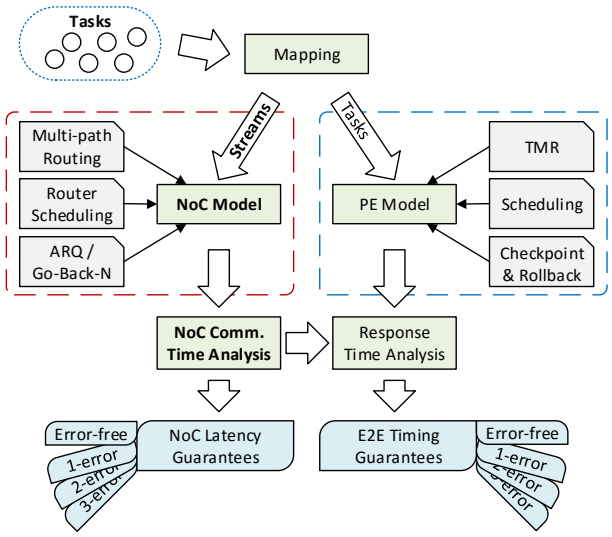

Fig. 3. Formal Timing Analysis flow considering the impact of fault-tolerance mechanisms on the NoC (left-hand side) and on the overall task execution (right-hand side).

The Response Time Analysis provides worst-case response times for the tasks in the system. It is referred to as end-to-end (E2E) response times in Figure 3 because it must enclose and bound all sources of interference, blocking and delays. This includes the scheduling used in each PE, the mapping, the latency to access resources (e.g. DRAM, flash memory) and the latency to communicate with other tasks. In a multi-core, calculating the latency of a communication or resource access is complex, as every communication leaving PE goes through the NoC, a shared resource. These latency bounds are provided by a separate analysis for the on-chip communication.

The Communication Time Analysis of the NoC [10] provides latency bounds for traffic streams on the chip. These can be e.g. cache line transfers, Direct Memory Access (DMA) transfers, sensor value, or a command for an actuator. The analysis must consider the transmission time and all interference that a packet suffers in the network. This is influenced e.g. by the topology of the network, the arbitration in the routers, Quality-of-Service (QoS) mechanisms and traffic classes. Naturally, it also depends on the pattern of the traffic injected by each PE in the system into the NoC.

The integration of fault-tolerance mechanisms and protocols impacts directly these analyses. In addition to the current models, the analysis must account for overheads generated by error detection and recovery, which come in various forms. For instance, ARQ handshaking in the NoC as well as checkpointing in the PEs have impact on the response time even in the error-free case and must be integrated in the respective models.

This causes the need for analyses, such as the ones presented by [10]–[12], to be merged and accurately account for the resulting impacts of errors on the whole system. The worst-case latency and the End-to-End (E2E) response time in different error scenarios, the k-error scenario, can be computed. The analysis of a k-error scenario considers the worst-case impact of k errors on the response time or latency. This tells the system designer whether the system is still able to meet its deadlines in the presence of k errors. The analysis considers that these errors occur inside the busy-window [11]. A realistic k can then be obtained by multiplying the busy-window length ($time$) by the expected error rate ($error \cdot time^{-1}$).

IV. Outline

We have discussed the design of a resilient and reliable Network-on-Chip for real-time mixed-critical systems. The devised solution discards the use of fault-masking in the router for the sake of low hardware overhead. Instead, resilient routers are proposed, where faults are allowed to become errors, whose effects are carefully limited and contained. The resulting network is resilient (i.e. highly available) and predictable under errors, but does not guarantee packet delivery. This is provided by transport protocols on an end-to-end basis. In a mixed-critical system, applications with different requirements must be served, regarding e.g. maximum latency, QoS, safety (freedom from interference). The approach allows flexibility in protecting traffic selectively according to the applications' requirements.

Acknowledgements

This work was partially funded by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 – spp1500.itec.kit.edu).

References

[1] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro, IEEE*, vol. 25, no. 6, pp. 10–16, Nov 2005.

[2] P. Axer, R. Ernst, B. Döbel, and H. Härtig, "Designing an analyzable and resilient embedded operating system," in *Proc. on Software-Based Methods for Robust Embedded Systems*, Braunschweig, Germany, 2012.

[3] B. Doebel and H. Hartig, "Can we put concurrency back into redundant multithreading?" in *Embedded Software (EMSOFT), 2014 International Conference on*. IEEE, 2014, pp. 1–10.

[4] M. Engel and B. Döbel, "The reliable computing base-a paradigm for software-based reliability." in *GI-Jahrestagung*, 2012, pp. 480–493.

[5] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks on chip," *ACM Comput. Surv*, vol. 44, 2012.

[6] A. Tanenbaum and D. Wetherall, *Computer Networks*. Pearson Prentice Hall, 2011.

[7] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 145–154.

[8] E. A. Rambo, A. Tschiene, J. Diemer, L. Ahrendts, and R. Ernst, "Failure Analysis of a Network-on-Chip for Real-Time Mixed-Critical Systems," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014.

[9] ——, "FMEA-Based Analysis of a Network-on-Chip for Mixed-Critical Systems," in *NOCS*, 2014.

[10] E. A. Rambo and R. Ernst, "Worst-case communication time analysis of networks-on-chip with shared virtual channels," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15, 2015, pp. 537–542.

[11] P. Axer, S. Quinton, M. Neukirchner, R. Ernst, B. Dobel, and H. Hartig, "Response-time analysis of parallel fork-join workloads with real-time constraints," in *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*. IEEE, 2013, pp. 215–224.

[12] P. Axer, D. Thiele, and R. Ernst, "Formal timing analysis of automatic repeat request for switched real-time networks," in *SIES*, Pisa, Italy, June 2014.