Adaptive Multi-Layer Techniques for Increased System Dependability

Bauer, Lars; Henkel, Jörg; Herkersdorf, Andreas; Kochte, Michael A.; Kühn, Johannes M.; Rosenstiel, Wolfgang; Schweizer, Thomas; Wallentowitz, Stefan; Wenzel, Volker; Wild, Thomas; Wunderlich, Hans-Joachim; Zhang, Hongyan

it - Information Technology Vol. 57(3) 8 June 2015

doi: http://dx.doi.org/10.1515/itit-2014-1082

Abstract: Achieving system-level dependability is a demanding task. The manifold requirements and dependability threats can no longer be statically addressed at individual abstraction layers. Instead, all components of future multi-processor systems-on-chip (MPSoCs) have to contribute to this common goal in an adaptive manner. In this paper we target a generic heterogeneous MPSoC that combines general purpose processors along with dedicated application-specific hard-wired accelerators, fine-grained reconfigurable processors, and coarse-grained reconfigurable architectures. We present different reactive and proactive measures at the layers of the runtime system (online resource management), system architecture (global communication), micro architecture (individual tiles), and gate netlist (tile-internal circuits) to address dependability threats.

Preprint

Adaptive Multi-Layer Techniques for Increased System Dependability

Lars Bauer, Jörg Henkel, Andreas Herkersdorf, Michael A. Kochte, Johannes Maximilian Kühn, Wolfgang Rosenstiel, Thomas Schweizer, Stefan Wallentowitz, Volker Wenzel, Thomas Wild, Hans-Joachim Wunderlich, Hongyan Zhang

Abstract: Achieving system-level dependability is a demanding task. The manifold requirements and dependability threats can no longer be statically addressed at individual abstraction layers. Instead, all components of future multi-processor systems-on-chip (MPSoCs) have to contribute to this common goal in an adaptive manner.
In this paper we target a generic heterogeneous MPSoC that combines general purpose processors along with dedicated application-specific hard-wired accelerators, fine-grained reconfigurable processors, and coarse-grained reconfigurable architectures. We present different reactive and proactive measures at the layers of the runtime system (online resource management), system architecture (global communication), micro architecture (individual tiles), and gate netlist (tile-internal circuits) to address dependability threats.

ACM CCS: Computer systems organization → Dependable and fault-tolerant systems and networks → Processors and memory architectures

Keywords: dependability, fault tolerance, graceful degradation, aging mitigation, online test and error detection, thermal management, multi-core architecture, reconfigurable architecture

1 Introduction

Achieving dependability by *"building dependable systems with non-dependable components"* already was a design goal for early computer systems [1] and it is currently pushed into new dimensions since the last decade [2]. The term *dependability* denotes a general concept comprising among others reliability, availability, and performability (i.e. the ability of the system to gracefully degrade in presence of faults) [3]. About a decade ago, dependability became a major design constraint after the ongoing technology scaling to nano-scale CMOS structures provided evidence that certain atomic-level effects – some of them known and measured for a long time – would start significantly impacting circuit functionality. These dependability threats comprise aging effects, thermal density, device variability, and increased susceptibility to soft errors. A comprehensive survey of different reliability threats and dependability approaches can be found in [4, 5].

In this paper, we target faults caused by aging effects. Aging is a change of the physical properties of structures on a chip, which leads to parametric degradations, performance variability over time, and eventually malfunctions [5]. Typical aging effects comprise *Hot-Carrier Injection* (HCI) and *Negative Bias Temperature Instability* (NBTI), detailed for instance in [4]. They affect individual transistors and typically lead to timing faults (lower transistor switching speed) and eventually to permanent faults (transistor not functional any more).

Additionally, thermal effects (like very high temperatures or thermal cycling) have a significant impact on how fast the chip structures age. Therefore, achieving high dependability is especially challenging for multi-processor systems-on-chip (MPSoCs). Their combination of cutting edge manufacturing technology and the resulting high integration and power density [6] exacerbate thermal effects and susceptibility to aging effects.

Figure 1 shows a typical heterogeneous MPSoC. Different components (so-called *tiles*) are connected via a network on chip (NoC). Applications executed on such architectures are typically composed of multiple tasks, where each task may be mapped on any of the tiles (not all tasks might be suitable for all tile types).

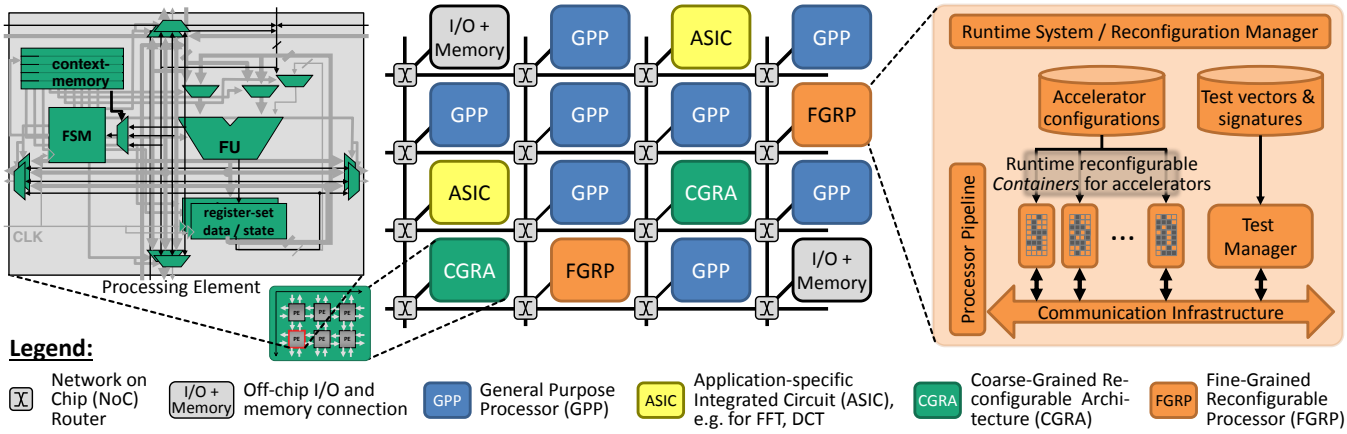

Figure 1: System Overview, showing an example how GPPs, FGRPs, and CGRAs are connected via a NoC; a zoom-out of a FGRP is shown on the right; a 2D array of processing elements (PEs) and a zoom-out of one PE is shown on the left

The MPSoC contains general purpose processors (GPP tiles) and access to peripherals and off-chip memory (I/O+Memory tiles). For some computationally complex kernels that are known to be demanded frequently (e.g. FFT, DCT), dedicated application-specific hardwired accelerators (ASICs in Figure 1) are provided. To accelerate applications that were not targeted initially, runtime reconfigurable tiles are integrated. In particular, the MPSoC contains fine-grained reconfigurable processors (FGRPs) and coarse-grained reconfigurable architectures (CGRAs) [7, 8].

FGRPs are processors along with a fine-grained reconfigurable fabric (like an embedded FPGA) that can be reconfigured with application-specific accelerators on demand. The right half of Figure 1 shows a zoom-out of a FGRP, where multiple runtime reconfigurable *Containers* are connected via a communication infrastructure to the processor pipeline and test facilities. A runtime system determines which accelerators are reconfigured into which containers, depending on applications' requirements. CGRAs are tightly coupled arrays of processing elements (PEs) that can be configured to implement application-specific accelerators (e.g. as replacement for an ASIC tile or like an additional ASIC tile). The left half of Figure 1 shows a 2D array of PEs (composing the CGRA) and a zoom-out of one PE. Each PE contains a functional unit (FU) along with a finite-state-machine (FSM) and context memory to control its operation and a register set to store temporary data.

To achieve dependability, all components have to contribute. For instance, the individual tiles can be optimized for dependability. But also the resource management and inter-tile communication need to consider dependability as optimization goal. The term *multi-layer dependability* denotes concepts, where dependability is not only treated at separate layers individually, but where additionally the layers interact to improve system-level dependability [9, 10, 11]. Typical layers in an MPSoC are the runtime system (online resource management), system architecture (global communication), micro architecture (individual tiles), and gate netlist (tile-internal circuits). Additionally, dependability can be treated at application layer, compiler/synthesis layer, or at physical device layer, but these layers are not in the focus of this paper.

Certain layers may allow more or less efficient handling of faults and in this paper we propose different *reactive* and *proactive measures* at different layers to address them. Reactive measures are meant to handle detected faults or errors in the MPSoC *after* detection. For instance, the FGRP and CGRA perform online test and online error detection of their reconfigurable fabric and the ASIC tiles to detect errors and to monitor the status of the hardware. Once a permanent fault was detected, they react and reconfigure themselves to avoid using the faulty resource or to replace it. Proactive measures are meant to control the MPSoC in a way that aims at avoiding faults *before* they occur, or masking resulting errors. For instance, the FGRP performs periodic online tests of all its reconfigurable resources, including those that are temporary unused, to ensure that the reconfigurable fabric is fault-free. Additionally, the FGRP performs aging mitigation and stress balancing by switching between different behaviorally equivalent configurations. At the system layer, tasks are proactively (re-)assigned to tiles in a way that avoids thermal hotspots before they appear. Correspondingly, communication channels are migrated in a way that makes task migration transparent to the application.

This work is part of the DFG funded priority program SPP 1500 "Dependable Embedded Systems"[1]. The paper covers three projects related to system- and hardware-architecture, namely VirTherm-3D (system-level task mapping and communication) [12, 13, 14], OTERA (FGRP) [15, 16, 17], and ARES (CGRA) [18, 19, 20, 21]. It is part of the IT special issue on "Embedded Systems". Other papers from SPP 1500 in this issue are: "Multi-layer software reliability for unreliable hardware" [22] and "Application-aware Cross-Layer Re-

[1] http://spp1500.itec.kit.edu/

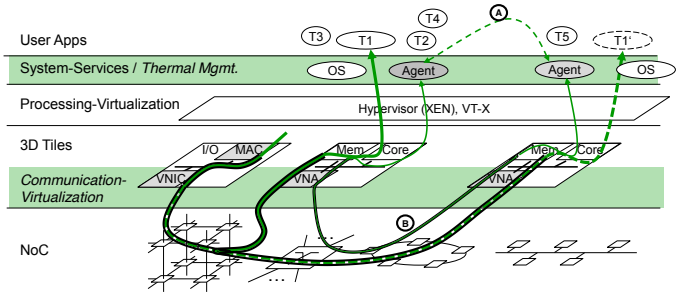

Figure 2: Thermal System Management and Communication Virtualization

liability Analysis and Optimization" [23].

Paper structure: Proactive and reactive measures for system-layer, FGRP, and CGRAs are presented in Sections 2 to 4, respectively. The feasibility of the concepts at the different layers is presented in the respective sections, and conclusions are drawn in Section 5.

2 Reliability at the system level

One critical dependability issue is thermal stress. Such stress can lead to transient faults in the short term or to accelerated aging in the long term. Thermal challenges worsen with the inception of 3D-integrated circuits, since the ratio of energy per surface area significantly increases. In the following we will discuss methods to tackle this issue on system level, either proactively by avoiding thermal hotspots or reactively on error occurrence [24, 25].

2.1 Proactive system level management: Agent system and communication virtualization

We propose to avoid thermal hotspots proactively through thermal management. The goal is to assign tasks to processing resources depending on their estimated impact on power dissipation and the actual monitored heat. If one part of the system is estimated to become too hot in the near future, then new tasks are started in other regions of the system or existing tasks are re-assigned at runtime.

We target platforms for applications with mixed criticality. The platforms have to cope different real-time demands and each application consists of a varying number of tasks as sketched in Figure 2. The tasks communicate explicitly via channels, such as in Kahn Process Networks or other data flow-oriented programming models.

Agent systems are a promising approach for the system management of future manycore platforms due to their superior scalability [12]. Our proposed hierarchical agent-based thermal management system initiates proactive task migration onto cooler processing resources. Every processing element is managed by one agent and these negotiate for power and tasks in a decentralized manner. The negotiation is inspired by a market-based economy.

The scope of our proposed system management goes beyond the avoidance and mitigation of thermal hotspots. Various aging effects also contribute to faults and must therefore be considered. Aging models are usually formulated on a very low level, e.g. transistor level, and need to be abstracted to be able to derive useful information, e.g. mean time to failure (MTTF), at the system level. We have done that and propose agents that mitigate aging at run time by keeping track of an aging budget and balancing the stress between different components at the microarchitecture to prevent premature failure of the system.

While dependable task migration is one important aspect, we identified the migration of the communication channels as an important performance and dependability factor. This involves both the communication to the outside world (see [13]) and inter-task communication (see [14]).

For our proposed communication migration we divided the migration into three phases: pre-migration, switchover and post-migration. While the switchover is a synchronization point between the task migration and communication migration, we propose to handle the communication migration separately from the task migration and transparently to the application.

This method is known as protection switching in the context of communication networks [26] and we propose to adapt this technique to on-chip communication to keep the downtime during task migration low and reduce the effects of the migration on the real-time behavior of the system. Instead of stopping the communication during all three phases, we instead forward messages from the old to the new task location or send the messages to both destinations during migration.

This communication migration actually reduces the peak latencies added by migration. Nevertheless we found that a pure software implementation creates additional stress on the processing elements, so that in the next steps we will integrate this in the communication hardware structure.

2.2 Reactive system level management: online testing and error detection

For many components, monitoring their utilization or allocating dedicated hardware to improve reliability is not economically feasible or sensible due to their low criticality. Instead, existing components in a MPSoC such as CGRAs can be used to periodically check such components and migrate their functionality once errors occurred. CGRAs realize functionality on a coarse, word-level granularity and thus achieve greater energy efficiency for most word-level accelerator functionality as for example FPGAs. With only as little information as

the component functionality, error free operation can be ascertained through online testing or error detection. To monitor components such as accelerator ASICs (cf. Figure 1) in an MPSoC, an online testing and error detection scheme has been developed to be used with a hardened CGRA [18]. The nomenclature used in this part is based on Gao et al. [27]. Both online testing and error detection are executed in a round robin time-multiplexed fashion. For online testing, input samples of the monitored components are recomputed on a shared, triple modular redundancy (TMR) secured CGRA. For online error detection, error detection is executed on the CGRA. Sharing the CGRA allows significant resource savings opposed to dedicated testing or error detection hardware.

The same test inputs are provided to the monitored component and the CGRA. If their results differ, it is assumed that an error occurred in the monitored component, as the CGRA is more reliable [19]. To do online checking or testing on the CGRA, it is reconfigured to implement the component's functionality or error detection algorithm. As typical for CGRAs, functionality can be mapped spatially and temporally. This means that individual operations that implement the functionality can be mapped to distinct PEs of the CGRA (spatial), or some operations can be mapped to the same PE (temporal) by using fast runtime reconfiguration (each PE can switch between multiple configuration contexts), or a combination of both. Thus, if the functionality does not fit spatially or if its implementation should occupy less PEs, then it can be mapped into the temporal domain. Thereby, throughput is traded for area, i.e. its implementation uses less resources, but requires more time steps to complete. This is expressed by the slow-down factor s. For instance, $s = 2$ means that area requirements are cut by half at the cost of typically increased execution time. For each of these online checks or tests, a time slot $T_{TW}[s]$ is allocated. From each of these T_{TW}, a reconfiguration overhead $T_{OV}[s]$ is subtracted. If an error occurred, it is detectable with probability q which may require several iterations of the check or test. Thus, for a certain fault in a component, we expect a specific temporal behavior, i.e. detection latency DL, probabilistic behavior, and confidence δ with which DL can be guaranteed. The spatial CGRA utilization can be reduced through s until the specified detection latency and confidence is just met.

Figure 3 depicts DL with δ error bars for four components sharing the CGRA running at 100MHz with slow-down factors ranging from 2 to 16. In this experiment each component is assigned a T_{TW} from $1ms$ to $1.55ms$ and T_{OV} of $1ms$. The occurred error is detectable with a probability of $q = 10^{-5}$. DL and δ steeply decrease with increasing T_{TW}. If more checks or tests can be conducted in a T_{TW}, the likelihood of detecting an error in the latter increases, reducing both DL and δ. If an error occurred, it can be ignored, noted or the affected functionality can be migrated to another instance of the component type or the CGRA.

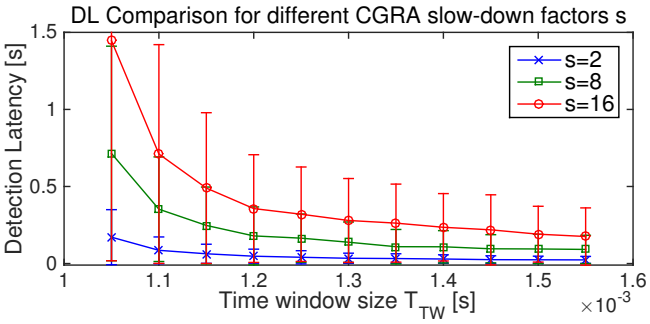

Figure 3: Online testing: Detection Latencies (DL) for various s, and T_{TW} with $T_{OV} = 1ms$ at 100MHz CGRA frequency

3 Reliability at the level of fine-grained reconfigurable processors

Aging in fine-grained reconfigurable fabrics can be mitigated by balancing the stress induced by accelerators over the available resources. First approaches executed a simple schedule of two accelerator configurations that use different logic resources of the fabric [28].

If the degradation due to aging progresses, intermittent and permanent faults may result. In fine-grained reconfigurable fabrics, these faults can be efficiently detected by online test and localized by online diagnosis [29] because of the highly regular structure of the fabric. In tile-based architectures supporting graceful degradation, the tile diagnosed faulty is not used for functional configurations any more [30].

To address aging-induced faults in fine-grained reconfigurable fabrics (the 'runtime reconfigurable Containers' in the FGRPs in Figure 1) and to increase system dependability, we employ a novel combined approach that comprises *pro-active* online tests at gate level and *reactive* architecture-level adaptation.

The online test methods consist of structural and functional tests of the fine-grained reconfigurable fabric before and after reconfiguration to detect emergent faults in cores as early as possible. In addition, the tests validate the error-free completion of the reconfiguration process. Periodic and on-demand functional tests are also conducted during normal system operation.

When a fault is detected and diagnosed, the architecture-level adaptation reacts accordingly. For instance, it avoids using the faulty region, triggers further tests of the region where the fault occurred (scheduled to execute during normal system operation), or it aims to use the partially faulty Container by reconfiguring those accelerator configurations to it that do not require the faulty parts for operation. This allows graceful degradation of the system with minimal performance impact.

Altogether, this combination of online testing and architecture-level adaptation allows for the first time *reliable runtime reconfiguration*, which is the foundation for system adaptation in FGRPs.

The dependability improvements in lifetime and reliability are achieved at very low hardware and performance cost by (i) evaluating online knowledge obtained from the runtime system and testing, (ii) flexible system adaptation (determining the accelerator configurations and dynamically rescheduling test runs and accelerator reconfigurations), (iii) utilizing partially faulty regions, and (iv) efficiently utilizing the combined infrastructure for test, runtime system, and application operation.

3.1 Proactive online tests

Two online test methods are combined to achieve reliable reconfiguration and high system dependability [15].

The Pre-Reconfiguration Test (PRET) checks for permanent faults in the fine-grained reconfigurable fabric by structurally testing regions of the fabric (called *Containers*, see Figure 1). The structural tests are partitioned in a set of test configurations, whereas each test configuration targets specific resources of the fabric, such as Lookup-Tables, Carry-Chains, or sequential elements. The hardware overhead for the generation and analysis of test patterns and responses is negligible. The test access and data transport is performed by reusing the existing communication infrastructure for Containers (cf. Figure 1). The test configurations for a Container are independent of each other and can be scheduled by the runtime system analog to functional configurations.

The Post-Reconfiguration Test (PORT) performs a functional test of the configured accelerators. The applied test stimuli are deterministically generated and application specific, i.e. they examine only the structures used by the respective accelerator. The sequence of applied stimuli can be partitioned into disjunct subsets of stimuli as long as the test response of each subset is independent of the accelerator's sequential state. These subsets are then scheduled by the runtime system. The communication infrastructure for Containers is used for test access and test data transport.

To ensure reliable reconfiguration of the hardware accelerators with minimal impact on performance, PRET and PORT have been tightly integrated into the runtime system and configuration scheduler. PRET tests exhaustively for structural faults in the fabric *prior* to the configuration of the desired hardware accelerator. *After* the configuration, PORT is applied periodically to test for faults in the Container interfaces and errors in its configuration bits that might have occurred after the configuration process. Periodic application of PRET and PORT by the runtime system provides different user-selectable trade-offs between fault detection latency and performance impact [15]. For instance, at a desired test latency of 3.8 seconds, only a marginal performance impact of at most 4.4% is induced. Once a fault is detected, it can be efficiently located by online diagnosis because of the regular structure of the fabric.

3.2 Reactive adaption for graceful degradation

Module Diversification [16] is a novel design method that enables fault tolerance to permanent and intermittent faults and to mitigate aging. For each hardware accelerator, multiple *diversified configurations*, which differ in their CLB usage, are created such that any single and a subset of multiple CLB faults in the containers can be tolerated by using an alternative accelerator configuration with identical function.

After a fault is detected in a Container, diagnosis will be performed on the faulty Container to determine the location of the fault(s). Knowledge about the required resources of the accelerator configurations and the specifically generated diversified configurations allow graceful system degradation in presence of single and multiple faults in the fabric: After selecting the accelerators to be configured, the runtime system selects and determines the placement of the respective diversified configurations. The configurations that do not require the faulty parts of the Container can still be configured into partially faulty Containers. In this way, the system performance is preserved.

3.3 Proactive stress mitigation

The diversified configurations exhibit different stress distributions among CLBs. This stress diversity is exploited to balance the stress among CLBs by optimally scheduling the operation time of each configuration [16]. Thus, stress is not concentrated on individual CLBs, and CLB aging is mitigated to increase lifetime.

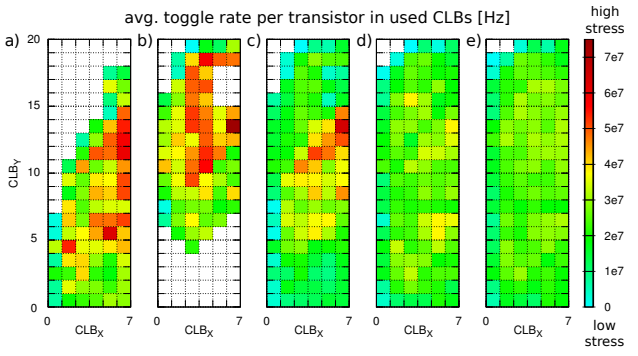

Figure 4: Switching activities of a), b) 2 max. diversified configurations, c) an alternating schedule thereof, d) a balanced scheduled with 4 configurations and e) 8 configurations [16]

The stress distributions in the CLBs of an accelerator using different configurations are shown in Figure 4. Figure 4a shows the switching activity, i.e. Hot-Carrier Injection (HCI) stress, in the CLBs used in the configuration without module diversification, with red values

for higher and blue values for lower activity, respectively. Figure 4b shows the switching activity for another configuration which is maximally diversified [16] w.r.t. a). Although configuration a) and b) use many different CLBs, there is still significant overlap between them. If both configurations are scheduled in a simple alternating manner, i.e. 50% time with configuration a) and 50% time with configuration b), the stress is not well balanced and the maximum stress is hardly reduced at all, as shown in Figure 4c. Figure 4d illustrates the significant reduction of the stress maxima when using four configurations and the *optimal* scheduling for stress balancing. With eight configurations the maxima can be reduced even further as shown in Figure 4e.

4 Reliability at the level of coarse-grained reconfigurable processors

CGRAs already contain many structures that allow realizing the paradigm shift towards high reliability embedded systems: reconfigurable, interchangeable and redundant components, a communication network, parallelism, and the tools to use this kind of architecture. However, current work focused only on using CGRAs as hardware accelerators. In our work, we widen the focus on using CGRAs as a reliability enhancer and exploit CGRAs' flexibility of fast reconfiguration to realize several functions in a single component:

- The CGRA can be used as a hardware accelerator until failures occur; that is, the reliability enhancing hardware is used from the very beginning, albeit for different purposes.
- When failures occur within other components, the CGRA is partially remapped, softly transitioning from hardware accelerator to full component replacement.
- When failures occur within the CGRA, the consequences are limited: we turn off individual failing PEs and remap their functionality within the CGRA.
- As the CGRA can dynamically take over virtually any functionality, it allows MPSoCs that can tolerate the failure of an initially unknown subset of their components.
- Instead of hardening every MPSoC component for increased reliability, we concentrate the hardening efforts onto few components that dynamically take over faulted functionality (cf. Section 2.2). This keeps the overall hardware overhead for the reliability small.
- Having a reliable CGRA, we also use it to monitor the functionality of other components.

4.1 Low-cost TMR

A common strategy to achieve fault tolerance is to introduce redundant logic by duplication with comparison (DWC) or by triple modular redundancy (TMR). The disadvantage of redundant hardware is its high hardware overhead and power consumption. To overcome these limitations we developed a strategy that exploits the built-in redundancy of CGRAs, utilizes spare FUs of a PE, and takes advantage of the fast reconfiguration mechanism of CGRAs to implement low-cost TMR.

We started by designing a flexible error handling module (FEHM). The FEHM monitors data to detect and mask permanent, transient and timing faults. Based on the FEHM we developed our low-cost TMR strategy [20] for CGRAs. Traditional TMR allows error detection and error masking. This can be implemented by adding two FUs and an FEHM to a Standard PE. In our FEHM PE-Cluster approach, TMR is realized by combining the FUs of three PEs in a fault tolerant data path instead of adding FUs to PEs. When FEHM is enabled, the FUs of the FEHM PE-Cluster form a single unit. However, when FEHM is disabled, each PE can also operate independent of the others. Our approach of an FEHM PE-Cluster is only viable if for each used FU two spare FUs are present. That means the ratio of spare FUs to used FUs, the redundancy ratio, must be equal or greater than two for each context. If, in the case of the FEHM PE-Cluster, the redundancy ratio is too small to find a valid TMR mapping for each context, we insert additional contexts by means of runtime reconfiguration to increase the number of spare FUs.

4.2 Dynamic remapping

One of the results of the FEHM implementation is to extend the capabilities of the CGRA to detect and to signal defects in PEs. The main goal of dynamic remapping is the development and evaluation of methods that – based on these signals – move (remap) functionality of defective PEs to unused, non-defective PEs and thereby allowing graceful degradation.

Since data paths are mapped to the CGRA within spatial and temporal domain, PEs are likely to be reused by different parts of the mapped data path. This has to be accounted for by updating all data paths that make use of the deactivated PE resources, a process referred to as remapping. To achieve this goal we have implemented a remapping flow that is based on our CGRA compiler.

The first task in our remapping flow [21] is the Architecture Correction. The system keeps an internal representation of the architecture and in case of a failure this representation is corrected such that it represents the current state of the architecture. Depending on the failure resources are removed from this representation. It is possible to remove a defective FU (FU-failure), defective register set (reg-failure), or, in case of other defective PE components, the entire PE (PE-failure).

Recompiling the entire application would be too time intensive. Therefore, we limit the remapping process to the PEs neighboring the defective PE. This is performed by the Subgraph Extraction stage. In the Schedu-

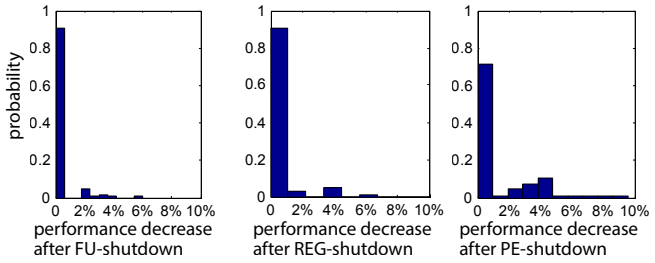

Figure 5: Probability histograms for the performance impact after remapping depending on the defective resource.

ling and Place & Route stages the actual mapping of the extracted application sub-graph is performed. Because the corrected architecture subgraph is targeted in this stage, the mapping process does not make use of any defective resource. Resources that were not needed in the previous mappings are now allocated in order to replace the defective components. The alternative mapping can potentially change the delay of the critical path. Based on a delay table the remapping process can estimate the new timing and, if necessary, the clock frequency is decreased accordingly in the Frequency Adaption stage. Finally, in the Configuration Generator stage the configuration data of the new mapping are generated. These data replace parts of the original configuration data in the system memory.

After integrating the remapping flow into our testing environment we performed experiments targeting a CGRA with 64 PEs and 8 contexts. The recovery from all possible single resource failures was evaluated. For instance, we mapped various signal processing algorithms (FIR filter, FFT, DCT and Raycasting) that on average required 87% of available registers, 74% of the FUs and 44% of the routing resources. Of course, the usage varies for each region in the PE array and hence, the complexity of the remapping problem strongly depends on which PE and what PE-component fails. For each FU, register, or entire PE of the array, a failure was simulated and the remapping engine activated to find a new valid mapping. Remapping might change the delay of some routings in such a way that the critical path changes. Figure 5 depict the probability that the performance is affected by a certain percentage. In most cases the performance is not affected at all. With a relatively small probability it is decreased by up to 10%.

5 Conclusion

In this paper we presented reactive and proactive measures at different layers to address dependability threats for future heterogeneous MPSoCs that combine general purpose processors along with dedicated application-specific hard-wired accelerators, fine-grained reconfigurable processors, and coarse-grained reconfigurable architectures. In particular, we focused on the layers of the runtime system, system architecture, micro architecture, and gate netlist.

By combining techniques of thermal-aware task mapping, communication virtualization, online testing, error detection, diversified accelerator configurations, and adaptive fault tolerance at these layers, we achieved stress mitigation and graceful degradation w.r.t. aging and permanent faults, which altogether improves system dependability.

Acknowledgement

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500 – `http://spp1500.itec.kit.edu`).

Literature

[1] J. von Neumann, *Probabilistic logics and synthesis of reliable organisms from unreliable components*, Automata Studies, pp. 43–98, 1956.

[2] S. Borkar, *Designing reliable systems from unreliable components: the challenges of transistor variability and degradation*, IEEE Micro, 25(6):10–16, 2005.

[3] A. Avizienis, J.-C. Laprie, B. Randell et al., *Basic concepts and taxonomy of dependable and secure computing*, IEEE Transactions on Dependable and Secure Computing, 1(1):11–33, 2004.

[4] J. Henkel, L. Bauer, N. Dutt et al., *Reliable on-chip systems in the nano-era: lessons learnt and future trends*, Design Automation Conference (DAC), 2013.

[5] J. Henkel, L. Bauer, J. Becker et al., *Design and architectures for dependable embedded systems*, IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 365–374, 2011.

[6] D. J. Frank, R. H. Dennard, E. Nowak et al., *Device scaling limits of Si MOSFETs and their application dependencies*, Proceedings of the IEEE, 89(3):259–288, 2001.

[7] H. P. Huynh and T. Mitra, *Runtime adaptive extensible embedded processors – a survey*, International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), pp. 215–225, 2009.

[8] H. Amano, *A survey on dynamically reconfigurable processors*, IEICE Transactions on Communications, 89(12):3179–3187, 2006.

[9] A. DeHon, H. Quinn, and N. Carter, *Vision for cross-layer optimization to address the dual challenges of energy and reliability*, Conference on Design, Automation and Test in Europe (DATE), pp. 1017–1022, 2010.

[10] N. P. Carter, H. Naeimi, and D. S. Gardner, *Design techniques for cross-layer resilience*, Conf. on Design, Autom. and Test in Europe (DATE), pp. 1023–1028, 2010.

[11] J. Henkel, L. Bauer, H. Zhang et al., *Multi-layer dependability: From microarchitecture to application level*, IEEE/ACM Design Automation Conference (DAC), 2014.

[12] T. Ebi, D. Kramer, W. Karl et al., *Economic learning for thermal-aware power budgeting in many-core architectures*, Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 189–196, 2011.

[13] H. Rauchfuss, T. Wild, and A. Herkersdorf, *Enhanced reliability in tiled manycore architectures through transparent task relocation*, Workshop on Dependability and Fault-Tolerance (VERFE) at ARCS, pp. 263–274, 2012.

[14] S. Wallentowitz, V. Wenzel, S. Rösch et al., *Dependable task and communication migration in tiled manycore system-on-chip*, Forum on Spec. and Design Lang., 2014.

[15] L. Bauer, C. Braun, M. E. Imhof et al., *Test strategies for reliable runtime reconfigurable architectures*, IEEE Transactions on Computers (TC), 62(8):1494–1507, 2013.

[16] H. Zhang, L. Bauer, M. A. Kochte et al., *Module diversification: Fault tolerance and aging mitigation for runtime reconfigurable architectures*, IEEE International Test Conference (ITC), pp. 1–10, 2013.

[17] H. Zhang, M. A. Kochte, M. E. Imhof et al., *GUARD: Guaranteed reliability in dynamically reconfigurable systems*, Design Automation Conference (DAC), 2014.

[18] J. M. Kühn, S. Eisenhardt, T. Schweizer et al., *Improving system reliability using dynamic functional verification on CGRAs*, Int'l Workshop on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART), 2012.

[19] T. Schweizer, P. Schlicker, S. Eisenhardt et al., *Low-cost TMR for fault-tolerance on coarse-grained reconfigurable architectures*, Int'l Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 135–140, 2011.

[20] T. Schweizer, A. Küster, S. Eisenhardt et al., *Using runtime reconfiguration to implement fault-tolerant coarse grained reconfigurable architectures*, Int'l Parallel and Distrib. Processing Symp. Workshop (IPDPSW), 2012.

[21] S. Eisenhardt, A. Küster, T. Schweizer et al., *Spatial and temporal data path remapping for fault-tolerant coarse-grained reconfigurable architectures*, IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011.

[22] M. Shafique, P. Axer, C. Borchert et al., *Multi-layer software reliability for unreliable hardware*, it - Information Technology, submitted, 2015.

[23] M. Glaß, H. Aliee, L. Chen et al., *Application-aware cross-layer reliability analysis and optimization*, it - Information Technology, submitted, 2015.

[24] H. Amrouch, V. van Santen, T. Ebi et al., *Towards inter-dependencies of aging mechanisms*, Int'l Conference on Computer-Aided Design (ICCAD), pp. 478–485, 2014.

[25] V. Narayanan and Y. Xie, *Reliability concerns in embedded system designs*, Computer, 39(1):118–120, 2006.

[26] J.-P. Vasseur, M. Pickavet, and P. Demeester, Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS, Elsevier, 2004.

[27] M. Gao, H.-M. Chang, P. Lisherness et al., *Time-multiplexed online checking*, IEEE Transactions on Computers (TC), 60(9):1300–1312, 2011.

[28] S. Srinivasan, R. Krishnan, P. Mangalagiri et al., *Toward increasing FPGA lifetime*, IEEE Transactions on Dependable and Secure Computing, 5(2):115–127, 2008.

[29] D. Milton, S. Dhingra, and C. E. Stroud, *Embedded processor based built-in self-test and diagnosis of logic and memory resources in FPGAs*, Int'l Conf. on Embedded Systems and Applications (ESA), pp. 87–93, 2006.

[30] A. Kanamaru, H. Kawai, Y. Yamaguchi et al., *Tile-based fault tolerant approach using partial reconfiguration*, International Workshop on Reconfigurable Computing (ARC), pp. 293–299, 2009.

Dr. Lars Bauer received the Dipl.-Inform and Dr.-Ing degrees from the University of Karlsruhe, Germany, in 2004 and 2009, respectively. He is currently a research assistant, lecturer, and group leader at the Karlsruhe Institute of Technology.

Address: Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES), E-Mail: lars.bauer@kit.edu

---

Prof. Dr. Jörg Henkel is directing the Chair for Embedded Systems (CES) at the KIT, Germany. He is an initiator and the spokesperson of the national priority program "Dependable Embedded Systems" of the German Science Foundation.

Address: Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES), E-Mail: henkel@kit.edu

---

Prof. Dr. Andreas Herkersdorf is head of the Institute for Integrated Systems at Technische Universität München. He is co-initiator of the national priority program "Dependable Embedded Systems" of the German Science Foundation.

Address: Technische Universität München, Institute for Integrated Systems, E-Mail: herkersdorf@tum.de

---

Dr. Michael A. Kochte received a Dr. rer. nat. (Ph.D.) degree from the University of Stuttgart in 2014. He is currently a researcher at the same university.

Address: Institut für Technische Informatik (ITI), Universität Stuttgart, E-Mail: kochte@iti.uni-stuttgart.de

---

Johannes Maximilian Kühn received his Dipl.-Inform. degree at the Eberhard Karls Universität Tübingen in 2012 and is since then a doctoral candidate under the guidance of Prof. Wolfgang Rosenstiel and Prof. Hideharu Amano (Keio University, Japan).

Address: Department of Computer Engineering, University of Tübingen, E-Mail: johannes-maximilian.kuehn@uni-tuebingen.de

---

Prof. Dr. Wolfgang Rosenstiel received his Ph.D. in 1984 from Karlsruhe University. Since 1990, he is Professor (Chair for Computer Engineering) at the Wilhelm-Schickard-Institute for Informatics at the University of Tübingen and serves currently as Dean of the Faculty of Science.

Address: Department of Computer Engineering, University of Tübingen, E-Mail: wolfgang.rosenstiel@uni-tuebingen.de

---

Dr. Thomas Schweizer received a Dr. rer. nat. (Ph.D.) degree from the University of Tübingen in 2010. He is currently a researcher at the same university.

Address: Department of Computer Engineering, University of Tübingen, E-Mail: thomas.schweizer@uni-tuebingen.de

---

Stefan Wallentowitz is a doctoral candidate working on efficient manycore inter-task communication.

Address: Technische Universität München, Institute for Integrated Systems, E-Mail: stefan.wallentowitz@tum.de

---

Volker Wenzel received a Diploma in Physics from the University of Mainz in 2013. He is currently a research assistant at the Karlsruhe Institute of Technology (KIT)

Address: Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES), E-Mail: volker.wenzel@kit.edu

---

Dr. Thomas Wild is member of the permanent scientific staff at the Institute for Integrated Systems of TU München. He works there on multi-/many-core architectures and their exploration on system level.

Address: Technische Universität München, Institute for Integrated Systems, E-Mail: thomas.wild@tum.de

---

Prof. Dr. Hans-Joachim Wunderlich is Director of the Institut für Technische Informatik (ITI), Universität Stuttgart. He has authored and co-authored more than 200 publications in the area of test, reliability, and fault tolerance.

Address: Institut für Technische Informatik (ITI), Universität Stuttgart, E-Mail: wu@informatik.uni-stuttgart.de

---

Hongyan Zhang received the M.Sc. in Electrical Engineering and Information Technologies from the Karlsruhe Institute of Technology in 2011. He is currently a research assistant at the same university.

Address: Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES), E-Mail: hongyan.zhang@kit.edu