Adaptive Embedded Computing with *i*-Core

Jörg Henkel, Lars Bauer, Artjom Grudnitsky and Hongyan Zhang
Karlsruhe Institute of Technology, Chair for Embedded Systems

{henkel, lars.bauer, grudnitsky, hongyan.zhang}@kit.edu

***Abstract*—Reconfigurable processors allow applications to access accelerators on a runtime-reconfigurable fabric (like an embedded FPGA) to speed-up execution. This work focuses on providing multi-tasking support and increasing dependability for reconfigurable processors. Two different task-scheduling strategies and multiple efficient dependability improvement schemes are presented and evaluated.**

I. Introduction

A reconfigurable processor consists of a regular processor core coupled with a reconfigurable fabric, allowing application-specific accelerators to be loaded at runtime. The processor core is implemented as an ASIC and the fine-grained reconfigurable fabric is implemented as an embedded FPGA, allowing a high degree of freedom in accelerator design and enabling applications from different domains to be run at a high performance on the same reconfigurable processor. Such platforms have been built in academia and industry (e.g. Xilinx Zynq-7000). To speed up execution of computationally intensive kernels, applications can use accelerators on the fabric via Special Instructions (SIs), which are extensions of the instruction set architecture (ISA) of the processor core.

A large portion of existing literature has focused on stand-alone reconfigurable processors, optimizing for single-tasking applications. However, while application-specific accelerators can provide significant speed-up in domains such as mobile computing (e.g. GSM, cryptography) and robotics (e.g. image/audio processing, feature matching), applications in these domains are typically composed of multiple tasks. Furthermore, systems for these applications are often dynamic multi-tasking systems, i.e. task arrival is not known at design time and task duration is dependent on input data (e.g. the recognized objects in a camera-based mobile robot). As long as such dynamically changing multi-tasking scenarios are not efficiently supported by reconfigurable processors, their inherent efficiency advantages are inaccessible for these demanding domains. The main focus of this work is therefore to enable efficient multi-tasking support and dependability improvement for reconfigurable processors by

- two task schedulers that are specifically optimized for runtime reconfigurable processors, one optimized for reducing the tardiness (i.e. the total time by which deadlines were missed over all tasks) and one for improving the makespan (i.e. completion time of the tasks), and
- online testing (ensure the fitness of the underlying reconfigurable fabric), fault tolerance and aging mitigation to increase system lifetime and adaptive runtime system (guarantees given reliability under varying soft-error rate while maximizing performance).

The paper was presented at APRES 2014.

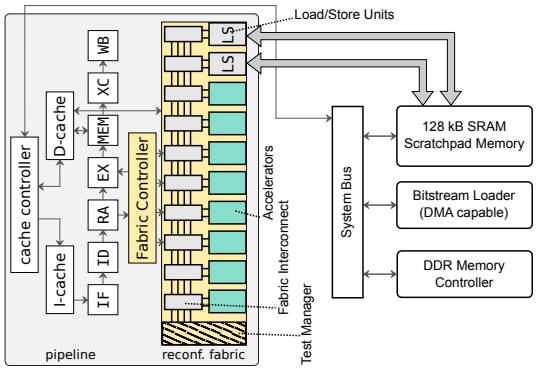

Fig. 1: *i*-Core SoC. *i*-Core extensions are shaded yellow. Based on [1].

Figure 1 shows the *i*-Core reconfigurable processor [8]. The core processor is a SPARC V8 in-order core. The mix-grained reconfigurable fabric [7] (fine-grained accelerators, coarse-grained interconnect) is tightly coupled to the core pipeline. SIs are multi-cycle instructions and generally include both arithmetic operations and memory access. The Fabric Controller manages SI execution by controlling accelerator modes, fabric interconnect and fabric-internal storage.

II. Extensions for Multi-Tasking

We now present a task-scheduler for **minimizing system tardiness** (i.e. the total time by which deadlines were missed over all tasks) for a given task-set. Tardiness reduction is important for application scenarios where at least some of the tasks have soft deadlines, e.g. reducing the tardiness for a video recording task leads to a reduced number of dropped frames. In [6] we present the Performance Aware Task Scheduling (PATS) strategy that aims to reduce tardiness of all running tasks. We use the notion of *task efficiency* in reconfigurable processors and observe that it changes over time for a task. A task that has just started executing a kernel (i.e. its required accelerators are not yet loaded on the fabric) will have a low task efficiency, while a task that has finished reconfiguring its accelerators will have a high efficiency and thus execute faster compared to having a low efficiency. Our scheduler favors executing tasks with high efficiency (unless it would lead to a deadline miss for a low-efficiency task), while tasks with low efficiency can perform their reconfigurations in parallel to that and thereby become high-efficiency tasks at a later time. PATS was compared with Earliest Deadline First (EDF), Rate-Monotonic Scheduling (RMS), and the scheduler used in the Molen processor[1] on reconfigurable processors with different fabric sizes and task-sets with different deadlines. PATS achieves a 1.45× better

[1] M. Sabeghi, V.-M. Sima, and K. Bertels, "Compiler assisted runtime task scheduling on a reconfigurable computer," in *Field Programmable Logic and Applications (FPL)*, 2009, pp. 44–50.

tardiness on average (max: 1.92×, min: 1.14× better) than the other schedulers.

For systems without deadlines, an important performance metric is the makespan (i.e. the completion time) of a taskset. To **improve the makespan** on a reconfigurable processor, we use MORP, a combined task scheduling and fabric allocation approach [1]. We observe that task efficiency is low after an application switches from one kernel to another (as it requires reconfiguration of accelerators for the new kernel), an effect we call Reconfiguration-induced Cycle Loss (RiCL). By reducing the RiCL of a taskset, we improve its makespan. The largest potential for RiCL reduction is in complex tasks that switch between different kernels during their execution time (e.g. video encoders). Such tasks are scheduled by our approach as *primary* tasks, which are initially assigned the full reconfigurable fabric. Using profiling and light-weight online adaptation, the approach predicts when the primary task will switch from one kernel to another, and a short time before that, a small share of the fabric is re-allocated to a *secondary* task. While accelerators of the secondary task are reconfigured, the primary task completes its current kernel. Upon switching to its next kernel, the task efficiency of the primary task drops (as its required accelerators are not yet available), and the system temporarily schedules the secondary task, which by this point has a higher efficiency than the primary task. The primary task reconfigures its accelerators while the secondary task is running, re-acquiring the fabric share of the secondary task at the end of its reconfigurations. At this point, the system schedules the primary task as its efficiency is high due to its completed configurations. Our approach achieves an average makespan reduction by 6.5% and 20.3%, compared to the SPT scheduler (Shortest Processing Time, optimal for makespan minimization on a non-reconfigurable processor) and the Molen scheduler, respectively.

III. Dependability in Reconfigurable Systems

In [4], a test scheme has been developed and integrated into the runtime system that schedules pre-configuration test (PRET) and post-configuration test (PORT) to ensure a reliable reconfiguration of the hardware accelerators with minimal impact on performance. PRET tests exhaustively for structural faults in the FPGA fabric prior to the instantiation of accelerators. After the instantiation, a periodic PORT test scheme is applied to test for faults in the container interfaces and errors in its configuration bits. The time a fault remains undetected in the system is reduced by up to two orders of magnitude at a marginal performance impact of at most 4.4%.

Module Diversification [5] is a design method that enables fault tolerance for permanent faults and aging mitigation. For each accelerator, multiple diversified configurations, which differ in their CLB usage, are created such that any single can be tolerated by using an alternative configuration. In addition, the stress diversity in diversified configurations is exploited to balance the stress among CLBs by optimally scheduling the operation time of each configuration. Stress reduction by up to 69% is achieved, which translates to an increase in lifetime by up to 222%.

GUARD [3] is a runtime system that guarantees the target reliability of SIs while optimizing the performance. At runtime, depending on the observed soft-error rate, the runtime system adaptively determines the scrubbing rate and the redundancy degree of accelerators (i.e. DWC or TMR for selected accelerators based on their vulnerability) for the SIs, such that performance is maximized for a given reliability constraint. Compared to related work which statically optimizes fault tolerance, GUARD provides up to 68.3% higher performance at the same target reliability.

IV. Conclusion

In this work we have presented 1) two task schedulers that improve soft-realtime capabilities and general performance on reconfigurable processors and 2) multiple dependability improvement schemes aiming to address different short-term and long-term dependability threats. In soft-realtime scenarios, the PATS scheduler achieves 1.45× better tardiness than comparable state of the art schedulers, while for makespan improvement our MORP scheduler achieves 6.5% to 20% improvement. Online testing reduces the time a fault remains undetected by up to two orders of magnitude at a marginal performance impact of at most 4.4%. With Module Diversification stress reduction by up to 69% and up to 222% lifetime increase are achieved. GUARD provides up to 68.3% higher performance at the same target reliability in comparison to statically optimized fault tolerance approach.

V. Acknowledgment

This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Invasive Computing" (SFB/TR 89). This work is supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500).

Selected Publications

[1] A. Grudnitsky, L. Bauer, and J. Henkel, "MORP: Makespan optimization for processors with an embedded reconfigurable fabric," in *Proceedings of the 22nd ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2014, pp. 127–136.

[2] M. Shafique, L. Bauer, and J. Henkel, "Adaptive energy management for dynamically reconfigurable processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 1, pp. 50–63, 2014.

[3] H. Zhang, M. A. Kochte, M. E. Imhof, L. Bauer, H.-J. Wunderlich, and J. Henkel, "GUARD: guaranteed reliability in dynamically reconfigurable systems," in *IEEE/ACM Design Automation Conference (DAC)*, accepted, 2014.

[4] L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, E. Schneider, H. Zhang, J. Henkel, and H.-J. Wunderlich, "Test strategies for reliable runtime reconfigurable architectures," *IEEE Transactions on Computers (TC), Special Section on Adaptive Hardware and Systems*, vol. 62, no. 8, pp. 1494–1507, 2013.

[5] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, and J. Henkel, "Module diversification: fault tolerance and aging mitigation for runtime reconfigurable architectures," in *IEEE International Test Conference (ITC)*, 2013, pp. 1–10.

[6] L. Bauer, A. Grudnitsky, M. Shafique, and J. Henkel, "PATS: a performance aware task scheduler for runtime reconfigurable processors," in *Field-Programmable Custom Computing Machines (FCCM)*, 2012, pp. 208–215.

[7] A. Grudnitsky, L. Bauer, and J. Henkel, "Partial online-synthesis for mixed-grained reconfigurable architectures," in *Design Automation and Test in Europe Conference (DATE)*, 2012, pp. 1555–1560.

[8] J. Henkel, L. Bauer, M. Hübner, and A. Grudnitsky, "i-Core: a run-time adaptive processor for embedded multi-core systems," in *Engineering of Reconfigurable Systems and Algorithms (ERSA)*, 2011.