DFG Sonderforschungsbereich/Transregio 89
Transregional Collaborative Research Center 89

Invasive Computing
Annual Report 2019

Friedrich-Alexander-Universität Erlangen-Nürnberg
Karlsruher Institut für Technologie
Technische Universität München

Transregional Collaborative Research Centre 89

Invasive Computing

Friedrich-Alexander-Universität Erlangen-Nürnberg
Karlsruher Institut für Technologie
Technische Universität München

Annual Report 2019

Coordinator
Prof. Dr.-Ing. Jürgen Teich
Lehrstuhl für Informatik 12
Friedrich-Alexander-Universität Erlangen-Nürnberg
Cauerstraße 11
91058 Erlangen

Managing Director
Dr.-Ing. Jürgen Kleinöder
Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstraße 1
91058 Erlangen

Administration, Management, and Public Relations
Dr. Sandra Mattauch
Stefanie Kugler
Lehrstuhl für Informatik 12
Friedrich-Alexander-Universität Erlangen-Nürnberg
Cauerstraße 11
91058 Erlangen

Preface

This report summarises the activities and scientific progress of the Transregional Collaborative Research Centre 89 “Invasive Computing” (InvasIC) in 2019.

The CRC/Transregio “Invasive Computing” is funded by the Deutsche Forschungsgemeinschaft in its third funding phase from July 2018 – June 2022. The research association aggregates about 60 of the best researchers from three excellent sites in Germany (Friedrich-Alexander-Universität Erlangen-Nürnberg, Karlsruher Institut für Technologie and Technische Universität München). This scientific team includes specialists in algorithm engineering for parallel algorithm design, hardware architects for reconfigurable MPSoC development as well as language, tool and application, and operating-system designers.

A special highlight in 2019 was the DATE conference in Bologna. Here, on March 26, our CRC/Transregio 89 organised a special session on “DFG Collaborative Funding Instruments” with Dr. Raabe from DFG introducing the different DFG funding instruments, followed by individual presentations of diverse currently funded initiatives in the area of computer science. These initiatives also presented their work during the whole week at booths as part of the conference exhibition.

As another highlight, you might want to take seven minutes to watch our new YouTube video episode “Invasive Computing for Experts”[1]?

Like in the previous year, we would like to thank all members of the CRC/Transregio “Invasive Computing” and all our partners from industry and academia for the fruitful collaborations and inspiring discussions in the last year! We do hope that you will enjoy reading about the progress the CRC/Transregio 89 achieved in 2019, as well as about our research planned for the future.

Jürgen Teich
Coordinator

[1] https://www.youtube.com/watch?v=p-TPiCES9cc

Contents

I

Invasive Computing

1 About InvasIC

The Idea of Invasive Computing

Our CRC/Transregio systematically investigates the novel paradigm of *invasive computing* for designing and programming future parallel computing systems. For systems with 1,000 or more cores on a chip, *resource-aware programming* is of utmost importance to obtain high *utilisation* as well as *computational* and *energy* and *power efficiency*. With this goal in mind, invasive computing was introduced to provide a programmer explicit handles to specify and argue about *resource requirements* desired or required in different phases of execution: In an *invade* phase, an application asks the operating system to allocate a set of processor, memory and communication resources to be *claimed*. In a subsequent *infect* phase, the parallel workload is spread and executed on the obtained claim of resources. Finally, if the degree of parallelism should be lower again, a *retreat* operation frees the claim again, and the application resumes a sequential execution. To support this idea of *self-adaptive* and *resource-aware programming*, not only new programming concepts, languages, compilers, and operating systems need to be developed, but also revolutionary architectural changes in the design of MPSoCs (multiprocessor systems-on-a-chip) to efficiently support invasion, infection, and retreat operations.[2] This includes new architectural concepts for a dynamic processors, interconnects, and memory reconfiguration, to give some examples.

Necessity and First Achievements

As predicted at the start of our journey in 2010, we will see more than 1,000 processor cores integrated on a single chip in 2022.[3] Yet, programming such large-scale processor systems is a nightmare if resource awareness is a must and certain execution qualities must be guaranteed. Using invasive computing, a programmer may specify resource requirements and, if available, the application will obtain as many exclusive resources to deliver a desired quality of execution. This *dynamic* and *application-driven isolation* is unique. Starting off from scratch in terms of invasive processor hardware, language, compiler, and operating system, we have genuinely fostered the *fundamentals of invasive computing*

[2] This focus on investigations on invasive MPSoCs has inspired us to give our CRC the acronym InvasIC, see `http://www.invasic.de` for more details.

[3] Some GPU devices already having surpassed this number today!

in the first funding phase: These include the definition of programming language elements for invasion primitives as well as constraints to argue about number, types, and state of resources that may be invaded (the *invasive command space*, project area A). A first language based on the programming language X10 by IBM as well as a compiler for translation of invasive X10 programs (project area C) onto invasive multi-tile architectures (investigated by project area B) and a run-time system (*i*RTSS) for managing their execution is available. Invasive applications exploiting different types of processor and communication resources of an invasive network-on-chip (*i*NoC) have shown considerable gains in resource utilisation and efficiency in comparison with their non-invasive counterparts.

Predictability. Or: Sharing is Not Caring!

By the fact that resources are temporally claimed (by default) in an exclusive manner, interference by other applications due to resource sharing may be reduced if not avoided completely. This *isolation*, combined with *run-to-completion* as the default mode of thread execution and bandwidth guarantees on communication links, allow us to provide predictable quality-of-service (QoS) also for communication. In the second funding phase, we played out this ace systematically by tackling (a) *predictability* of (b) *multiple execution qualities* of parallel invasive programs and including their (c) *mapping optimisation*. Our recent findings include new language constructs to define so-called *requirements* on desired, respectively amended qualities of execution. Addressed qualities include performance (e. g. execution time, throughput, etc.), *security* and *fault tolerance*. Through the analysis of application requirements from different domains including stream processing and malleable task applications, not only efficiency but also predictable execution qualities can now be demonstrated for applications stemming from robotics, imaging, as well as HPC. As another new yet very important facet of invasive computing, a particular focus of the second funding phase was devoted to the problem of *dark silicon* and *energy-* and *power-efficient computing*.

The Missing Link: Beating Run-Time Uncertainties and Run-Time Requirement Enforcement

The isolation gained by invasive computing is essential to establish *composability*. This, in turn, paves the way for an independent and static analysis of individual program qualities in dependence of only resource

claim properties, giving an unprecedented gain in predictability. Yet, even if this **-predictability*[4] (boundedness of any of the above non-functional properties through the invasion of resources) can be shown to hold, (a) the *effective bounds* (either lower or upper) as well as (b) their *variability* might still be too big or too coarse to be desirable or affordable in practical application fields such as embedded real-time control. Also, claiming resources exclusively might keep these either *underutilised* (in case of low application workload demands) or *inefficiently used* (e. g. when running a claim always at maximal processor speeds) in order to safely guarantee timing bounds also for the worst-case input.

Our current third funding phase is therefore dedicated to the missing link: Beating the uncertainty caused by variation of *program input*, *machine state* and *environment* at run time. The envisioned solution: *Run-time requirement enforcement*. Formally, we want to investigate *hybrid techniques* combining (a) static analysis of the robustness of desired qualities in dependence of input and state fluctuations and (b) systematic generation of suitable *run-time requirement enforcers (RRE)* (additional code that either *locally* or *globally* observes and controls the satisfaction of requirements in respective *corridors* at run time). This also includes the generation of necessary program-specific *run-time requirement monitors (RRM)*. With these techniques, we want to reach our final goals and vision formulated already at the beginning of our mission: *Invasive computing will be a—if not the—vehicle for providing* resource awareness *for a mixture of* best-effort *and* predictable quality *applications. We do believe huge application and business fields in embedded systems will become accessible for multicore technology through the foundations of invasive computing.*

[4] J. Teich et al. “Language and Compilation of Parallel Programs for *-Predictable MPSoC Execution using Invasive Computing”. In: *Proceedings of the 10th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. Lyon, France, Sept. 21–23, 2016, pp. 313–320. DOI: 10.1109/MCSoC.2016.30.

2 Participating University Groups

Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Hardware-Software-Co-Design

- Prof. Dr.-Ing. Jürgen Teich
- PD Dr.-Ing. Frank Hannig
- Dr.-Ing. Stefan Wildermann

Lehrstuhl für IT-Sicherheitsinfrastrukturen

- Prof. Dr.-Ing. Felix Freiling

Lehrstuhl für Verteilte Systeme und Betriebssysteme

- Prof. Dr.-Ing. Wolfgang Schröder-Preikschat
- Dr.-Ing. Timo Hönig

Karlsruher Institut für Technologie

Institut für Anthropomatik und Robotik

- Prof. Dr.-Ing. Tamim Asfour

Institut für Programmstrukturen und Datenorganisation

- Prof. Dr.-Ing. Gregor Snelting

Institut für Technik der Informationsverarbeitung

- Prof. Dr.-Ing. Jürgen Becker

Institut für Technische Informatik

- Prof. Dr.-Ing. Jörg Henkel
- Dr.-Ing. Lars Bauer

Technische Universität München

Lehrstuhl für Entwurfsautomatisierung

- Prof. Dr.-Ing. Ulf Schlichtmann
- Prof. Dr.-Ing. Daniel Müller-Gritschneder

Lehrstuhl für Integrierte Systeme

- Prof. Dr. sc. techn. Andreas Herkersdorf
- Prof. Dr.-Ing. Walter Stechele
- Dr.-Ing. Thomas Wild

Lehrstuhl für Rechnerarchitektur & Parallele Systeme

- Prof. Dr. Michael Gerndt

Lehrstuhl für Wissenschaftliches Rechnen

- Prof. Dr. Hans-Joachim Bungartz
- Prof. Dr. Michael Bader

Universität Bremen

Arbeitsgruppe für kombinatorische Optimierung und Logistik

- Prof. Dr. Nicole Megow

II

Research Program

3 Overview of Research Program

To investigate the main aspects of invasive computing, the CRC/Transregio is organised in five project areas:

Area A: Fundamentals, Language and Algorithm Research
Research in project area A focuses on the basic concepts of invasion and resource-aware programming as well as on language issues, algorithmic theory of invasion and on analysis and optimisation techniques for application characterisation and hybrid (mixed static/dynamic) core allocation.

Area B: Architectural Research
Project area B investigates micro- and macroarchitectural requirements, techniques and hardware concepts to enable invasive computing in future MPSoCs.

Area C: Compiler, Simulation, and Run-Time Support
The focus of project area C is on software support for invasive computing including compiler, simulation and operating-system functionality as well as on design space exploration with a special focus on run-time management.

Area D: Applications
Applications serve as demonstrators for the diverse and efficient deployment of invasive computing. The applications have been chosen carefully from the domains of robotics and scientific computing in order to demonstrate distinct and complementary features of invasive computing, for example its capability to provide quality-predictable execution of parallel programs.

Z2: Validation and Demonstrator
A hardware demonstrator will serve again as the key concept for validation of invasive computing principles. It will allow for co-validation and demonstration of invasive computing through tight integration of hardware and software research results and to decide on the further roadmap of specific hardware for invasive computing.

Four working groups **Run-Time Requirement Monitoring and Enforcement**, **Memory Models, Architecture and Management**, **Benchmarking and Evaluation** and **Power and Thermal Aspects** defined on top of these project areas support the interdisciplinary research.

Research Area	Project	
A: Fundamentals, Language and Algorithm Research	Basics of Invasive Computing *Prof. Dr.-Ing. G. Snelting, Prof. Dr.-Ing. J. Teich*	A1
	Characterisation and Analysis of Invasive Algorithmic Patterns *Prof. Dr. M. Bader, Dr.-Ing. S. Wildermann*	A4
	Scheduling Invasive Multicore Programs Under Uncertainty *Prof. Dr. N. Megow*	A5
B: Architectural Research	Adaptive Application-Specific Invasive Micro-Architectures *Dr.-Ing. L. Bauer, Prof. Dr.-Ing. J. Becker, Prof. Dr.-Ing. J. Henkel*	B1
	Invasive Tightly-Coupled Processor Arrays *Prof. Dr.-Ing. J. Teich*	B2
	Power-Efficient Invasive Loosely-Coupled MPSoCs *Prof. Dr.-Ing. J. Henkel, Prof. Dr. sc. techn. A. Herkersdorf*	B3
	Generation of Distributed Monitors and Run-Time Verification of Invasive Applications *Prof. Dr.-Ing. D. Müller-Gritschneder, Prof. Dr.-Ing. U. Schlichtmann*	B4
	Invasive NoCs and Memory Hierarchies for Run-Time Adaptive MPSoCs *Prof. Dr.-Ing. J. Becker, Prof. Dr. sc. techn. A. Herkersdorf*	B5
C: Compiler, Simulation, and Run-Time Support	Invasive Run-Time Support System (*i*RTSS) *Dr.-Ing. L. Bauer, Prof. Dr.-Ing. J. Henkel, Dr.-Ing. T. Hönig, Prof. Dr.-Ing. W. Schröder-Preikschat*	C1
	Compilation and Code Generation for Invasive Programs *Prof. Dr.-Ing. G. Snelting, Prof. Dr.-Ing. J. Teich*	C3
	Security in Invasive Computing Systems *Prof. Dr.-Ing. F. Freiling, Prof. Dr.-Ing. W. Schröder-Preikschat, Prof. Dr.-Ing. G. Snelting*	C5
D: Applications	Invasive Software-Hardware Architectures for Robotics *Prof. Dr.-Ing. T. Asfour, Prof. Dr.-Ing. W. Stechele*	D1
	Invasive Computing and HPC *Prof. Dr. M. Bader, Prof. Dr. H.-J. Bungartz, Prof. Dr. M. Gerndt*	D3
Z: Administration	Validation and Demonstrator *Prof. Dr.-Ing. J. Becker, PD Dr.-Ing. F. Hannig, Dr.-Ing. T. Wild*	Z2
	Central Services *Prof. Dr.-Ing. J. Teich*	Z
WG: Working Groups	Run-Time Requirement Monitoring and Enforcement *Prof. Dr.-Ing. F. Freiling, Dr.-Ing. T. Hönig, Prof. Dr.-Ing. D. Müller-Gritschneder*	WG1
	Memory Models, Architecture and Management *Prof. Dr. sc. techn. A. Herkersdorf, Prof. Dr.-Ing. W. Schröder-Preikschat, Prof. Dr.-Ing. G. Snelting*	WG2
	Benchmarking and Evaluation *Prof. Dr. M. Gerndt, Prof. Dr.-Ing. W. Stechele*	WG3
	Power and Thermal Aspects *Prof. Dr.-Ing. J. Henkel, Prof. Dr. N. Megow, Dr.-Ing. S. Wildermann*	WG4

4 Research Projects

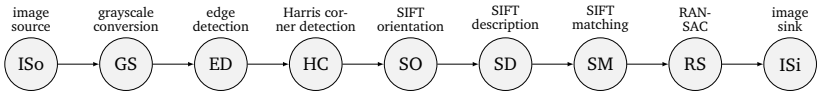
A1: Basics of Invasive Computing

Gregor Snelting, Jürgen Teich

Joachim Falk, Frank Hannig, Pouya Mahmoody, Behnaz Pourmohseni, Tobias Schwarzer, Maximilian Wagner, Stefan Wildermann

The goal of Project A1 is to develop the theoretical foundations for *Run-Time Requirement Enforcement (RRE)* of invasive programs and to investigate the formal tractability of invasive X10 programs. The research in Project A1 focuses on (a) establishing the theory and semantics of RRE, (b) development of central and distributed RRE techniques, (c) development of strict and loose RRE techniques in support of hard and soft non-functional requirements, respectively, and (d) applying theorem provers to formalise the semantics of invasive X10 programs.

In 2019, Project A1 has particularly focused on developing the theory of RRE and investigating the practice of different classes of RRE techniques, namely, centralised vs. distributed and strict vs. loose. Together with Project A4, we have developed an isolation-aware application characterisation and timing analysis approach that enables a predictable execution of invasive application programs on tiled manycore architectures under arbitrary combinations of inter-application temporal/spatial isolation schemes [Pou+19b]. Moreover, in collaboration with Project A4 and Project B3, we have developed a thermally composable Hybrid Application Mapping (HAM) approach that enables a thermally safe execution and, thereby, an uninterrupted enforcement of timing requirements for real-time applications [Pou+19a]. Finally, in cooperation with Project A5, we studied the complexity and suitable algorithms for run-time mapping and scheduling of task graphs (DAGs) on multicore architectures [Sim+20]. The following presents a selection of the results of our research in 2019.



A1

Figure 4.1: Actor graph of an object detection algorithm chain with often strongly input-dependent workload. Without RRE techniques, the end-to-end latency of execution might vary enormously from frame to frame.

Run-Time Requirement Enforcement (RRE)

For the study of different classes of RRE techniques, we considered the enforcement of soft- and hard timing requirements for streaming, e. g. image processing, applications. In the following, we present an illustrative case study of an object detection application, composed of nine actors which process a stream of input images in a pipelined fashion, see Figure 4.1. Let the timing requirements be given as an upper bound UB_L on the end-to-end latency of each execution of the application program. In order to avoid any overreservation or underutilisation of claimed resources in reaction to unknown input workload to be processed by each actor for each frame, we investigated *application-level* enforcement support where one (centralised) or more (distributed) *RRE actors* with special system privileges are introduced into the actor graph to enforce the given timing requirement by adjusting a set of so-called *enforcement control knobs*. In the above image-processing case study, the RRE actors are privileged to adjust the number of active cores $n \in [1, 4]$ claimed for each enforced actor and the dynamic voltage and frequency scaling (DVFS) mode $m \in [1, 20]$ of these cores on a per-input basis.

The first step towards achieving an enforced execution of invasive programs is to develop an *enforcement strategy*, in principle the functionality of the RRE actor(s). The enforced execution of each actor is then achieved by transforming the actor graph to also comprise the RRE actor(s). At run time, these actors establish an enforced execution of the program by adjusting the enforcement control knobs per input before the enforced actor(s) begin to process it.

Enforcement Strategy Development

The enforcement strategy of an RRE actor describes how it adjusts the value of its control knobs in the event of input variation to enforce a given set of requirements. As shown in Figure 4.2, our approach for developing enforcement strategies involves two main steps: (a) *profiling*

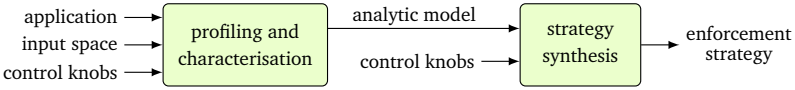


Figure 4.2: The flow of developing an enforcement strategy. First, in a profiling and characterisation step, an analytic model is derived which formally describes the non-functional properties of each or a set of enforced actors as a function of its input and RRE control knobs. Based on that, in the strategy synthesis step, (Pareto-)optimal control knob settings for different input scenarios are derived which constitute the enforcement strategy.

and characterisation of the non-functional properties of interest followed by (b) an *RRE strategy synthesis* step.

In the first step, we identify how the input uncertainty affects the property (here, timing) to be enforced. This can be achieved, e. g. by means of input sampling and application profiling [RHT19]. Table 4.1 summarises the result of such a profiling for our case study where the execution time of each actor, running on a single core at maximum frequency, is recorded for a test sequence of 9,149 images. The results denote that, as opposed to other actors, the execution times of the SD and SM actors contribute significantly to the latency of the application (see their overall latency contributions) and strongly depend on the content of each input image (see the standard deviation of their latency), making them promising candidates for enforcement.

After the candidate actors for enforcement are identified, an analytic model is established which formally describes the non-functional behaviour of each selected actor as a function of some feature(s) of its input and the setting of the enforcement control knobs that can be adjusted by the RRE actor. In the image-processing case study, the number i of corners in an input image (which is extracted by the HC actor) provides a suitable index of the workload introduced by that image for each enforced actor. Therefore, our analytic model derives the execution latency of each actor, SD and SM, as a function $L(i, n, m)$ of the number i of corners in an input image, the number n of active cores decided by the RRE and the selected DVFS mode m for them.

Table 4.1: Summary of the recorded execution times of different actors (each running on a single core at maximum frequency) when processing a test sequence of 9,149 input images.

actor	GS	ED	HC	SO	SD	SM	RS
average latency [ms]	0.21	0.18	1.50	1.79	146.86	21.02	0.01
latency std. deviation [ms]	0.09	0.08	0.64	0.80	106.15	15.04	0.03
overall latency contribution	0.1%	0.1%	0.9%	1.0%	85.6%	12.3%	0.0%

Once the enforcement analytic model has been derived, the second step of developing the enforcement strategy is performed, namely, the *RRE strategy synthesis*, see Figure 4.2. Here, provided the enforcement analytic model and the enforcement control knobs (decision space of the RRE actors), Design Space Exploration (DSE) [Sch+19b] can be performed to identify the (Pareto-)optimal control knob settings for different input scenarios [Spi+19; SWT19]. In our case study, the space of possible values of i (number of corners) in the input images is subdivided into intervals, each denoting one input scenario and corresponding to one (Pareto-)optimal control knob setting. These intervals and their respective (n, m) setting constitute the enforcement strategy which can be stored as a look-up table to be used by the RRE actor at run time although more complex and state-dependent enforcement strategies can be thought of.

In [Tei+20a], we have discussed details of the analyses and techniques used in the two steps of enforcement strategy development. There, we have also introduced the concept of *requirement strictness* which denotes the minimum rate $s \in (0, 1]$ of requirement satisfaction that must be achieved through enforcement, specified by the user. In [Tei+20a], we also demonstrated how the requirement strictness can be incorporated into the profiling and characterisation step surveyed above to systematically construct enforcement strategies that deliver the demanded requirement satisfaction rate while using the provided relaxation of loose requirements (compared to the strict case with $s = 1$) to optimise secondary objectives e. g. energy efficiency. In [Tei+20b], we have shown how apart from latency requirements, also power and reliability requirements can be enforced systematically.

Enforcement Implementation

Enforcement can be realised in a distributed or a centralised fashion. In *centralised enforcement*, a single RRE actor is generated for the whole program to establish an enforced execution based on its global knowledge on the current state of the program and its claimed resources. In *distributed enforcement*, multiple RRE actors are generated, each enforcing a *sub-region* in the requirement interval for one part of the program, e. g. an actor or a tile, based on local knowledge on the respective program part and invaded hardware region (claim). In the following, we present the implementation of distributed and centralised enforcement schemes, exemplified for our case study. To that end, the timing requirement is assumed to impose a latency upper bound of $UB_L = 115\,\text{ms}$ from which we dedicate $20\,\text{ms}$ to the non-enforced actors

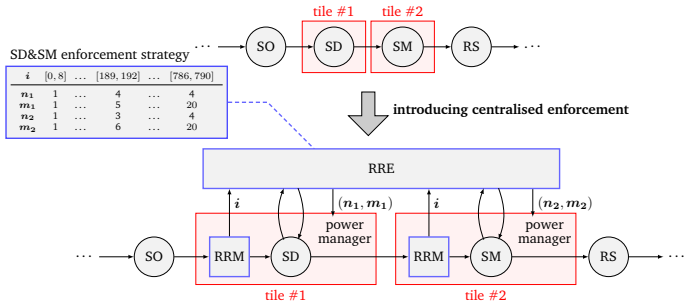


Figure 4.3: Implementation of centralised enforcement using a statically computed energy-optimal enforcement strategy for SD and SM actors with a hard latency upper bound of $UB_L = 95$ ms for both actors collectively.

in the object detection chain, resulting in a latency budget of 95 ms for the two enforced actors.

In case of a centralised enforcement scheme, a single instance of an RRE actor enforces the overall latency upper bound of $UB_L = 95$ ms for both SD and SM actors collectively. The implementation of a centralised enforcement scheme is depicted in Figure 4.3. The statically computed enforcement strategy for the two enforced actors is provided as a look-up table to the RRE actor. At run time, once an image is ready to be processed, the number i of corners in it becomes known. Prior to processing that image, the RRE actor retrieves the (n, m) settings corresponding to i corners from the enforcement strategy table for each enforced actor and instructs the power manager to use these settings on each respective tile. For guaranteeing the latency upper bounds, we assume that the (n, m) setting of each tile is held constant while an image is being processed by the actor running on that tile.

For a distributed enforcement scheme, the latency budget of 95 ms is further subdivided to obtain a latency budget per enforced actor. Taking into account the recorded profiles in Table 4.1, we dedicate a latency budget of 80 ms to the SD actor and 15 ms to SM. Figure 4.4 shows the implementation of a distributed enforcement scheme for the two enforced actors. The statically computed enforcement strategy for each actor is provided as a look-up table to the local RRE actor in charge of enforcing the assigned latency bound for the respective enforced actor.

As a merit of profit, the potential energy savings of each enforcement scheme has been investigated in addition to the verification of its capability in enforcing the latency requirement. For a variety of requirement strictness levels, Table 4.2 presents the average dynamic

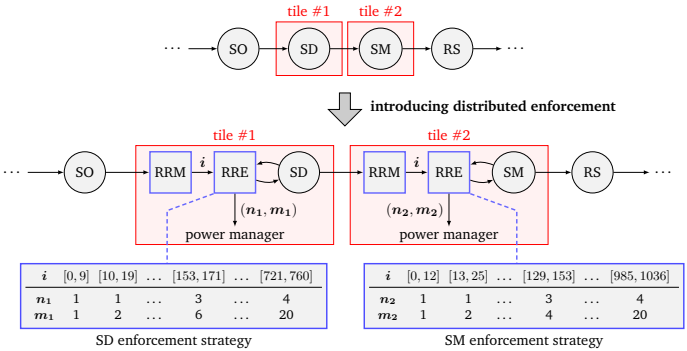


Figure 4.4: Implementation of distributed enforcement using statically computed energy-optimal enforcement strategies for SD and SM actors with hard latency upper bounds of $UB_L = 80$ ms and $UB_L = 15$ ms, respectively.

energy savings of the SD and SM actors compared to the non-enforced scenario (where $n = 4$ and $m = 20$ per actor) for the two schemes of distributed and centralised enforcement. Compared to distributed enforcement, the centralised scheme is able to save slightly more dynamic energy. Moreover, according to the enforcement strategies in Figure 4.3 and Figure 4.4, a centralised RRE can enforce the given requirement for input images with up to $i = 790$ corners, while the enforceable workload in case of distributed enforcement is restricted to $i = 760$. Whereas in the above case study, only a single latency requirement has been investigated, [Tei+20b] describes principles for the simultaneous enforcement of multiple requirements such as latency, power, and reliability. Rather than simple lookup tables, it is shown that this typically involves the generation of not only input-, but state-dependent enforcement strategies.

Table 4.2: Average dynamic energy savings for the SD and SM actors using distributed and centralised enforcement schemes in dependence of requirement strictness. The energy consumption without enforcement ($n = 4$, $m = 20$) serves as a baseline.

requirement strictness	distributed enforcement				centralised enforcement			
	SD	SM	SD+SM	overall	SD	SM	SD+SM	overall
50%	41.2%	38.9%	40.8%	36.8%	41.8%	43.6%	42.1%	37.9%
84%	40.4%	38.7%	40.1%	36.1%	41.2%	42.9%	41.5%	37.3%
97.7%	39.5%	38.3%	39.3%	35.4%	39.4%	40.9%	39.7%	35.7%
100%	37.6%	37.2%	37.6%	33.8%	38.5%	40.2%	38.8%	35.0%

A1

Isolation-Aware Application Characterisation Enforcement of hard timing requirements necessitates a timing analysis to derive safe bounds on the worst-case timing behaviour of applications, taking into account the temporal/spatial isolation scheme among concurrent applications on different resources. Together with Project A4, we have developed an isolation-aware DSE and timing analysis for heterogeneous tiled many-core architectures. Our analysis enables deriving safe yet tight upper bounds on the worst-case timing behaviour of invasive programs by excluding pessimistic interference scenarios which cannot occur under the given isolation scheme among applications. The proposed approach and analysis is presented in [Pou+19b].

Thermal Composability Modifications in the DVFS setting of cores changes their power consumption. This, in turn, affects the chip temperature and may create thermal hot spots which are then counteracted by the chip power manager, e. g. using power gating. Obviously, RRE techniques must be able to either cope and react to such events or, alternatively, try to avoid any thermal violation by construction, e. g. by restricting the range of allowable DVFS settings. In collaboration with Project A4 and Project B3, we have developed a thermally composable Hybrid Application Mapping (HAM) approach which enables providing the RRE actors with a range of DVFS settings that are guaranteed not to induce thermal violations. The proposed approach and its experimental evaluation are detailed in [Pou+19a]. Based on this approach, the enforcement strategy can be developed for a restricted range of control knob settings to guarantee thermal safety and exclude power manager interferences without jeopardising the thermal integrity of the chip.

Publications

[Pou+19a] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel. “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems”. In: *40th IEEE Real-Time Systems Symposium (RTSS)*. 2019.

[Pou+19b] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich. “Isolation-Aware Timing Analysis and Design Space Exploration for Predictable and Composable Many-Core Systems”. In: *31th Euromicro Conference on Real-Time Systems (ECRTS)*. Stuttgart, Germany, 2019, 12:1–12:24. DOI: 10.4230/LIPIcs.ECRTS.2019.12.

[Pou+20] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich. “Real-Time Task Migration for Dynamic Resource Management in Many-Core Systems”. In: *Workshop on Next Generation Real-Time Embedded Systems (NG-RES)*. 2020.

[RHT19] S. Roloff, F. Hannig, and J. Teich. *Modeling and Simulation of Invasive Applications and Architectures*. Springer, May 2019. 168 pp. DOI: 10.1007/978-981-13-8387-8.

[Sch+19b] T. Schwarzer et al. “Compilation of Dataflow Applications for Multi-Cores Using Adaptive Multi-Objective Optimization”. In: *ACM Transactions on Design Automation of Electronic Systems* 24.3 (Mar. 2019), 29:1–29:23. DOI: 10.1145/3310249.

[Sim+20] B. Simon, J. Falk, N. Megow, and J. Teich. “Energy Minimization in DAG Scheduling on MPSoCs at Run-Time: Theory and Practice”. In: *Workshop on Next Generation Real-Time Embedded Systems*. 2020.

[Spi+19] J. Spieck, S. Wildermann, T. Schwarzer, J. Teich, and M. Glaß. “Data-Driven Scenario-based Application Mapping for Heterogeneous Many-Core Systems”. In: *Multicore/Many-core Systems-on-Chip (MCSoC)* (Singapore). Oct. 1–4, 2019.

[SWT19] J. Spieck, S. Wildermann, and J. Teich. “Run-Time Scenario-Based MPSoC Mapping Reconfiguration Using Machine Learning Models”. In: *1st ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. 2019.

[Tei+20a] J. Teich, P. Mahmoody, B. Pourmohseni, S. Roloff, W. Schröder-Preikschat, and S. Wildermann. “Run-Time Enforcement of Non-functional Program Properties on MPSoCs”. In: *A Journey of Embedded and Cyber-Physical Systems*. Ed. by J.-J. Chen. Springer, 2020.

[Tei+20b] J. Teich, B. Pourmohseni, O. Keszocze, J. Spieck, and S. Wildermann. “Run-Time Enforcement of Non-Functional Application Requirements in Heterogeneous Many-Core Systems”. In: *Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2020, pp. 629–636.

A4: Characterisation and Analysis of Invasive Algorithmic Patterns

Michael Bader, Stefan Wildermann

Jan Spieck, Tobias Schwarzer, Behnaz Pourmohseni, Alexander Pöppl, Joachim Falk

Project A4's mission is to explore and establish application characterisation in the invasive computing paradigm and to investigate and evaluate how algorithms and applications may exploit and profit from the resulting predictability features.

A major challenge is that our study of invasive proxy applications in Phase II and the experiences gained reveal that application execution is getting increasingly dynamic:

- Workload scenarios and execution phases of an application vary strongly, such that a static worst- or average-case characterisation for a single scenario becomes inadequate.
- Performance is becoming highly uncertain, as external influences, differences in manufacturing and measures to enforce power limits or energy budgets affect quality numbers in an unforeseeable way.
- Hardware faults are expected to occur more often thus making resources temporally or, due to manufacturing variability and ageing, even permanently unavailable.

The enforcement of non-functional requirements, as investigated in Project A1, concentrates on controlling the claimed resources, and thus always happens within a corridor defined by one respective operating point, see, e. g. [Tei+20a; Tei+20b]. Project A4 expands this approach beyond the boundaries of a single operating point by considering algorithmic adaptations of the application as well as modifications of the resource claim. Our goal is to provide optimised and robust application execution in the context of above aspects by characterising and enforcing adaptation of the application behaviour (e. g. algorithmic variants), the application structure (i. e. the actor model), and the set of allocated resources by means of re-invasion and run-time requirement

enforcement (RRE) to such variances and fault scenarios. In Phase III, we focus on programming and characterising applications with varying execution phases and changing workload. In the following, we report on results achieved in 2019.

SWE-X10 – Multi-Level Actor Graphs

After extending SWE-X10 with a block-adaptive multi-rate local time stepping scheme (LTS), we started to work on making its spacial resolution adaptive as well. Instead of a uniform resolution for the entire grid as before, we want to enable different parts of the simulation domain to be simulated at different resolutions. Our solution concept for this is a multi-level actor graph (see Fig. 4.5). During construction of the actor graph, we create actors for all resolution levels we would like to support. At the beginning of the actor graph execution, each actor exchanges information about its resolution with all its connected neighbours (on the same level, as well as the resolution level above and below). This enables the actors to listen to, receive and interpret updates from its respective active neighbours. Resolution updates may be triggered by an actor on integral timestep multiples. This allows receiving actors to have a clear cut-off where they stop listening to the old actor and start listening to the replacement with the updated resolution. After sending a resolution update notification to its neighbours, the actor starts refining (or coarsening) its bathymetry and cell quantities as necessary, and then sends it, along with all its accumulated boundary information, to the actor on the new grid level. Upon receiving the refinement data, that actor configures its data structures and domain boundaries and sets its finite state machine (FSM) to the appropriate state.

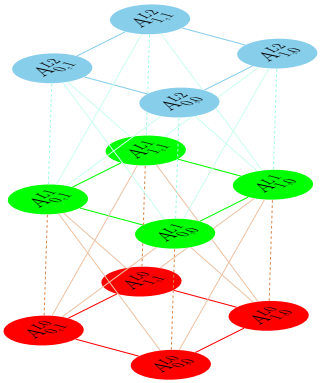


Figure 4.5: Multi-level actor graph with 2×2 actors per level. Each actor on each level is connected to its neighbouring level counterparts, and its neighbours in the two spacial dimensions in the same and neighbouring levels.

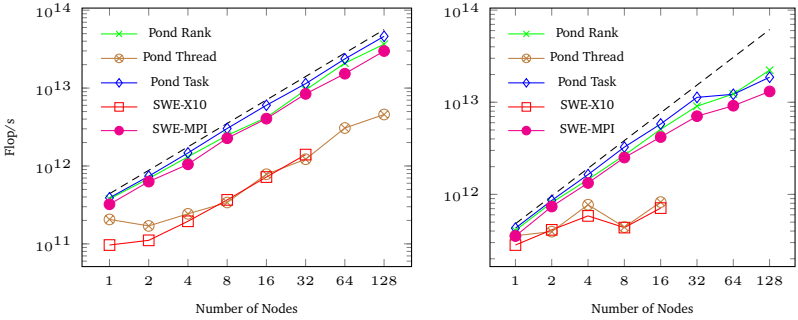
UPC++ actor library In the previous year, we implemented a prototype for an actor library utilising a more widely used HPC language and environment. We chose C++-14 as a language and UPC++ as the communication library. First tests showed a significantly better performance compared to the prior solution in actorX10[5]. In this year, we expanded on the work performed at LBNL to develop and evaluate different execution strategies for the UPC++ actor library [PBB19]. In all cases, they implement the execution semantics as specified for actorX10. This requires functionality that watches for changes in the channels and activates the corresponding actor when such a change happens, until the actor eventually signals its termination. For all execution strategies, this necessitates one or more *event loops* per rank. In the loop, the application needs to make tokens from other ranks available, and then invoke the local actors' FSM.

For the **rank-based execution strategy**, the event loop for each UPC++ rank is realised in the actor graph. Initially, the run time executes any UPC++ communication for inter-actor communication. Thereafter, the FSM of actors with changes to their connected channels is triggered. This strategy is best suited for a small number of actors per UPC++ rank, as all actors within a rank are processed sequentially. Parallelism is achieved instead using multiple UPC++ ranks per physical node.

The **thread-based execution strategy** uses multiple C++ threads per UPC++ rank to parallelise actor execution. Each actor is mapped onto its own operating system thread, and the actor graph is mapped to the main thread of the rank. The individual actors and the actor graph each have their own event loop. The actor graph queries the UPC++ run time for incoming tokens and notifies affected actors accordingly. An actor's event loop contains a call to the UPC++ run time for rank-internal updates, followed by an FSM invocation if the actor has been triggered. This method closely resembles the execution strategy implemented in actorX10.

Finally, the **task-based execution strategy** parallelises the actor execution using OpenMP tasks. As with the rank-based execution strategy, there is only a single event loop per rank, in the actor graph instance. Similarly to the other parallelisation models, the run time is queried for progress to process incoming communication. Then, we iterate through

[5] S. Roloff et al. "ActorX10: An Actor Library for X10". In: *Proceedings of the 6th ACM SIGPLAN X10 Workshop (X10)* (Santa Barbara, CA, USA). ACM, June 14, 2016, pp. 24–29. DOI: 10.1145/2931028.2931033.



(a) Weak Scaling Test **(b)** Strong Scaling Test

Figure 4.6: Scaling tests on the NERSC Cori KNL partition. The setup for the weak scaling test specified $4,096 \times 4,096$ cells per node, and the strong scaling test was performed using an overall size of $16,384 \times 16,384$ for the simulation domain.

all local actors, and for those that have a positive trigger count, we schedule an OpenMP task that queries for progress and invokes the actor's FSM. The tasks are then executed asynchronously on worker threads by the OpenMP run time. The advantage of this approach compared to the thread-based execution strategy is the possibility to match the number of threads employed by a node to its computational resources. Instead of a one-to-one mapping of actors to operating system threads, work is distributed onto a number of threads specified at run time. When an actor is not triggered, it will not create tasks to be scheduled, and it therefore does not use up any CPU resources.

Evaluation We compared Pond, our proxy application for the UPC++ actor library, to SWE-X10, and the BSP-based SWE. In all tests, we used a radial dam break scenario, which is easily scaled to any size.

The weak scaling test was performed with a per-node workload of $4,096 \times 4,096$ grid cells per node. We performed tests starting on a single node up to 128 nodes. The base workload per core was set at 256×256 grid cells per logical thread. Results are shown in Fig. 4.6a. Two out of the three execution models of Pond outperform SWE-X10 significantly. *Pond Thread* performs worst, at a similar level as SWE-X10. *Pond Rank* performs on a level competitive with SWE-MPI, managing to yield a roughly 20% performance benefit over SWE-MPI for the largest run with 128 nodes. *Pond Task* outperforms the other implementations

in this test and is on average 38% faster than SWE-MPI, with over 50% higher performance for the run with 128 nodes.

We also performed a strong scaling test to explore the scalability limits of the actor library (results in Fig. 4.6b). For the test, we set the size of the simulation domain to $16,384 \times 16,384$ grid cells. For a single-node configuration, this led to a patch size of 512×512 grid cells, down to a patch size of 64×64 grid cells with 128 nodes. The performance of SWE-MPI degenerates gradually as the size of each core's working set shrinks. For the actor-based solutions, the more constrained patch size leads to a more sudden drop in performance, as smaller patch sizes – necessary to evenly distribute the grid – lead to more actors and therefore more coordination overhead. As before, Pond manages to retain its performance lead, and again manages to outperform the traditional BSP-based SWE.

MPI actor library Furthermore, we implemented[6] the actor computational model using MPI and OpenMP, the prevalent APIs for HPC applications today. Here, we compare inter-actor communication using two-sided communication primitives to communication using one-sided communication. In a comparison with the UPC++ actor library on the CoolMUC cluster, performance is comparable to the UPC++ actor library. A large benefit of the MPI implementation is the possibility to run actor-based applications on any modern supercomputer.

Scenario-based Application Mapping for Heterogeneous Manycore Systems

The mapping of application tasks to suitable resources on heterogeneous manycore architectures is an important factor in meeting the special requirements needed in the design of embedded systems [Sch+19b]. However, a single mapping of application tasks may be neither efficient nor feasible to enforce a set of requirements in case of an input-dependent workload. In particular, applications that process a stream of input data require a dynamic mapping approach which also involves mapping reconfiguration at run time, which we have investigated for soft-real time [GSW19] and hard-real time applications [Pou+19c].

Since searching for optimal mappings at run time could introduce an intolerable computation overhead and furthermore is in need of, frequently unavailable, run-time information, hybrid application mappings

[6] Master's Thesis of Bruno Miguel: *A Distributed Actor Library for HPC Applications* (Nov 2019)

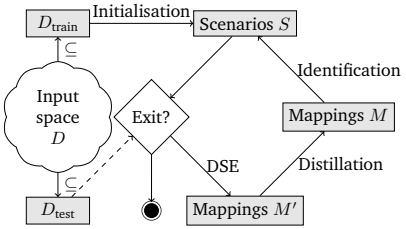


Figure 4.7: Structure of the proposed iterative scenario optimisation loop.

(HAM) [Pou+19b] have prevailed in recent years. HAM techniques compute Pareto-optimal mappings at design time by design space exploration and then select a suitable mapping for the current input data at run time. However, determining optimised mappings for each input data is typically not feasible due to the size of possible input data. As a remedy, we propose to group data with similar execution characteristics into so-called *workload scenarios* for which specialised mappings are computed. Finding these scenarios and scenario-optimised mappings forms an optimisation problem that is elucidated in the next paragraph.

Design-Time Data-Driven Scenario and Mapping Optimisation

[Spi+19] proposes a data-driven optimisation technique for detecting scenarios and finding optimised mappings, which is solely based on measuring non-functional execution properties (like latency) for a given set of input data. Here, scenario selection and mapping optimisation form two interdependent problems that are solved by an iterative design flow displayed in Fig. 4.7.

In the first step, a representative set of training data $D_{\text{train}} \in D$ and test data $D_{\text{test}} \in D$ is selected from the set D of all possible input data. Then, an initial set of scenarios is chosen, e. g. a random distribution of the training data $d \in D_{\text{train}}$ into an arbitrary number of scenarios. This is necessary to initiate the optimisation loop. Based on this scenario distribution, Pareto-optimal mappings for each scenario are determined by a multi-objective optimisation DSE run using evolutionary algorithms. The resulting set of mappings M' is distilled to a reduced set M by clustering similar mappings and selecting sample representatives. This reduced number of mappings improves and accelerates the process of the subsequent scenario identification. Scenarios are identified by clustering the vectors $v(d) = \begin{pmatrix} p(d, m_1) & \dots & p(d, m_\ell) \end{pmatrix}^T, m_1, ..., m_\ell \in M$ of each data $d \in D_{\text{train}}$ with $v(d)$ consisting of the performance numbers

$p(d, m)$ of data d under each mapping $m \in M$. Here, the data tuples corresponding to the vectors in one cluster form a scenario. For these revised scenarios, tailored mappings can be found by a further optimisation cycle that also refines the scenario distribution. This optimisation loop is stopped once a termination criterion applies, e. g. the quality of the scenarios converges. The evaluation of the proposed scenario-based mapping optimisation shows that we can easily obtain average performance improvements of up to 10% compared to state-of-the-art mapping optimisation under a highly variable workload for a ray-tracing and stitching application.

Scenario-Based Run-Time Application Mapping Reconfiguration

We propose a run-time manager (RTM) that uses the optimised scenarios and scenario-optimised mappings from design time to select mappings for a stream of unknown input data at run time in [SWT19]. The integration of the RTM into the application flow is displayed in Fig. 4.8. First, we have to identify the scenario of incoming data so that we can select suitable mappings. For black-box applications, however, the scenario of input data cannot generally be determined before the data is executed. Instead, we conclude the scenarios based on the non-functional execution properties $e(d, m)$ of the current data $d \in D$ under the active mapping m. Thus, scenario identification forms a classification problem that is solved by neural networks, which are trained with execution data $e(d, m)$ labelled with the best-suited scenario. Based on the identified scenarios of the current and preceding data, we select a mapping for subsequent input data. Therefore, we need knowledge about the expected scenario sequence of the input stream at run time. For this reason, we use training sequences similar to the run-time stream to teach an RTM model strategy for mapping selection. Experiments show that genetic-programming-based models offer a low training time, low run-time overhead, and highly accurate solutions. Here, the reconfiguration overhead when switching between two mappings and the error susceptibility of the scenario identification model are taken into consideration.

Evaluations for a ray-tracing and stitching application show that the proposed run-time manager selects scenario-based mappings that provide a significant speedup of up to 13% compared to a fixed, single mapping. Furthermore, the RTM achieves high scenario identification rates of over 90% and low run-time overhead smaller than 0.001%.

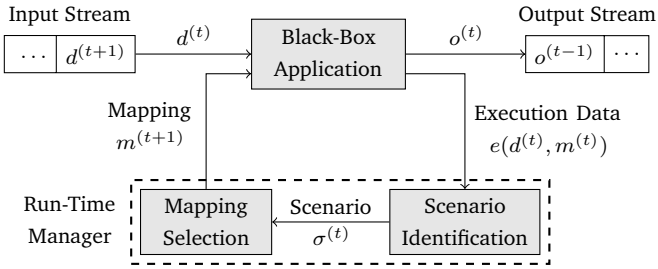


Figure 4.8: Integration of a run-time manager into the application flow. Based on the execution data e of the current input $d^{(t)}$ under the current mapping $m^{(t)}$, a mapping for the next input $d^{(t+1)}$ is selected.

Thermal Composability

The ongoing process technology downsizing has given rise to an increased on-chip temperature, so that the thermal integrity of the platform must be monitored and enforced at run time by means of Dynamic Thermal Management (DTM) techniques which use countermeasures, e. g. DVFS and power gating. This, however, can lead to situations where the execution of one application influences the execution of other applications: Due to heat transfer among neighbouring cores, the mapping used for launching an application may affect the temperature profile of the neighbouring cores that are used by other applications, leading them to a thermal violation and, thus, exposing those applications to DTM countermeasures, even though they did not induce the thermal violation in the first place.

In cooperation with Projects B3 and A1, we developed for the first time a thermally composable HAM methodology that enforces thermal safety proactively at the launch time of applications and, thereby, prevents DTM interferences that react to thermal violations in [Pou+19a].

Publications

[GSW19] D. Gabriel, W. Stechele, and S. Wildermann. “Resource-Aware Parameter Tuning for Real-Time Applications”. In: *Architecture of Computing Systems – ARCS 2019*. Ed. by M. Schoeberl, C. Hochberger, S. Uhrig, J. Brehm, and T. Pionteck. Springer International Publishing, 2019, pp. 45–55. DOI: 10.1007/978-3-030-18656-2_4.

[PBB19] A. Pöppl, S. Baden, and M. Bader. “A UPC++ Actor Library and Its Evaluation on a Shallow Water Proxy Application”. In:

Parallel Applications Workshop, Alternatives To MPI+X. IEEE. Denver, Colorado, United States of America: IEEE, Nov. 2019.

[Pou+19a] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel. “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems”. In: *40th IEEE Real-Time Systems Symposium (RTSS)*. 2019.

[Pou+19b] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich. “Isolation-Aware Timing Analysis and Design Space Exploration for Predictable and Composable Many-Core Systems”. In: *31th Euromicro Conference on Real-Time Systems (ECRTS)*. Stuttgart, Germany, 2019, 12:1–12:24. DOI: 10.4230/LIPIcs.ECRTS.2019.12.

[Pou+19c] B. Pourmohseni, S. Wildermann, M. Glaß, and J. Teich. “Hard Real-Time Application Mapping Reconfiguration for NoC-Based Many-Core Systems”. In: *Real-Time Systems* (2019), pp. 1–37. DOI: 10.1007/s11241-019-09326-y.

[Sch+19b] T. Schwarzer et al. “Compilation of Dataflow Applications for Multi-Cores Using Adaptive Multi-Objective Optimization”. In: *ACM Transactions on Design Automation of Electronic Systems* 24.3 (Mar. 2019), 29:1–29:23. DOI: 10.1145/3310249.

[Spi+19] J. Spieck, S. Wildermann, T. Schwarzer, J. Teich, and M. Glaß. “Data-Driven Scenario-based Application Mapping for Heterogeneous Many-Core Systems”. In: *Multicore/Many-core Systems-on-Chip (MCSoC)* (Singapore). Oct. 1–4, 2019.

[SWT19] J. Spieck, S. Wildermann, and J. Teich. “Run-Time Scenario-Based MPSoC Mapping Reconfiguration Using Machine Learning Models”. In: *1st ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. 2019.

[Tei+20a] J. Teich, P. Mahmoody, B. Pourmohseni, S. Roloff, W. Schröder-Preikschat, and S. Wildermann. “Run-Time Enforcement of Non-functional Program Properties on MPSoCs”. In: *A Journey of Embedded and Cyber-Physical Systems*. Ed. by J.-J. Chen. Springer, 2020.

[Tei+20b] J. Teich, B. Pourmohseni, O. Keszocze, J. Spieck, and S. Wildermann. “Run-Time Enforcement of Non-Functional Application Requirements in Heterogeneous Many-Core Systems”. In: *Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2020, pp. 629–636.

A5: Scheduling Invasive Multicore Programs Under Uncertainty

Nicole Megow

Bertrand Simon

The goal of Project A5 is to develop algorithms that can handle uncertain input data. We design and mathematically analyse algorithms for scheduling and resource management using the *invasive computing paradigm*. Our methods shall give performance guarantees concerning the predictability and also quantify how hardware and software requirements affect the performance and predictability. This may reveal also optimisation potential in the systems architecture or algorithms.

This project is of foundational character and aims for theoretical guarantees by building on methods from algorithms theory and mathematical optimisation. Our main focus lies on *provable worst-case guarantees* and *coping with uncertainty*. When predictability is crucial, e. g. in safety-critical applications, most multicore systems still rely on single-core usage. The reason is that the current state-of-the-art approaches in real-time scheduling do not capture the difficulties in scheduling parallel workloads in a predictable way. In this project, we develop algorithms with guarantees on predictability and resource utilisation as is required to exploit the full power of parallel computing and particularly the benefits of invasive computing also for safety-critical applications. We expect to develop algorithms that are not only efficient from a theoretical standpoint but can be efficiently implemented in practice.

Speed-scaling for Power Management

Speed-scaling (or frequency/voltage scaling) is the main technique for power management, in both academic research and practice. It involves *dynamically* changing the speed of a processor which is allowed by current microprocessors. Major research questions ask for dynamic algorithms that determine a schedule for a given set of tasks and also decide at which speed $s \geq 0$ the processor(s) shall run at any time. Running a processor at a certain speed requires a certain amount of

power. Power is typically modelled as a monomial (convex) function of speed, $P(s) = s^\alpha$ with a small constant $\alpha > 1$. Given a fixed deadline, we compute the optimal power distribution and schedule that minimises energy consumption.

A5

We focused in collaboration with Project A1 on the problem of scheduling a set of tasks with precedence constraints on multiple processing units, i. e. on a multicore chip. Several variants of this problem are relevant and studied in the literature. First, regarding the allowed set of speeds, it can be either continuous in a given interval, or be part of a prescribed set specific to the chip. Second, the tasks may already be mapped to cores, in which case the problem is solely to compute the appropriate speeds; otherwise, the algorithm should also compute the schedule of tasks to cores.

We surveyed the known results in related domains and reformulate them in this setting as well as providing new algorithms which all have strong performance guarantees compared to an optimal solution. These algorithms are built on various techniques such as Integer Linear Programming, fractional relaxations, Convex Programming, or combinatorial algorithms. One of the most important conclusions of this work concerns the problem in which tasks are already mapped for a special class of graphs named series-parallel, which includes many actual applications. This problem is solvable in polynomial time for continuous speeds and all graphs, but admits a fast linear-time algorithm, which can handle thousands of tasks in a few milliseconds on a standard computer. This continuous solution can also be rounded to a prescribed set of speeds while staying within a few percents of the optimal solution in our experiments. These results have been accepted for publication in [Sim+20].

Scheduling Self-Suspending Tasks

In computing systems, a job may suspend itself (before it finishes its execution) when it has to wait for certain results from other (usually external) activities. For real-time systems, such self-suspension behaviour has been shown to induce performance degradation. Hence, the researchers in the real-time systems community have devoted themselves to the design and analysis of scheduling algorithms that can alleviate the performance penalty due to self-suspension behaviour. As self-suspension and delegation of parts of a job to non-bottleneck resources is pretty natural in many applications, researchers in the operations research (OR) community have also explored scheduling algorithms for

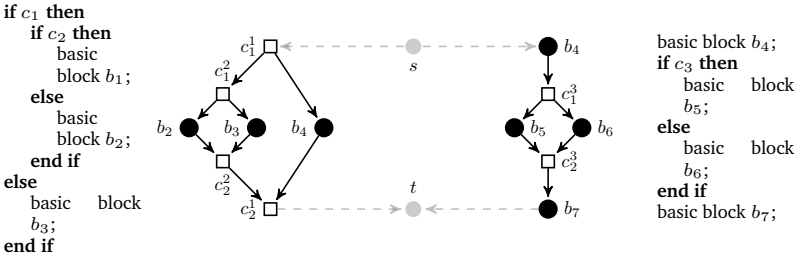


Figure 4.9: Two example source code excerpts and their representation as conditional DAGs. Each b_i represents a sequence of statements and each c_i represents some Boolean expression. The dummy source s and sink t show how the parallel execution of two source code excerpts can be represented with a single task.

A5

systems with such suspension behaviour, called the *master-slave* problem in the OR community.

In our paper [Che+19], we first review the results for the master-slave problem in the OR literature and explain their impact on several long-standing problems for scheduling self-suspending real-time tasks. Then, we provide a systematic study of different approximation metrics with respect to resource augmentation factors (speedup) of several heuristic algorithms that can be applied for different self-suspension models. These models represent different degrees of uncertainty about the suspension pattern and suspension duration for tasks. We obtain small constant factors for both uniprocessor and multiprocessor systems, and demonstrate that different approximation metrics can create different levels of difficulty for the approximation. Our experimental results show that such more carefully designed schedules can significantly outperform the state-of-the-art.

Conditional DAG Scheduling

As parallel processing became ubiquitous in modern computing systems, parallel task models have been proposed to describe the structure of parallel applications. The workflow scheduling problem has been studied extensively over past years, focusing on multiprocessor systems and distributed environments (e. g. grids, clusters). In workflow scheduling, applications are modelled as directed acyclic graphs (DAGs). DAGs have also been introduced in the real-time scheduling community to model the execution of multi-threaded programs on a multicore architecture. The DAG model usually assumes a fixed DAG structure capturing only straight-line code. We studied the more general conditional

DAG model which allows uncertainty in the DAG structure by allowing the presence of conditional control-flow (if-then-else) constructs; see Fig. 4.9. In particular, we considered the problem of computing the worst-case makespan for conditional DAGs, that is, the maximum amount of time a multicore system might need to execute the conditional DAG. Due to the presence of control-flow instructions, it is uncertain which parts of the conditional DAG, respectively, the modelled application, are actually going to be executed and, therefore, computing the worst-case makespan is non-trivial even for fixed-priority scheduling algorithms. This problem is crucial when scheduling time-critical applications on multicore systems.

We perform a thorough analysis on the worst-case makespan of a conditional DAG task under list scheduling (a.k.a. fixed-priority scheduling). The problem is solvable in polynomial time for well-nested conditional DAGs on single-core systems. For general conditional DAG tasks, the problem is intractable even on a single processor. We show several hardness results concerning the complexity of the optimisation problem on multiple processors for conditional DAGs with a well-nested structure. Complementing these negative results, we designed an exact pseudo-polynomial time algorithm and a polynomial time approximation with a strong performance guarantee, both for certain practice-relevant conditional DAG structures on multicore systems. These results have been accepted for publication in [Mar+20].

Further Research

Thermal-aware Mapping In collaboration with Project B3, we studied a problem named thermal-aware mapping. Executing a job on a chip component increases its temperature and the one of nearby components. This effect depends on multiple factors such as the power consumed by a core, the local thermal conductivity or the ambient temperature. If the temperature of a component exceeds a certain threshold, dynamic thermal management techniques are triggered in order to avoid overheating. Triggering these techniques results in performance losses and must then be avoided. We considered the theoretical aspects of the problem consisting of mapping tasks to components and deciding a maximum power that each component is allowed to consume while ensuring that no component exceeds the threshold temperature. Project B3 already noticed that, on actual data, a greedy algorithm computes a solution which is not far from the optimal. Hence, the question was to decide whether it is conceivable to compute efficiently the optimal solution. We have been able to answer negatively this question for instances which

require too many irregularities to be realistic. The complexity of solving this problem for noisy but realistic instances remains open.

Time Cost Trade-off Problem We also explored the relation between the speed-scaling problem previously described and the time-cost trade-off problem with convex cost functions. The framework of this problem is the following: we are given a graph with weights on the edges, and we are allowed to invest some cost for each edge in order to decrease its weight. The goal is to obtain a graph with a bounded critical path while investing the minimum possible cost. A low-complexity algorithm for this problem was recently proposed by Dorit Hochbaum[7]. The previous algorithms were more expensive and often relied on solving large convex programs. This new approach used only minimum cut algorithms and obtain a much better complexity. We however noticed an issue in this approach, and, after exchanging with the author, contributed to the correction of this algorithm. The connection between this problem and the speed-scaling setting was not noticed so far to the best of our knowledge. We believe that we can gain new insights for each problem using their connection and the techniques currently used to solve the other problem.

Learning-augmented Algorithms One of the most common ways to deal with uncertainty in the theoretical scheduling literature is to consider online problems: no information on the future instance is known, and therefore any pathological scenario has to be handled by the algorithms. This model is by essence pessimistic and algorithms usually perform better in practice than what can be guaranteed in theory. This downside has been reduced in refined models, which for instance consider the knowledge of probability distributions on the future inputs, but this might not be realistic. A new line of research assumes the knowledge of predictions, which typically come from machine learning. The novelty of this approach is that no assumption is made on the quality of the prediction. Indeed, if an instance is not well-covered by the training set, the prediction can be arbitrarily bad. Hence, a guaranteed algorithm cannot fully trust such predictions. The quality of an algorithm is then described by two quantities. The robustness describes its worst-case performance, which should not be much worse than the one of an online algorithm. The consistency describes its performance

[7] D. S. Hochbaum. "A polynomial time repeated cuts algorithm for the time cost tradeoff problem: The linear and convex crashing cost deadline problem". In: *Computers & Industrial Engineering* 95 (2016), pp. 64–71.

when the prediction is accurate, and so depends on the error made by the prediction. The objective is then to obtain an algorithm which makes use of good predictions to improve its performance, but still has a worst-case guarantee in case these predictions were irrelevant. We currently explore such algorithms in the context of invasive computing.

Publications

[Che+19] J.-J. Chen, T. Hahn, R. Hoeksma, N. Megow, and G. von der Brüggen. "Scheduling Self-Suspending Tasks: New and Old Results". In: *31st Euromicro Conference on Real-Time Systems (ECRTS)*. 2019.

[Mar+20] A. Marchetti-Spaccamela, N. Megow, J. Schlöter, M. Skutella, and L. Stougie. "On the Complexity of Conditional DAG Scheduling in Multiprocessor Systems". In: *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2020.

[Sim+20] B. Simon, J. Falk, N. Megow, and J. Teich. "Energy Minimization in DAG Scheduling on MPSoCs at Run-Time: Theory and Practice". In: *Workshop on Next Generation Real-Time Embedded Systems*. 2020.

B1: Adaptive Application-Specific Invasive Micro-Architectures

Lars Bauer, Jürgen Becker, Jörg Henkel

Marvin Damschen, Tanja Harbaum, Fabian Lesniak

Project B1 investigates mechanisms that provide run-time adaptivity: in the micro-architecture (μArch) and by using a run-time–reconfigurable fabric. In the first two funding phases, the concepts of state-of-the-art reconfigurable processors have been advanced towards invasion, and their benefits have been exploited in the invasive computing project. In the current phase, the focus of Project B1 shifts towards run-time requirement enforcement and different multi-objective optimisations. Run-time requirement enforcement includes analysing and enforcing WCETs and security requirements, with respect to running real-time and best-effort applications on the *i*-Core at the same time. Furthermore, Project B1 is working on improving the performance and flexibility of the μArch by optimising the intra-tile memory hierarchy and introducing approximate accelerators, which allows to adjust the desired calculation accuracy during run time to fulfil different run-time requirements.

Information Leakage Protection

In Phase II, Project B1 introduced a Dynamic Intra-tile Cache Architecture (DICAR). It provides reallocatable cache tiles which can be remapped on demand, according to the individual cache requirements of the applications on each processor. Originally, the cache architecture was designed with functional goals in mind, while not considering security concepts in the first place. Therefore, we have investigated possible information leakage between individual cache tiles. Information leakage can happen on both the hardware and software level. Since the DICAR is purely implemented in hardware, we focus on methods to detect and prohibit information leakage on the hardware level.

The DICAR offers isolation between cache tiles of individual processors, thus guaranteeing that no information stored to the cache will be visible on another processor without sharing it explicitly or through the

main memory. However, vulnerability to information leakage can not be ruled out at design time with a one-hundred per cent guarantee. Possible weaknesses include untested or unexpected hardware behaviour, which may result from incorrect implementation or even degradation of the integrated circuit. To detect leakage, we propose measuring the bitrate of data written or read by each CPU to each of its cache tiles. The observed bitrate at each cache tile is being compared to a previously determined reference.

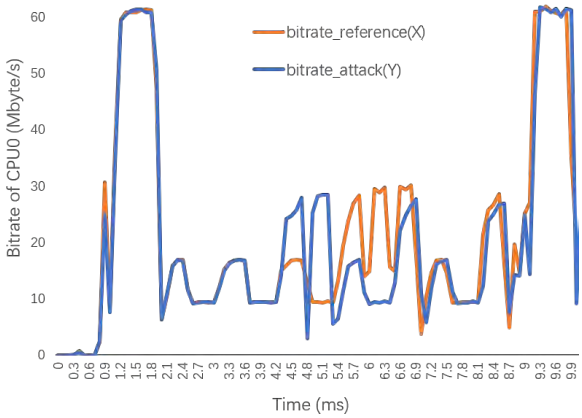


Figure 4.10: Cache bitrate reference compared to the leakage case.

Figure 4.10 shows a comparison between the reference bitrate and the deviating measurement in case of information leakage. In this case, information is leaked by a secondary processor which has unexpected read access to cached values of the primary CPU. However, this mechanism is only feasible for data flow centric, repetitive tasks due to the dependency on a reference bitrate measurement. Tasks with non-uniform memory access patterns can deviate from the reference even without leakage.

To further improve the applicability of the information leakage protection, Project B1 will work on improving the correlation algorithms and investigate on how to detect leakage for less deterministic applications.

WCET Enforcement for Opportunistic Run-Time Reconfiguration

In Phase II, we have developed methods and tools that allow us to determine a safe and tight upper bound for the worst-case execution time (WCET) of the *i*-Core, even when using Special Instructions (SIs) and

reconfiguring them during run time. In the last year, we presented an offline method that decides which SIs shall be implemented in hardware and which shall be emulated in software in order to minimise the guaranteeable WCET. However, this so-called *WCET configuration* (WCET-Cfg) has a noticeably reduced average-case execution time (ACET) compared to a configuration that optimises for the average case (ACET-Cfg, which –on the other side– has a significantly higher WCET). And in this year, we succeeded to combine the best of both, i. e. we provide the improved performance of the ACET-Cfg, while still being able to guarantee the minimised WCET of the WCET-Cfg.

The main idea is to utilise the *slack*, i. e. whenever an application executes, then it will often execute faster than the guaranteed WCET. The reason is that the guaranteed WCET is normally an upper bound of the generally unknown actual WCET. We decompose the guaranteed WCET of a job (e. g. encoding a video frame) into the WCET for one iteration of its outer-most loop, e. g. encoding a single Macro-Block (MB) of the frame. After each iteration of the kernel (i. e. the outer-most loop), we determine how much slack we accumulated so far by using a performance counter. We start executing with the WCET-Cfg, but after some time we have accumulated enough slack to reconfigure to the ACET-Cfg. During reconfiguration, the accumulated slack will reduce drastically, but afterwards the ACET-Cfg will most likely increase the slack faster than the WCET-Cfg did. However, it can happen, that the ACET-Cfg executes slower than the WCET-Cfg, as it had a larger WCET. Eventually, it could happen that we violate the WCET that we initially guaranteed for encoding the video frame. To ensure that this never happens, we *enforce* the WCET, by reconfiguring back to the WCET-Cfg early enough to guarantee that the WCET can never be violated. Therefore, we need to determine the amount of slack that we need to accumulate before *safely* switching to the ACET-Cfg.

First, it takes at most $\mathrm{WCET}_{\mathrm{reconf}}^{\text{ACET-Cfg}}$ cycles to configure the ACET-Cfg. Then, a kernel iteration (encoding one MB) lasts at most $\mathrm{WCET}_{\mathrm{iter}}^{\text{ACET-Cfg}}$ cycles, which means that the accumulated slack can reduce by at most $(\mathrm{WCET}_{\mathrm{iter}}^{\text{ACET-Cfg}} - \mathrm{WCET}_{\mathrm{iter}}^{\text{WCET-Cfg}})$ cycles per iteration. Finally, switching back to the WCET-Cfg takes at most $\mathrm{WCET}_{\mathrm{reconf}}^{\text{WCET-Cfg}}$ cycles. In summary, the minimum accumulated slack to be able to switch to the ACET-Cfg and remain within the WCET guarantee even in the worst case is:

$$\mathrm{WCET}_{\mathrm{reconf}}^{\text{ACET-Cfg}} + \left(\mathrm{WCET}_{\mathrm{iter}}^{\text{ACET-Cfg}} - \mathrm{WCET}_{\mathrm{iter}}^{\text{WCET-Cfg}}\right) + \mathrm{WCET}_{\mathrm{reconf}}^{\text{WCET-Cfg}} \quad (4.1)$$

To safely switch back from the ACET- to the WCET-Cfg, the reconfiguration needs to be triggered when the slack is less than:

$$\left(\text{WCET}_{\text{iter}}^{\text{ACET-Cfg}} - \text{WCET}_{\text{iter}}^{\text{WCET-Cfg}}\right) + \text{WCET}_{\text{reconf}}^{\text{WCET-Cfg}} \tag{4.2}$$

For any value larger than that, there is enough accumulated slack available to safely execute one more iteration of the kernel (even in the worst case) and switch back to WCET-Cfg afterwards. Practically, it is unlikely that we will have to switch back (and we never observed that case in our experiments). But it is important to note, that the system is prepared to do so if needed and that it will do so timely enough to enforce the initially guaranteed WCET to encode one frame. Normally, the ACET-Cfg executes faster and the accumulated slack is then available to execute other *i*-lets or to use power-saving modes etc.

To evaluate our approach, we benchmarked the main kernel of an H.264 video encoder (the `EncodeMacroBlock` kernel). It executes SIs to encode MBs with inter-frame prediction (I-MBs) or intra-frame prediction (P-MBs). The SIs used to accelerate I-MBs and P-MBs differ partially and thus the WCET- and ACET-Cfg differ as well. The ACET-Cfg focuses more on those SIs for I-MBs, as they dominate the performance for most videos (based on profiling), whereas the WCET-Cfg must not use any profiling information but has to consider the worst case.

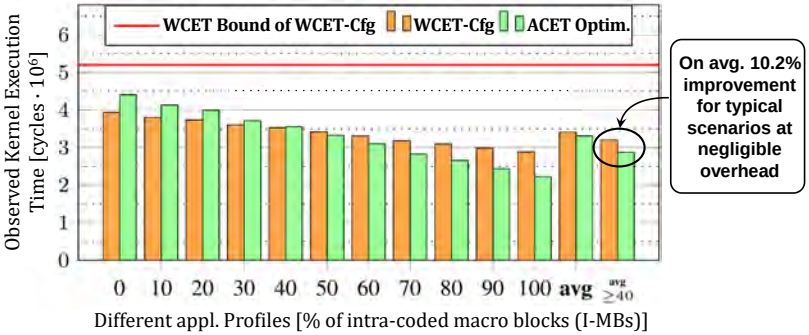


Figure 4.11: Optimised execution time of `EncodeMacroBlock` for different execution profiles.

Figure 4.11 shows execution time results of the `EncodeMacroBlock` kernel when our approach is applied for different I-MB/P-MB ratios. A kernel iteration that encodes a P-MB takes up to 3744 cycles longer when using the ACET-Cfg compared to the WCET-Cfg, as the ACET-Cfg is optimised for I-MBs ($\text{WCET}_{\text{iter}}^{\text{ACET-Cfg}} = 16374$ vs. $\text{WCET}_{\text{iter}}^{\text{WCET-Cfg}} = 12630$). The reconfiguration bandwidth was 800 MB/s (as supported by Xilinx

UltraScale+ FPGAs) and switching between WCET- and ACET-Cfg takes 45658 cycles. Our approach is beneficial for frames that contain at least 50% I-MBs (i. e. at least moderate motion in the video). The maximum execution time reduction is 23.0%. For frames that contain less than 50% I-MBs, the execution time can be slowed down (as expected) by up to 11%. The reason is that such scenarios were rare and thus the ACET-Cfg did not optimise for them. But even if they happened, the statically guaranteed WCET bounds were never violated and actually it was never required to reconfiguring back to the WCET-Cfg. When considering the typical execution profiles with at least 40% I-MBs, then the average execution time reduction is 10.2% compared to continuously executing the WCET-Cfg. Note that these execution time reductions were achieved on top of an already highly optimised system. Just using the *i*-Core accelerators already reduces the guaranteeable WCET by more than $10\times$ compared to executing software only. On top of that, our WCET-optimised configuration reduces the WCET by more than 29%. And on top of that, our opportunistic run-time reconfiguration reduces the average-case execution time by 10.2% while still ensuring the guaranteed optimised WCET (by enforcement) and by negligible overheads (adding slack monitoring using a single performance counter). More details about this work are available in [DBH19a].

Preemption of the Partial Reconfiguration Process to Enable Real-Time Computing with FPGAs

In a collaboration with the University of Pisa, we developed a high-performance reconfiguration controller [Ros+18] for the *i*-Core. It allows to *preempt* and *resume* a reconfiguration, which ensures that reconfiguration requests from a high-priority task can no longer be delayed be reconfigurations from low-priority tasks, while at the same time ensuring progress for the low-priority reconfigurations (instead of aborting and restarting them).

Near Memory *i*-Core Operation

As part of the Intra-Tile Memory Hierarchy Reorganisation, we are trying to improve the accessibility of the *i*-Core Tile-Local Memory (TLM). Previously, Project B1 enabled all generic processors in a tile to be able to use the *i*-Core SIs remotely. To go one step further, we are working on making the *i*-Core available as a generic near-memory accelerator for its TLM. This allows processors to write data to the *i*-Core

TLM in a fire-and-forget fashion, which is then picked up by an *i*-Core accelerator for further processing.

Compared to Remote-SIs, operations on data are not explicitly executed with an instruction, but rather run implicitly by memory access. Thereby, the *i*-Core can run asynchronously with respect to the issuing generic processor, allowing both to run in parallel. Additionally, the *i*-Core has advantages of near-memory computing: Having a dedicated connection to the TLM, the *i*-Core features high memory bandwidth. While not using the main AHB bus, *i*-Core operations are not slowing down the generic cores in the tile.

Work on this topic is still ongoing. Once the overall design and concept is finished, this feature will be implemented and evaluated on the proFPGA prototyping system.

Publications

[Bau+19] L. Bauer et al. “Analyses and Architectures for Mixed-Critical Systems: Industry Trends and Research Perspective”. In: *International Conference on Embedded Software (EMSOFT)*. Invited Special Session Extended Abstract. New York City, NY, USA, Oct. 2019, 13:1–13:2.

[Dam19] M. Damschen. “Worst-Case Execution Time Guarantees for Runtime-Reconfigurable Architectures”. Dissertation. Chair of Embedded Systems (CES), Department of Informatics, Karlsruhe Institute of Technology, Germany, 2019.

[DBH19a] M. Damschen, L. Bauer, and J. Henkel. “WCET Guarantees for Opportunistic Runtime Reconfiguration”. In: *IEEE/ACM 38th International Conference on Computer-Aided Design (ICCAD)*. Westminster, CO, USA, Nov. 2019.

[DBH19b] M. Damschen, L. Bauer, and J. Henkel. “Worst-Case Execution Time Guarantees for Runtime-Reconfigurable Architectures”. Ph.D. Forum at IEEE/ACM 22nd Design, Automation and Test in Europe Conference (DATE). Florence, Italy, Mar. 2019.

[Dam+19] M. Damschen, M. Rapp, L. Bauer, and J. Henkel. “i-Core: A runtime-reconfigurable processor platform for cyber-physical systems”. In: *Embedded, Cyber-Physical, and IoT Systems*. Ed. by S. S. Bhattacharyya, M. Potkonjak, and S. Velipasalar. Springer International Publishing, 2019.

[Har19] T. Harbaum. “Dynamisch adaptive Mikroarchitekturen mit optimierten Speicherstrukturen und variablen Befehlssätzen”. Dissertation. Institut für Technik der Informationsverarbeitung (ITIV), Fakultät für Elektrotechnik und Informationstechnik, Karlsruher Institut für Technologie (KIT), June 25, 2019.

[Ros+18] E. Rossi, M. Damschen, L. Bauer, G. Buttazzo, and J. Henkel. “Preemption of the Partial Reconfiguration Process to Enable Real-Time Computing with FPGAs”. In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 11.2 (Nov. 2018), 10:1–10:24. DOI: 10.1145/3182183.

B2: Invasive Tightly-Coupled Processor Arrays

Jürgen Teich

Andreas Becher, Marcel Brand, Frank Hannig, Dominik Walter

B2

Project B2 investigates invasive computing on Tightly-Coupled Processor Arrays (TCPAs) (see Fig. 4.12). These have been shown to provide a highly energy-efficient and, at the same time, timing-predictable acceleration for many computationally intensive loop applications from diverse areas such as scientific computing, digital signal and image processing. In terms of latency and throughput, TCPAs are timing-predictable in the number of cycles when executing a loop application in parallel.

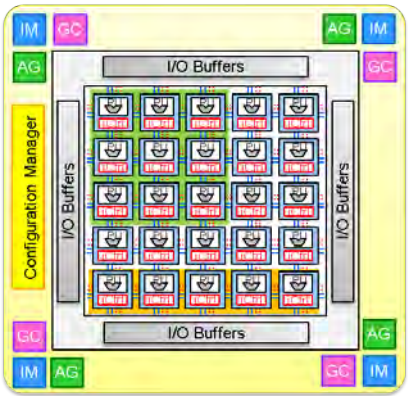


Figure 4.12: TCPA with 5x5 Processing Elements (PE), and four Global Controllers (GC), Invasion Managers (IM), I/O Buffers, and Address Generators (AG).

In the current funding phase, the problem of run-time enforcement of non-functional execution qualities also for parallel loop programs executed on TCPAs is in the focus of this project. In order to enforce a given set of non-functional requirements of a loop nest when executed in parallel on an invasive TCPA, an overprovisioning of resources (the

invaded region of TCPA processors) shall be greatly avoided. To do so, completely novel techniques need to be developed summarised as (a) self-invasion of claim sizes of latency-bound programs, (b) self-power adjustment, and (c) self-selection of redundancy scheme. Further investigations include (d) the exploitation of approximate loop computing on TCPAs in order to stay within execution time bounds or to save energy, (e) invasive floating-point TCPAs with invadable precision that will open a new dimension of applications, and (f) tile-level timing analysis and scheduling of communications (data transfers) between tiles. In the following, we report on investigations performed and results achieved in 2019 with a focus on topics (e) and (f).

FloaTCPAs - Floating-Point Functional Units

Initially, TCPAs only supported single-cycle fixed point instructions[8]. In order to open their applicability to a myriad of scientific computing problems where typically floating-point calculations are the default, we extended the most recent PE architecture based on orthogonal instruction processing[9] by floating-point function units (FPUs). The support of floating-point operations other than just additions/subtractions and multiplications such as divisions and square root instructions, to name a few, does not allow to support a single-cycle instruction execution but required the design of a completely pipelined PE architecture. In this realm, we have investigated area, execution latency, and power trade-offs. Moreover, as often, accuracy may be traded off for any of the other objectives, we investigated whether emerging concepts of approximate computing may also be beneficially applicable in this context to realise speed/energy trade-offs.

Approximate Array Computing

With the goal to save time and energy when executing loop programs in parallel on 100 or more PEs of a TCPA, we are currently exploiting

[8] V. Lari, S. Muddasani, S. Boppu, F. Hannig, and J. Teich. "Design of Low Power On-Chip Processor Arrays". In: *Proceedings of the 23rd IEEE International Conference on Application-specific Systems, Architectures, and Processors (ASAP)* (Delft, The Netherlands). IEEE Computer Society, July 9–11, 2012, pp. 165–168. DOI: 10.1109/ASAP.2012.10.

[9] M. Brand, F. Hannig, A. Tanase, and J. Teich. "Orthogonal Instruction Processing: An Alternative to Lightweight VLIW Processors". In: *Proceedings of the IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)* (Seoul, Republic of Korea). IEEE Computer Society, Sept. 18–20, 2017, pp. 5–12. DOI: 10.1109/MCSoC.2017.17.

concepts of the research area of *approximate computing*. A major driver for not wanting to compute a loop nest accurately is in line with the major goal of our CRC/Transregio indeed to enforce performance and/or energy requirements also on massively parallel processor arrays such as TCPAs for a wide range of loop bounds. In [Bra+19a], we proposed the revolutionary idea and concept of *anytime instructions*. An anytime instruction denotes a floating-point instruction that encodes the number a of mantissa bits of a floating-point operation to be accurately computed during instruction execution in the instruction itself. As the computation of the mantissa is typically on the critical path of any floating-point operation, anytime instructions do allow for a tight control of the execution latency of an instruction with the penalty of errors in the least significant mantissa bits. In 2019, we developed functional units with this type of programmable accuracy, so-called *anytime functional units* (AFU). In the following, we illustrate the concept of anytime instructions for floating-point divisions, see [Bra+19a] for details.

Anytime Division Based on the idea above of anytime instructions, we present the instruction format, architecture, and implementation of a concrete functional unit for anytime division instructions called Anytime Division Functional Unit (ADFU) for normalised floating-point numbers, see [Bra+19a].
An anytime division instruction on three registers is specified as follows:

```
DIV_a  target  dividend  divisor
```

Here, `a` denotes the number of most-significant mantissa bits of the division `dividend/divisor` that shall be accurately computed and stored in the register `target`. Note that the exponent and sign of the quotient shall always be computed accurately.

First, we analysed the error of an anytime division based on the number a of calculated most-significant bits of the mantissa of the quotient. Let a normalised floating-point number $F = se_{E-1} \dots e_0 m_{M-1} \dots m_0$ consist of 1 sign bit s, E exponent bits and M mantissa bits. The result of an anytime division $\mathrm{div}(F_1, F_2, a)$ of a floating-point number F_1 by a floating-point number F_2, when computing a, $1 \leq a \leq M+1$, most-significant mantissa bits accurately, is obtained by:

$$\mathrm{div}(F_1, F_2, a) = -1^{s_1 - s_2} \cdot 2^{\Sigma_{k=0}^{E-1}\left(2^k \cdot (e_{1,k} - e_{2,k})\right)} \cdot \Sigma_{k=0}^{a-1}\left(2^{-k} \cdot q_{M'-k}\right),$$

where

$$q_{M+1} \dots q_0 = \frac{1 m_{1,M} \dots m_{1,0}}{1 m_{2,M} \dots m_{2,0}}.$$

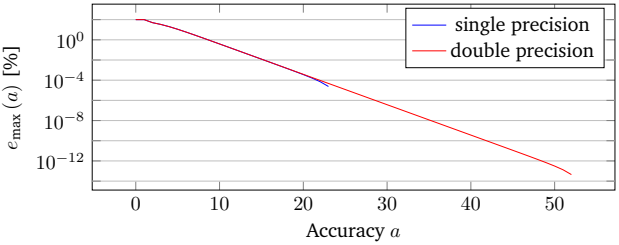


Figure 4.13: Maximum relative error $e_{\max}(a)$ for an anytime division with accuracy a on the x-axis.

B2

Figure 4.13 shows the maximum relative error $e_{\max}$ produced by an anytime division in dependence of a. It can be seen that for single-precision, it suffices to calculate 12 bits to produce a result with a relative error of below 0.1%, which would effectively reduce the latency of the division by half. Also interestingly, if we want to reduce the latency of a double-precision floating-point division by half, the relative error would never exceed a maximum relative error of 0.00001%. The design of a functional unit that implements the anytime division is depicted in Fig. 4.14. The instruction decoder splits an incoming instruction word into the address fields of the dividend `dividend`, divisor `divisor`, and result `target`. Depending on the decoded accuracy field in the instruction indicating the desired accuracy a, the controller terminates the division prematurely, loads the calculated exponent and sign, and sets the appropriate target address and the write enable bit. Initially, both operands are read from the register file and split into the sign, exponent, and mantissa. The calculations for the result's sign bit and exponent are computed in one cycle and stored until the mantissa division, being on the critical path, finishes. The complete division of the mantissa is performed using p pipeline stages. After the controller infers the end of the mantissa calculation the floating-point number is normalised before being written back to the target register.

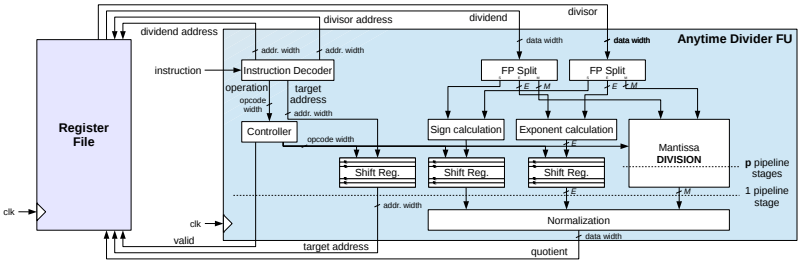


Figure 4.14: Anytime Division Functional Unit.

The latency L_{ADFU} of an anytime instruction, given the number of bits to be calculated a, the length of the mantissa M, the number of pipeline stages p, the latency of the register file Δ_{regf}, and the latency of the normalisation step (= 1) in clock cycles is:

$$L_{\text{ADFU}}(a) = \left\lceil \frac{a}{\left\lceil \frac{M'+1}{p} \right\rceil} \right\rceil + \Delta_{\text{regf}} + 1.$$

B2

As a case study, we implemented an iterative square root computation, the Babylonian method, using the ADFU:

$$\sqrt{S} = \lim_{n\to\infty} x_n; \quad x_0 \approx \sqrt{S} \approx \frac{S}{2}; \quad x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{S}{x_n} \right).$$

The results of the case study are shown in Fig. 4.15. As can be seen, the mean error amounts to less than 0.13% for `DIV_10` instructions after only 10 iterations. By mixing the accuracy of the different iterations (later iterations executed with higher accuracy), the latency of the overall algorithm can even be reduced by up to 54.77% without reducing the accuracy of the end result.

Currently, we are also investigating the analysis and propagation of errors in loop programs when executed using anytime instructions, see [Kön19; Kes+20].

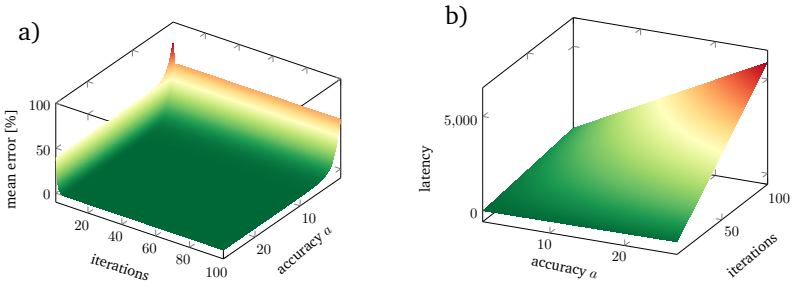


Figure 4.15: *a)* Mean error of an iterative square-root program using an anytime division compared to a reference square-root program specified in C++ with float data types – compiled to an X86 target by the GNU C++ compiler in version 8.1 – where accuracy *a* describes the number of calculated most-significant mantissa bits and iterations how many iterations in the square-root calculation are performed. *b)* The latency of an iterative square-root calculation using an approximate division where accuracy *a* describes the number of calculated mantissa bits and iterations how many iterations in the square-root calculation are performed.

Tile-level TCPA Timing Analysis and Guarantees

TCPAs have been shown to provide timing predictability up to a single clock cycle due to scheduling each iteration as well as each single computation within a loop nest at a statically determined clock cycle [Bra+19b]. This full predictability of performance, however, assumes that the synchronously clocked processor array is not stalled due to the lack of input data or lack of memory to store outbound results. For processing sufficiently large loop nests, the surrounding buffers of a TCPA as shown in Fig. 4.12 must be fed with new input data, respectively results drained early enough, because just a single empty input buffer or a single full output buffer would lead to an immediate freeze of the clock signal and thus stop in processing the loop nest. As a result, techniques are needed to feed and drain the I/O buffers of a TCPA readily to maintain the predictability guarantee. The resulting problems of scheduling communication are illustrated in the following.

A typical TCPA is surrounded by multiple I/O buffers, from which the connected processing elements can read and/or write to, see Fig. 4.12. These buffers serve as an intermediate memory between the processing array and the data source/sink, which will be in the context of an invasive multi-tile architecture as shown in Fig. 4.16. Also, it is necessary to fill those buffers with input data from the originating tile and later sent the written output data back to a receiver tile (see Fig. 4.16). As typically, the amount of I/O data may not fit entirely into the buffers, a continuous buffer refilling and draining during run time is inevitable.

As mentioned before, if any transfer would be too slow and the corresponding buffer has not been filled/drained in time, a TCPA would

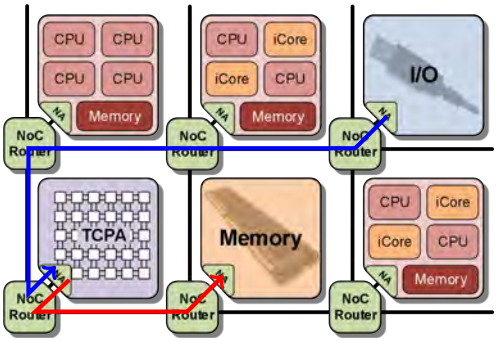


Figure 4.16: Typical tile-level communication of a TCPA. An I/O tile serves as the data source, while a memory tile is used as the data sink.

need to stall its execution until the required data has arrived. Since such a behaviour would break the full predictability as given by a loop schedule, we have to determine and schedule remote DMA (RDMA-like) data transfers between the tiles asynchronously to the next loop execution on the TCPA. Currently, we are developing constraint sets and sound I/O buffer request sequences, where each data transfer can be seen as an instance of a task. Using this model, a proper I/O transfer scheduling problem must be solved such that no deadline will be missed. The final goal is to provide such a tile-to-tile timing predictability also over multiple tiles as shown in Fig. 4.16.

Finally, we are studying the capability of TCPAs to efficiently implement deep learning algorithms such as convolutional neural networks by layer-parallel processing [Hei+19a].

Publications

[Bra+19a] M. Brand, M. Witterauf, F. Hannig, and J. Teich. "Anytime Instructions for Programmable Accuracy Floating-Point Arithmetic". In: *Proceedings of the ACM International Conference on Computing Frontiers (CF)* (Alghero, Sardinia, Italy). ACM, Apr. 30–May 2, 2019, pp. 215–219. DOI: 10.1145/3310273.3322833.

[Bra+19b] M. Brand, M. Witterauf, É. Sousa, A. Tanase, F. Hannig, and J. Teich. "*-Predictable MPSoC Execution of Real-Time Control Applications Using Invasive Computing". In: *Concurrency and Computation: Practice and Experience* (Feb. 2019). DOI: 10.1002/cpe.5149.

[Hei+19a] C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. "Efficient Mapping of CNNs onto Tightly Coupled Processor Arrays". In: *Journal of Computers (JCP)* 14.8 (Aug. 2019), pp. 541–556. DOI: 10.17706/jcp.14.8.541-556.

[Kes+20] O. Keszocze, M. König, M. Brand, and J. Teich. "Error Analysis for Loop Programs Using Anytime Instructions in Approximate Computing". In: *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*. Stuttgart, Germany, 2020.

[Kho19] F. Khosravi. "System-Level Reliability Analysis and Optimization in the Presence of Uncertainty". Dissertation. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Aug. 5, 2019.

[Kön19] M. König. *Approximative Schleifen mit Anytime Instruktionen in der Simulationsumgebung Daisy*. Master Thesis. Friedrich-Alexander University Erlangen-Nürnberg (FAU). Sept. 1, 2019.

B3: Power-Efficient Invasive Loosely-Coupled MPSoCs

Jörg Henkel, Andreas Herkersdorf

Hossein Bardareh, Nguyen Anh Vu Doan, Heba Khdr, Martin Rapp, Mark Sagi, Thomas Wild

The overall goal of Project B3 is to optimise power/energy efficiency under power density and temperature constraints. Within the paradigm of invasive computing, the goal is to ensure that invaded *claims* remain thermally reliable while maintaining the ability that *teams* can invade and execute *i*-lets to *infect* new resources. The pursued objectives are:

Objective 1: Improve power/energy efficiency under power density and temperature constraints.

Objective 2: Develop an adaptive and self-aware system for run-time power, energy, and thermal management.

Objective 3: Refine the run-time model and online estimation of power density and temperature for invasive computing.

The related scientific challenges include the maximisation of the performance under given temperature constraints or under a given energy budget, and minimising the energy consumption or peak power under a certain performance requirement. For this, accurate run-time power information is needed. The derivation of such information from related run-time activity signals is an added scientific challenge which we solve by adapting machine learning algorithms to this problem. In addition, these goals require deep insight into the system, which often is tackled by design-time models. Design-time models, however, perform poorly due to run-time variations and unpredicted conditions coming from the hardware, the environment, and the workloads/applications. Therefore, by applying novel thermal management techniques based on self-awareness and machine learning, we are trying to achieve the objectives without relying on the design-time models as much as possible.

Smart Thermal Management

Heterogeneous multicores are favourable as they have different kinds of processing elements which can provide large performance gains. However, due to the failure of Dennard scaling, power densities are increasing, thereby on-chip temperatures are elevating. To keep the temperature within safe limits, Dynamic Thermal Management (DTM) is implemented on the chip to downscale the voltage and frequency levels of all cores, when the temperature of any core exceeds a predefined thermal threshold. This, however, leads to significant performance losses. In [HKR19], we proposed smart thermal management techniques for heterogeneous multicores that improve thermal efficiency through exploiting heterogeneity parameters both at the chip level and the application level. Here, thermal efficiency is the ability to maximise the performance while keeping the temperature within safe limits.

B3

In [Rap+19b], we proposed a run-time algorithm called PCGov that combines task-agnostic task mapping and task-aware dynamic power budgeting for manycores with shared distributed Last-Level Cache (LLC). PCGov exploits a trade-off between power budget and LLC latency in task mapping while dynamic power budgeting reallocates the power budgets according to the task's execution phases which further increases the performance. Also, it has been shown that our power budget reallocation algorithm is very effective in avoiding thermal violations. Moreover, the proposed PCGov algorithm has a low overhead for both task mapping and power budget reallocation. In [Rap+19a], we further demonstrated that maximum performance in manycores with distributed LLC can not be achieved with static mapping, but task migration is required instead. We proposed a run-time algorithm called PCMig that maximises the performance by doing predictions on task migrations based on both the tasks phases and manycore state. PCMig uses a performance prediction model to rate potential migration candidates before actual task migration by considering the relative impact of power budget and LLC latency.

Circuit ageing has become a major reliability concern in current and upcoming technology nodes. In [AKH19], we focused on the ageing effect from physics to CAD tools. In order to avoid ageing-induced timing errors, we proposed in [KAH19] a paradigm shift in designing guardbands which selects the guardband types on-the-fly with respect to the workload-induced temperatures aiming at optimising for performance under temperature and reliability constraints. Negative Capacitance Field-Effect Transistor (NCFET) has recently attracted significant attention since it exhibits a considerable improvement in the circuit's

performance and energy saving. In [Rap+19c], we presented a novel methodology to model NCFET at the system level to investigate its impact on the performance, power, energy, and cooling trade-offs of a manycore. Since in NCFET, voltage reduction increases the leakage power unlike in conventional CMOS, in [Sal+19], we proposed the first NCFET-aware DVS technique that selects the optimal voltage to minimise the power following the dynamics of workloads.

Thermal and Timing Enforcement

The common practice to avoid thermal emergencies on the chip is to employ a dynamic thermal management (DTM) unit on the hardware. DTM will take countermeasures such as dynamic voltage and frequency scaling (DVFS) and power gating to cool down the chip, if the peak temperature of the chip exceeds a predefined thermal threshold. However, such countermeasures might lead to the violation of real-time constraints of the applications. Therefore, in [Pou+19a], we proposed a thermally composable Hybrid Application Mapping (HAM) methodology that enforces thermal safety proactively at the launch time of applications. This hybrid methodology employs both a thermal-safety analysis at design time and a set of lightweight thermal-safety admission checks to be considered in the mapping-selection process at run time. As a result, the proposed methodology enables providing thermally safe real-time guarantees by preventing DTM interferences.

Thermal-aware Simulation

In order to do manycore thermal simulations in open systems, in [PH19], we presented a toolchain called HotSniper that tightly couples together Sniper manycore simulator, McPat power modelling framework, and Hotspot temperature modelling tool. HotSniper allows for interval thermal simulation of manycores, which is several times faster than the cycle-accurate manycore thermal simulations and at the same time is more accurate than trace-based manycore thermal simulations. The HotSniper toolchain provides efficient means to perform thermal-aware hardware-software codesign of manycore processors in domain of embedded systems.

Accurate Power Estimation

Power and thermal management rely on accurate run-time dynamic power information to take informed and effective management deci-

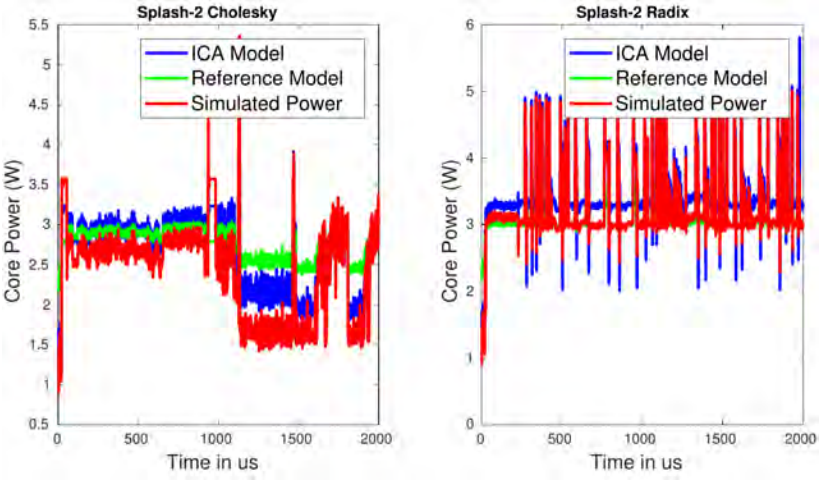


Figure 4.17: Estimated power using ICA compared to estimated power of a state-of-the-art approach and high accuracy power simulations for 2 benchmarks

sions. Measuring power consumption of each core and the uncore is cost prohibitive and often not possible. Therefore, run-time power information is model-based. These techniques usually approximate the switching activity of the logic gates constituting the cores and uncore components through performance counters. However, these performance counters show strong collinearity between each other. The state-of-the-art approach of minimising collinearity relies heavily on so-called microbenchmarks which have to be tailored to specific processor subcomponents to generate subcomponent specific power consumption models. In [Sag+19] we propose a methodology using a representation learning algorithm, i. e. Independent Component Analysis (ICA), to find performance counter representation which automatically minimises collinearity. With this, no microarchitectural knowledge is needed to generate the microbenchmarks to generate power models and generic workloads depicting complex processor activity can be used to generate accurate power models. Resulting power estimations and the actual power are shown in Fig. 4.17.

Publications

[AKH19] H. Amrouch, H. Khdr, and J. Henkel. “Aging Effects: From Physics to CAD”. In: *Harnessing Performance Variability in Embed-*

ded and High-performance Many/Multi-core Platforms. Springer, 2019, pp. 43–69.

[HKR19] J. Henkel, H. Khdr, and M. Rapp. “Smart Thermal Management for Heterogeneous Multicores”. In: *Design, Automation & Test in Europe (DATE)*. IEEE. 2019, pp. 132–137.

[KAH19] H. Khdr, H. Amrouch, and J. Henkel. “Dynamic Guardband Selection: Thermal-Aware Optimization for Unreliable Multi-Core Systems”. In: *Transactions on Computers (TC)* (2019).

[PH19] A. Pathania and J. Henkel. “HotSniper: Sniper-Based Toolchain for Many-Core Thermal Simulations in Open Systems”. In: *Embedded Systems Letters (ESL)* (2019).

[Pou+19a] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel. “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems”. In: *40th IEEE Real-Time Systems Symposium (RTSS)*. 2019.

[Rap+19a] M. Rapp, A. Pathania, T. Mitra, and J. Henkel. “Prediction-Based Task Migration on S-NUCA Many-Cores”. In: *Design, Automation & Test in Europe (DATE)*. IEEE. 2019, pp. 1579–1582.

[Rap+19b] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel. “Power-and Cache-Aware Task Mapping with Dynamic Power Budgeting for Many-Cores”. In: *IEEE Transactions on Computers* (2019).

[Rap+19c] M. Rapp, S. Salamin, H. Amrouch, G. Pahwa, Y. Chauhan, and J. Henkel. “Performance, Power and Cooling Trade-Offs with NCFET-based Many-Cores”. In: *Design Automation Conference (DAC)*. ACM. 2019, p. 41.

[Sag+19] M. Sagi, N. A. V. Doan, T. Wild, and A. Herkersdorf. “Multi-core Power Estimation using Independent Component Analysis based Modeling”. In: *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. Oct. 2019.

[Sal+19] S. Salamin, M. Rapp, H. Amrouch, G. Pahwa, Y. Chauhan, and J. Henkel. “NCFET-Aware Voltage Scaling”. In: *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2019.

B4: Generation of Distributed Monitors and Run-Time Verification of Invasive Applications

Ulf Schlichtmann, Daniel Müller-Gritschneder

Marcel Mettler, Bing Li, Li Zhang

Introduction

The goal of Project B4 in the third funding phase is to provide run-time monitoring support by evolving the monitors developed in the first two phases into a distributed monitoring system capable of supporting run-time verification of user-defined application properties such as latency, throughput, power, reliability and security. The run-time verification chain consists of probes, property checkers and event handlers. Probes are blocks that trace system events or states. Property checkers analyse the probed trace to generate the verdict whether a certain property is violated or validated. The verdict is forwarded to the event handler that can trigger a reaction or generate a log message. In this context, Project B4 addresses the major research challenge to enable run-time verification for invasive multi-tile architectures. This is achieved by working on the following research questions: How can we generate a programmable hardware monitoring system with probes and property checkers and insert software probes and property checkers into the application code? How do we establish communication between all distributed system components? What are the trade-offs between implementing the property checking in hardware blocks, SW annotated to the application or additional monitoring tasks? Also, at design time, the mapping of the application on the platform is unknown. On top, even the exact resource types are unknown as the run-time environment may select from a set of candidate operating points in the invade phase. So another question is how to dynamically configure the system when the mapping information becomes available? Another important aspect is the usability of such a system, which leads to the following questions: Which automation support is required for the configuration of the distributed monitoring system with user-defined properties? Here we plan to investigate two domain-specific languages (DSLs), a so-called Instru-

mentation Language (IL) and a Property Language (PL), as interfaces for the invasive compiler toolchain to define probing data and property checks. We intend to develop an automation tool to generate the run-time verification codes to configure the HW probes and monitors as well as to generate the SW probes and monitors from the IL/PL specification. This leads to a highly automated flow to produce and forward verdicts to event handlers, such as Run-time Requirement Enforcers (RRE) from Project A1 and Project A4.

Specifically, we investigate (a) non-intrusive instrumentation approaches to extract events and system states, (b) distributed monitoring architectures for the verification of functional and non-functional properties, and (c) languages to express run-time verification properties. The target monitoring architecture can be used in a pre-deployment stage for integration testing of software components and post-deployment as fault recovery measure to maintain a safe and secure system state.

Currently, Project B4 generalises its monitoring approaches to address tile-based systems. In the following, the results of our research in 2019 are presented.

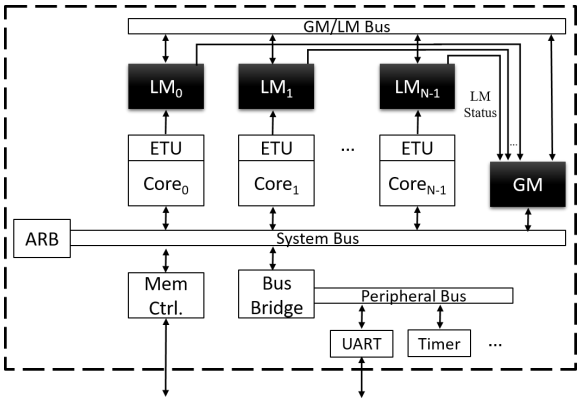


Figure 4.18: Hierarchical Monitoring System for embedded MPSoCs.

Run-Time Monitoring

In [MMS20], we propose a decentralised monitoring architecture for embedded MPSoCs. The hardware-based approach supports the monitoring of inter- and intra-thread requirements for logical and timing supervision. It is built up hierarchically with local monitors for each core and one global monitor. By resource sharing of the monitoring

hardware, it is able to support the concurrent supervision of multiple threads or applications and to reduce the hardware overhead. The monitoring system can be used post-deployment for logical and timing supervision or pre-deployment for run-time verification.

SRAM Ageing - Analysis and Countermeasures

On-Chip SRAMs are an integral part of safety-critical System-on-Chips. At the same time however, they are also most susceptible to reliability threats such as Bias Temperature Instability (BTI), originating from the continuous trend of technology shrinking. BTI leads to a significant performance degradation, especially in the Sense Amplifiers (SAs) of SRAMs, where failures are fatal since the data of a whole column is destroyed. As BTI strongly depends on the workload of an application, the ageing rates of SAs in a memory array differ significantly and the incorporation of workload information into ageing simulations is vital. Especially in safety-critical systems precise estimation of application specific reliability requirements to predict the memory lifetime is a key concern. In [Lis+19] we present a workload-aware ageing analysis for On-Chip SRAMs that incorporates the workload of real applications executed on a processor. According to this workload, we predict the performance degradation of the SAs in the memory. We integrate this ageing analysis into an ageing-aware SRAM design exploration framework that generates and characterises memories of different array granularity to select the most reliable memory architecture for the intended application. We show that this technique can mitigate SA degradation significantly depending on the environmental conditions and the application workload.

As a next step, we address countermeasures to reduce ageing. Usually guardbands are added to the design to prevent failures of the embedded system before its end of life.
In [LMS19], we present the Mitigation of AGIng Circuitry (MAGIC), a low-cost circuitry to effectively mitigate ageing in SAs by wear-levelling. The circuitry consists of an array of XOR gates and a counter. MAGIC modifies the mapping of SRAM banks to physical addresses. Updating the counter value distributes the stress of highly used addresses, e. g. corresponding to program stack data, onto the complete SRAM array. We evaluate the wear-out for on-chip SRAM data memory loads of a typical embedded application. MAGIC mitigates SA degradation up to around 48% for three years of ageing while introducing minimal area and performance overhead.

Timing Optimisation

We also continued and extended our past work on timing optimisation of complex integrated circuits.

In [Zha+20], we cover several techniques that can enhance the resilience of timing of digital circuits. Using post-silicon tuning components, the clock arrival times at flip-flops can be modified after manufacturing to balance delays between flip-flops. The actual delay properties of flip-flops will be examined to exploit the natural flexibility of such components. Wave-pipelining paths spanning several flip-flop stages can be integrated into a synchronous design to improve the circuit performance and to reduce area. In addition, with this technique, it cannot be taken for granted anymore that all the combinational paths in a circuit work with respect to one clock period. Therefore, a netlist alone does not represent all the design information. This feature enables the potential to embed wave-pipelining paths into a circuit to increase the complexity of reverse engineering. In order to replicate a design, attackers have to identify the locations of the wave-pipelining paths, in addition to the netlist extracted from reverse engineering. Therefore, the security of the circuit against counterfeiting can be improved.

Outreach and Further Achievements

In May, we welcomed Andrew B. Kahng (University of California, San Diego) in the InvasIC seminar for a very interesting talk on “On the road to Self-Driving IC Design Tools and Flows” which generated a lot of interest within TUM and from Munich industry.

Ulf Schlichtmann gave two invited talks on topics related to TUM EDA’s invasive computing research. In June 2019, he visited Xidian University (Xi’an, China) together with Dr. Li Zhang for a presentation on “Machine Learning Approaches for Efficient Design Space Exploration of Application-specific NoCs”. In October, he visited the AI College of National Chiao Tung University in Tainan, Taiwan, for a presentation on “Novel Ideas in Timing of Digital Circuits”.

Also, in 2019, Daniel Müller-Gritschneder finished his habilitation procedure successfully [Mue19] and was promoted to “Privatdozent”.

Publications

[LMS19] A. Listl, D. Mueller-Gritschneder, and U. Schlichtmann. “MAGIC: A Wear-leveling Circuitry to Mitigate Aging Effects in Sense Amplifiers of SRAMs”. In: *2019 IEEE 17th International New Circuits and Systems Conference (NEWCAS)*. July 2019.

[Lis+19] A. Listl, D. Mueller-Gritschneder, U. Schlichtmann, and S. Nassif. “SRAM Design Exploration with Integrated Application-Aware Aging Analysis”. In: *Design, Automation, and Test in Europe (DATE)*. Mar. 2019, pp. 1249–1252.

[MMS20] M. Mettler, D. Mueller-Gritschneder, and U. Schlichtmann. “Runtime Monitoring of Inter- and Intra-Thread Requirements on Embedded MPSoCs”. In: 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID) (Jan. 2020).

[Mue19] D. Mueller-Gritschneder. *Advanced Virtual Prototyping and Communication Synthesis for Integrated System Design at Electronic System Level*. 2019.

[Zha+20] G. L. Zhang, M. Brunner, B. Li, G. Sigl, and U. Schlichtmann. “Timing Resilience for Efficient and Secure Circuits”. In: *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2020.

B5: Invasive NoCs and Memory Hierarchies for Run-Time Adaptive MPSoCs

Jürgen Becker, Andreas Herkersdorf

Nidhi Anantharajaiah, Leonard Masing, Sven Rheindt, Akshay Srivatsa

In invasive computing architectures, the invasive NoC (*i*NoC) makes up the basic communication infrastructure among all tiles, enabling and supporting invasive concepts from the application level down to hardware realisation. However, for inter-tile communication, not only data transport is a key issue, but also aspects of memory accessibility and the availability of data close to where it is processed play an important role. In the third funding phase, we look at how the *i*NoC can be made more flexible, and can adapt to different applications at run time to improve performance while still meeting Quality of Service (QoS) requirements. Further, we examine two complementary approaches for reducing the synchronisation overheads and improving the data locality using region-based cache coherence (RBCC) and near memory acceleration (NMA).

Hybrid Prototyping

When developing novel NoC features, the size of the target architecture plays an important role. Many advanced techniques for shortcuts, adaptive routing or multicast are specifically designed for improving traffic flows in large networks. However, existing design and verification methods struggle in this situation. They are either too slow (hardware simulations) or have size and cost limitations (hardware emulation). A hybrid prototype provides a solution to these challenges by combining a Virtual Platform (VP) with an FPGA-based prototype [MLB19]. The approach can be applied to many different use-cases, however it is specifically useful for NoC design since all routers may be mapped onto the FPGA with full cycle-accurate accuracy while the remaining architecture is modelled in a virtual platform. This includes the tiles, memories and network interface, leaving more resources for a larger mesh of routers. The hybrid prototype is realised by an efficient interfacing

based on PCIe. It contains a collector unit that gathers respectively distributes data between the VP side and dummy network adapters on the FPGA that connect to the local ports of each router. A synchronisation mechanism is introduced which enables evaluations beyond a purely functional level. Furthermore, a throughput evaluation highlights the potential of this approach for scalable design and verification.

Block-Based Multicast Routing

Multicast traffic, i. e. communication between one source and multiple destinations is extensively used in large-scale multiprocessor systems. It is a characteristic property of applications using coherency protocols in distributed shared memory systems, neural network implementations and fault-tolerant applications using redundant hardware components to name a few examples.

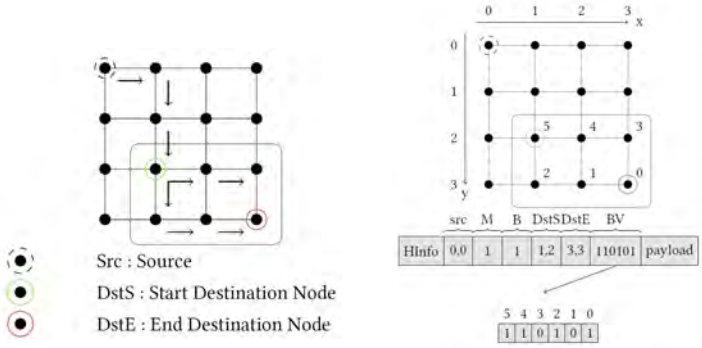


Figure 4.19: (a) A 4x4 Mesh with 6 destinations in a block defined by DstS and DstE (b) A 4x4 Mesh with 4 destination nodes (0,2,4,5) in a block defined by DstS, DstE and a BV.

Usage of multiple unicast packets in the above examples degrades the performance and is an inefficient use of resources. Therefore in [Ana+19] a block-based multicast technique which is dynamic and scalable, with lower packet overhead and latency is proposed. The presented technique uses paths with lower hop count when compared to multiple unicasts with Dimension Order Routing (DoR). It also has a decrease in Address Overhead (AO) of 33% for a 4x4 Mesh with 6 destinations and 79% for 128x128 for 16 destinations compared to all destination encoding commonly used in multicast techniques.

In the following, an overview of the block-based concept using an example is provided. When a source needs to send data to a group of destinations, a block is identified which contains all the potential destination nodes as illustrated in Figure 4.19(a). A bit vector (BV) field shown in Figure 4.19(b) is used to indicate which are the destination nodes within the block. Since the block can be defined at run time, dynamic mapping of bit vector index onto the nodes within the block is implemented. This is explained using an example shown in the figure. Source node is (0,0), destinations are (1,2),(2,2),(1,3) and (3,3). A block is defined by DstS (1,2) and DstE (3,3). There are four destinations within this block identified by the BV. BV indices are mapped to the nodes within the block as shown.

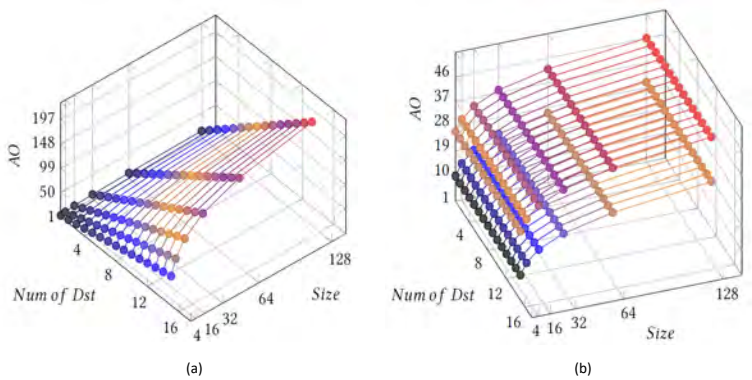


Figure 4.20: (a) AO of a multicast when all destination encoding is used and (b) AO of a block-based multicast with (above graph) and without (below graph) bit vector, respectively.

The block-based technique is beneficial especially for large mesh networks as the address size gradually increases, which contributes to greater Address Overhead (AO) in multicast packets. To illustrate the benefit of the block-based technique, 16 potential destinations grouped together in a block of 4x4 in a 2D Mesh network are used as an example. Different network sizes of 4x4, 8x8, 16x16, 32x32, 64x64 and 128x128 are considered. In Figure 4.20(a) AO of multicast packets using all destination encoding is presented, and results of block-based multicast is shown in Figure 4.20(b). AO of block-based multicast scales better and has a 79% decrease in overhead for 128x128 NoC when compared to all destination encoding.

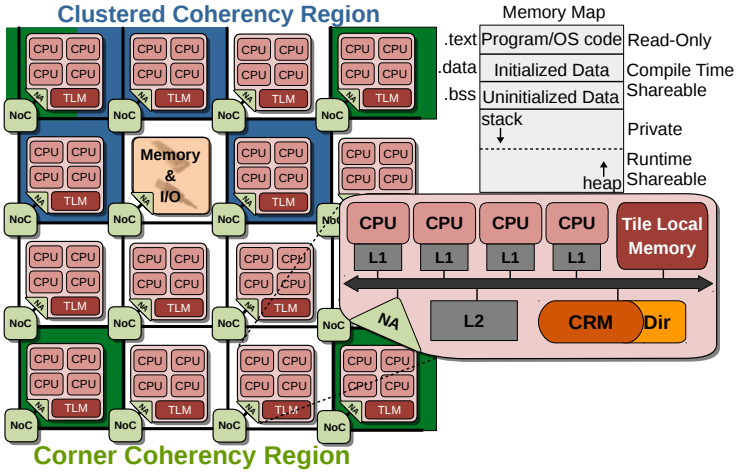


Figure 4.21: A 4×4 InvasIC architecture with multiple coherence regions.

Modern MPSoCs have evolved into tile-based systems with physically distributed memory architectures. To be able to use these systems with the classical shared memory programming paradigm, cache coherence is an essential aspect. Global coherence spanning all tiles does not scale well and is not even necessary for applications with limited degrees of parallelism. We presented a region-based cache coherence (RBCC)[10] concept, which confines coherence support to a selectable cluster of tiles. A hardware Coherency Region Manager (CRM) module (Fig. 4.21) was designed to dynamically configure coherency regions, specific to each application's requirements, enabling a shared memory programming environment. This RBCC was extended with *RBCC-malloc()* [Sri+19] which tailors coherence to actually shared application working-sets within the coherency regions at run time. As an example, the feature extraction task of a video streaming application (developed by Project D1) was executed for both shared memory (enabled by RBCC) and message passing modes, on our FPGA prototype. Experiments revealed an application acceleration of up to 42% when using shared memory mode compared to a message passing-based implementation, and a

[10] A. Srivatsa, S. Rheindt, T. Wild, and A. Herkersdorf. "Region Based Cache Coherence for Tiled MPSoCs". In: *2017 30th IEEE International System-on-Chip Conference (SOCC)*. Sept. 2017.

reduction in directory resources (BRAM) by 57% compared to global coherence for a region size of 4-tiles.

Previously, the CRM only supported coherence for the Tile-Local Memory (TLM). Now, this has been extended to include the global DDR memory, allowing applications with large data-sets to also use the RBCC concept. RBCC also supports flexible coherency regions at run time through the CRM's re-configuration sub-module. When an application wants to expand/shrink the coherency region, the re-configuration sub-module performs the necessary context switches, like clearing the directory sharer information entries and updating the configuration tables.

Current research in this area focuses on executing and evaluating shared memory workloads (PARSEC, SPLASH-2) on our FPGA prototype in collaboration with Project C1. This not only requires the existing hardware coherence mechanisms, but also additional features like false-sharing resolution and barrier support. The false-sharing problem is resolved in hardware by detecting and writing-back only modified data in a fine-granular manner. For applications that use barriers for synchronisation, a *CRM-barrier()* instruction was implemented which notifies the application when all coherence messages have been successfully executed and acknowledged.

Near-Memory Acceleration

The recent trend towards tile-based manycore architectures has helped to tackle the memory wall by physically distributing memories and processing nodes. However, new challenges for MPSoCs arose due to the emergence of ever increasing memory intensiveness of applications with big, irregular and cache unfriendly data sets. Distributed operating systems and applications face a data-to-task locality challenge. Many recent approaches therefore leverage in- or near-memory computing to reduce energy-hungry and performance-degrading data movement and instead perform the computation close to where the data is stored.

Our work in collaboration with Project C1 on software-defined hardware-managed queues enables efficient inter-tile communication by leveraging application-specific queues with arbitrarily sized elements. Queue and memory management, intra- and inter-tile data transfer, and task invocation are entirely handled by a dedicated near-memory hardware module. Only the dynamic queue creation at run time is performed in software. The evaluation with the MPI-based NAS benchmarks shows a reduction in execution time by up to 48% for the communication

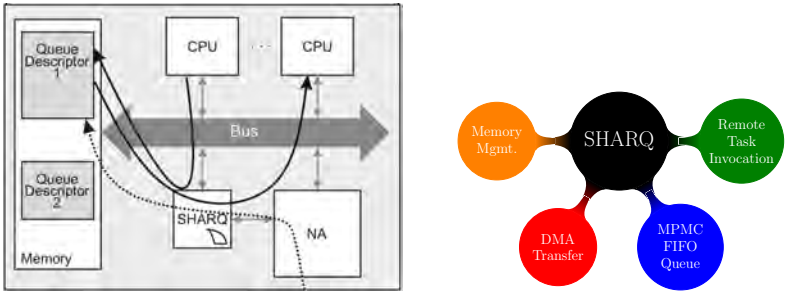


Figure 4.22: Architecture and features of SHARQ.

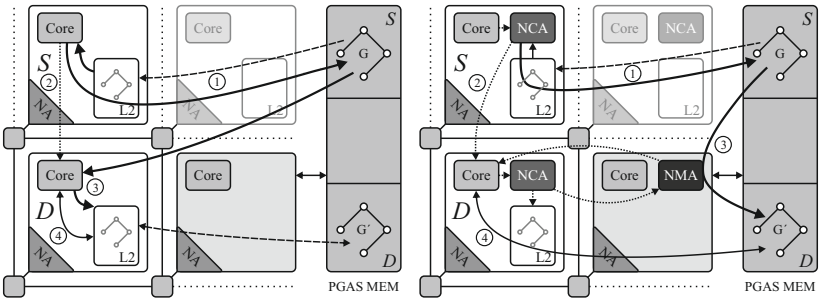


Figure 4.23: Far-from memory Pegasus (left) vs. near-memory NEMESYS (right).

intense IS kernel in a 4x4 tile design on an FPGA platform with a total of 80 LEON3 cores [Rhe+19b].

As some tile-based architectures omit inter-tile cache coherence, they require a different programming model based on explicit messages. Inter-tile communication in e. g. the partitioned global address space (PGAS) programming paradigm is allowed via a remote procedure call (RPC)-like programming language construct. The more modern PGAS languages are object-oriented and thus require the transfer of object graphs as well.

In a collaboration with Project C3, we developed NEMESYS: NEar-Memory Graph Copy Enhanced SYstem-Software, which outsources the memory-intensive and cache unfriendly graph copy operation to a near-memory hardware accelerator. The operation is not only performed near-memory, but also the hardware accelerator relieves the CPUs from the graph copy duty (similar to a DMA unit for non-pointered data). The evaluation with the X10 IMSuite benchmarks, featuring distributed graph algorithm kernels, showed a speedup in execution time between

1.35x and 3.85x compared to a state-of-the-art approach. A comparison of the mechanisms is depicted in Fig. 4.23 [Rhe+19a].

Due to the high importance of data-to-task locality and the relevance of efficiently transferring queue elements and object graphs, both contributions [Rhe+19b; Rhe+19a] provide performant and scalable solutions to mitigate the locality wall.

Publications

[Ana+19] N. Anantharajaiah, F. Kempf, L. Masing, F. M. Lesniak, and J. Becker. "Dynamic and Scalable Runtime Block-based Multicast Routing for Networks on Chips". In: *Proceedings of the 12th International Workshop on Network on Chip Architectures*. NoCArc. Columbus, Ohio: ACM, 2019, 10:1–10:6. DOI: 10.1145/3356045.3360718.

[MLB19] L. Masing, F. Lesniak, and J. Becker. "Hybrid Prototyping for Manycore Design and Validation". In: *Applied Reconfigurable Computing*. Springer International Publishing, 2019, pp. 319–333.

[Rhe+19a] S. Rheindt, A. Fried, O. Lenke, L. Nolte, T. Wild, and A. Herkersdorf. "NEMESYS: Near-Memory Graph Copy Enhanced System-Software". In: *MEMSYS 19: The International Symposium on Memory Systems*. Washington DC, 2019.

[Rhe+19b] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf. "SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures". In: *Proceedings of the 19th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2019.

[Sri+19] A. Srivatsa, S. Rheindt, D. Gabriel, T. Wild, and A. Herkersdorf. "CoD: Coherence-on-Demand – Runtime Adaptable Working Set Coherence for DSM-Based Manycore Architectures". In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by D. N. Pnevmatikatos, M. Pelcat, and M. Jung. Cham: Springer International Publishing, 2019, pp. 18–33.

C1: Invasive Run-Time Support System (*i*RTSS)

Lars Bauer, Jörg Henkel, Timo Hönig, Wolfgang Schröder-Preikschat

Gabor Drescher, Christoph Erhardt, Tobias Langer, Sebastian Maier, Jonas Rabenstein, Florian Schmaus

Project C1 investigates operating-system support for invasive applications. It provides methods, principles and abstractions for the application-aware *extension*, *configuration* and *adaptation* of invasive computing systems. These are technically integrated into the *invasive Run-time Support System* (*i*RTSS), a highly scalable native operating system in close contact and constant touch with a standard Unix-like host operating system. The project works address special-purpose MPSoC-based as well as general-purpose multicore/manycore machines.

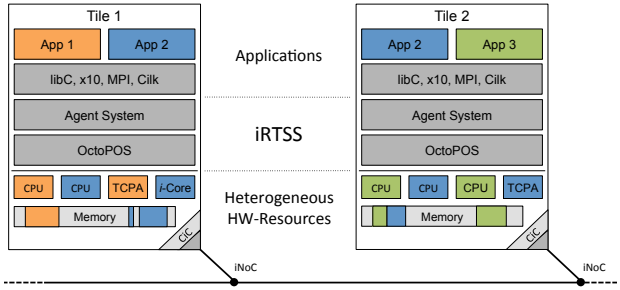


Figure 4.24: *i*RTSS architecture on a multi-tile system. Colours indicate invaded resources.

Architectural Overview

Figure 4.24 provides a high-level view of the current *i*RTSS architecture. Key elements are OctoPOS[11], the parallel operating system (POS) that

[11] Prefix 'Octo' indicates the 8th generation within a particular family of special-purpose operating systems—but also refers to a nature which is highly parallel in its actions as well as adaptable to its environment: the octopus, being able to act in parallel by means of its tentacles, adapt itself through colour change, and, due to its highly developed nervous system, attune to dynamic environmental conditions and impact.

implements the *mechanisms* of *i*RTSS to make all capabilities of the underlying hardware available to higher (software) levels, and the agent system, which provides global *i*RTSS *strategies* for resource management through means of self-adaption to cope with the scalability problem in large multicore systems, logically residing between the run-time libraries for various kind of invasive-parallel applications and the OctoPOS kernel. OctoPOS makes the computing platform (LEON, x86-64) accessible to processes as a "virtual machine" that functions either as native ("satellite") neighbouring to or as guest ("planet") managed by a host operating system (Linux). The key aspect in the design and development of OctoPOS is to make all the capabilities of the underlying hardware available to higher (software) levels in an *unfiltered* way. In the reporting phase, particular attention was paid to developing concepts for operating-system support for RRM and RRE (Figure 4.25; Project A1, see also [Tei+20a]).

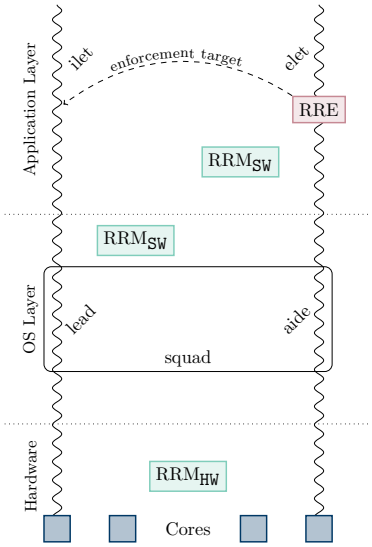


Figure 4.25: Squads: Operating-system support for RRM and RRE.

C1

Background Noise and Energetic System Software

With the OctoPOS kernel, Project C1 provides an alterable operating systems kernel which supports vertical migration of system functions in terms of hardware offerings and software demands. The main concerns are to achieve smooth user-/system level transitions of predictable time/energy demand along with *i*RTSS calls (downward direction) and signals (upward direction).

Negative effects that are caused by background noise within the system are loss of performance and increased energy demand. To mitigate such effects, our system software combines measurement-based techniques to provide predictions on the expected resource demand. With analysis techniques that are based on neural networks, accurate energy demand estimations are generated [HHS19]. This includes online estimations for sequences of program code (i. e. *i*-lets) that have not been assessed in an offline analysis previously. At the operating system level,

we investigate the effect of contention on the energy demand [Rei+19] to adapt the system's resource demand (i. e. energy and time) dynamically at run time. Fit-to-measure operating systems improve system responsiveness and ensure low resource demands. Such systems are achieved by tailor-made system software that adapts the operating system structure statically to the underlying hardware architectures [Hei+19b]. Cross-cutting concerns that occur dynamically at run time require holistic analysis methods that consider both, energy demand and time demand (i. e. performance) [Sie+19]. In particular, we combine automated worst-case analysis to enforce a smooth operation of the overall system by reducing unnecessary background noise.

Asynchronous Abstract Machines

In many systems, the concurrent execution of different applications may lead to interference, which surfaces in the form of increased cache/TLB misses, bad branch prediction and a decreased *instructions-per-cycle* performance of the CPU. OctoPOS can limit this interference through the exclusive allocation of cores to specific applications (in the form of claims). However, noise and interference also occur within applications when they perform heterogeneous work (e. g. executing different threads, thread pools) or when they interact with other parts of the system (like OS or library code).

C1

Therefore, we now went one step further with our new system design that is based on *asynchronous abstract machines* [Mai+19]. It allows for additional partitioning within applications and even the operating system by structuring the whole system into a number of *machines*, as shown in Fig. 4.26. These machines feature a light-weight task scheduler and are dedicated to a specific group of homogeneous tasks (i. e. shared code and data). They offer an asynchronous, task-based interface for efficient interaction between machines via messages. A dedicated OS component, which is aware of all machines in the system, is responsible for dynamic and exclusive allocation of cores to machines depending on their current workload. With *asynchronous abstract machines*, our goal is to make frequent transitions between homogeneous workloads fast (by scheduling light-weight tasks within machines) and costly transitions between heterogeneous workloads rare (by dedicating cores to specific machines for extended periods). Our implementation in an independent prototype features a variety of reusable machines (e. g. for file/network I/O, database access, encryption, compression), and its performance evaluation showed promising results.

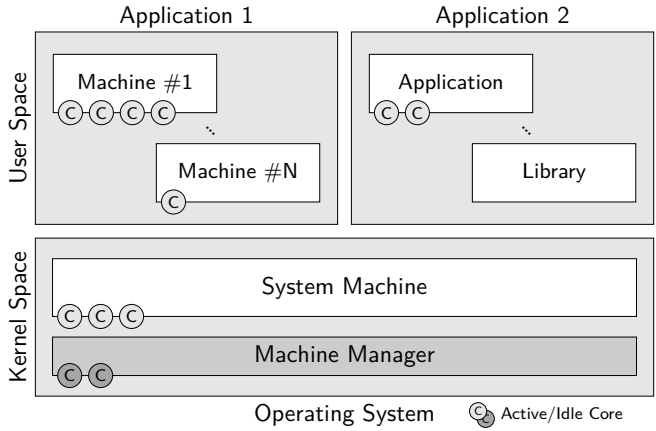


Figure 4.26: Overview of a machine-based system and its core allocations [Mai+19].

Software-defined Hardware-managed Queues

C1

In close collaboration with Project B5 on *software-defined hardware-managed queues* (SHARQ) for inter-tile communication [Rhe+19b], we were able to combine the flexibility of traditional software queues (i. e. dynamic creation, arbitrary queue length and element size) with the performance-benefits of hardware-acceleration (i. e. inter-tile DMA transfers, remote atomic operations). With SHARQ it is possible to avoid an additional roundtrip for memory allocation that is typically required when transferring data to other tiles for remote processing. In addition, the close interaction between SHARQ and the C*i*C allows to schedule handler tasks on the receiver side automatically if required.

After an extensive requirement analysis, the HW/SW interface specification, and the implementation of the required hardware and software modules, several use-cases for SHARQ could be identified. We are currently using SHARQ in the implementation of our distributed TCP/IP network stack, for the internal communication of our MPI library and to offload system tasks to dedicated cores efficiently. In the future, SHARQ could also offer a hardware-acceleration for the communication between *asynchronous abstract machines* even across isolation domains. Our evaluation with the MPI-based NAS parallel benchmarks (shown in Fig. 4.27) and several microbenchmarks showed significant performance benefits compared to software-only approaches.

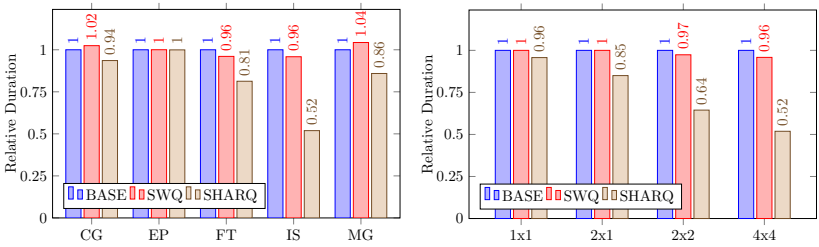


Figure 4.27: Performance and tile scalability of the NAS parallel benchmarks for our original MPI implementation (BASE), for a variant using software queues (SWQ) and for a SHARQ-accelerated variant (SHARQ) [Rhe+19b].

Virtual Shared Memory

In tile-based MPSoCs, tiles dedicated for computation usually are provided with a relatively small local memory. Access to larger memory is provided via interconnect/NoC. In these setups, cache coherency and memory consistency are generally only maintained within the scope of a tile. This provides better system scalability since e. g. no coherency protocols have to be implemented in the NoC. This, however, poses the challenge of efficiently accessing memory via NoC: Data flow dependent applications have to take care to both, enforce the order of any remote writes and to explicitly flush caches when reading from remote memory in order to guarantee correctness.

C1

To address this issue, we investigate the implementation of virtual shared memory (VSM) in OctoPOS. VSM establishes a shared memory view to the system memory with consistency guarantees by means of the operating system. To this end, remote memory is page-wise locally replicated. Applications then transparently work on the local copies. Consistency with the original memory page is restored with explicit system calls for establishing memory consistency. After issuing such a system call, any local changes are applied back to the original page and thus become globally visible.

A first API has been proposed and a prototype of the VSM mechanism has been integrated into OctoPOS and is currently explored. We are planning to evaluate the benefits of different consistency models in respect of parallel workloads with the prototype.

Publications

[Hei+19b] B. Heinloth, M. Ammon, D. Nguyen, T. Hönig, V. Sieh, and W. Schröder-Preikschat. “Cocoon: Custom-Fitted Kernel Com-

piled on Demand". In: *Proceedings of the 10th Workshop on Programming Languages and Operating Systems (PLOS)*. ACM. New York, NY, USA: ACM Digital Library, 2019, pp. 1–7. DOI: 10.1145/3365137.3365398.

[HHS19] T. Hönig, B. Herzog, and W. Schröder-Preikschat. "Energy-Demand Estimation of Embedded Devices Using Deep Artificial Neural Networks". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*. ACM Digital Library, 2019, pp. 617–624. DOI: 10.1145/3297280.3297338.

[Mai+19] S. Maier, T. Hönig, P. Wägemann, and W. Schröder-Preikschat. "Asynchronous Abstract Machines: Anti-noise System Software for Many-core Processors". In: *Proceedings of the 9th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)* (Phoenix, AZ, USA). ACM, 2019, pp. 19–26. DOI: 10.1145/3322789.3328744.

[Rap+19a] M. Rapp, A. Pathania, T. Mitra, and J. Henkel. "Prediction-Based Task Migration on S-NUCA Many-Cores". In: *Design, Automation & Test in Europe (DATE)*. IEEE. 2019, pp. 1579–1582.

[Rap+19b] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel. "Power-and Cache-Aware Task Mapping with Dynamic Power Budgeting for Many-Cores". In: *IEEE Transactions on Computers* (2019).

[Rei+19] S. Reif, P. Raffeck, H. Janker, L. Gerhorst, T. Hönig, and W. Schröder-Preikschat. "Earl: Energy-Aware Reconfigurable Locks". In: *Proceedings of the 9th Embedded Operating Systems Workshop (EWiLi)*. Forthcoming. ACM. New York, NY, USA: ACM SIGBED Review, 2019.

[Rhe+19b] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf. "SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures". In: *Proceedings of the 19th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2019.

[Sch+19a] S. Schuster, P. Wägemann, P. Ulbrich, and W. Schröder-Preikschat. "Proving Real-Time Capability of Generic Operating Systems by System-Aware Timing Analysis". In: *Proceedings of the 25th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE Computer Society, 2019, pp. 313–330. DOI: 10.1109/RTAS.2019.00034.

[Sie+19] V. Sieh et al. "Combining Automated Measurement-Based Cost Modeling With Static Worst-Case Execution-Time and Energy-Consumption Analyses". In: *IEEE Embedded Systems Letters* 11.2 (June 2019), pp. 38–41.

C1

[Tei+20a] J. Teich, P. Mahmoody, B. Pourmohseni, S. Roloff, W. Schröder-Preikschat, and S. Wildermann. “Run-Time Enforcement of Non-functional Program Properties on MPSoCs”. In: *A Journey of Embedded and Cyber-Physical Systems*. Ed. by J.-J. Chen. Springer, 2020.

[WS19] A. Würstlein and W. Schröder-Preikschat. “T-IBE-T: Identity-Based Encryption for Inter-Tile Communication”. In: *Proceedings of the 12th European Workshop on Systems Security (EuroSec)*. ACM Digital Library, 2019, pp. 1–6. DOI: 10.1145/3301417.3312500.

C3: Compilation and Code Generation for Invasive Programs

Gregor Snelting, Jürgen Teich

Jorge A. Echavarria, Andreas Fried, Frank Hannig, Michael Witterauf

Project C3 investigates compilation techniques for invasive computing architectures. Its central role is the development of a compiler framework for code generation as well as program transformations and optimisations for a wide range of heterogeneous invasive architectures, including RISC cores, tightly-coupled processor arrays (TCPAs), and *i*-Core reconfigurable processors.

In the second funding phase, a major focus of research has been the quest for (higher) predictability of non-functional aspects of invasive parallel program execution, i. e. performance, fault tolerance, and security.

In the current third funding phase, our major research focus is on run-time requirement enforcement of multiple non-functional execution qualities such as user-specified performance and energy corridors. This includes (a) constant latency/throughput loop processing, (b) approximate loop processing including language extensions to specify and analyse errors of imprecise loop variables, and (c) a symbolic code generator for TCPA targets to support symbolic loop parallelisation techniques. Moreover, we are investigating (d) compiler optimisations for the *i*-Core, (e) optimisations for FPGAs, and (f) discovery of invasive programming patterns.

We now shortly summarise results on individual aspects achieved during 2019.

Symbolic Loop Compilation for TCPAs

TCPAs exploit all levels of parallelism that loops provide, including iteration-level parallelism. Iteration-level parallelism permits the overlapped processing of subsequent iterations of a loop to support resp. increase instruction-level parallelism, but as a consequence puts more pressure on the number of functional units. That is because not hav-

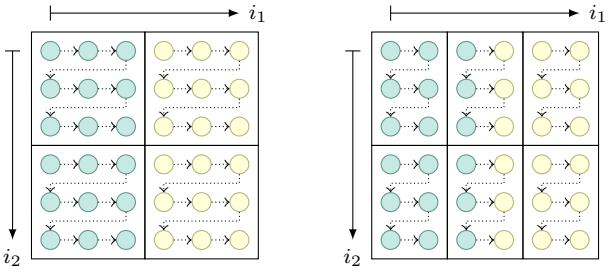


Figure 4.28: A 2×2 (left) and 2×3 (right) tiling of a 6×6 loop program. Each tile is sequentially executed by one processing element, as indicated by the dashed edges. The two colours represent two parts of the iteration space with different instructions. As can be seen, tiling influences both the instructions and the control flow within a processing element and thus the generated program.

ing enough functional units raises the *initiation interval* of a loop nest, the interval between starting two consecutive iterations, a crucial performance metric of loops. To lighten the burden on functional units, mutually exclusive instructions—that means instructions from statements in disjoint parts of the iteration space (for example, input and output instructions)—may be scheduled in the same relative time slot on the same functional unit. However, this sharing makes the control flow iteration-dependent, which necessitates the generation of a set of *distinct programs* for the processing elements (see Fig. 4.28).

C3

Generating this set of programs must be *symbolic* because (a) loops usually have parametric bounds and (b) in invasive computing, we do not know how many processing elements are available until run time. Therefore, we know neither how many distinct programs must be generated for the processors of a TCPA and which instructions they contain. Alternatives to symbolic code generation, like generating and storing all possible programs or full just-in-time compilation, are prohibitively expensive, especially in embedded systems. In 2019, we proposed as a breakthrough in loop compilation the first solution to symbolic code generation as a hybrid approach that uses a representation for symbolic programs, called *polyhedral syntax trees* [WHT19].

The idea is to frontload all intractable sub-problems (in particular the NP-hard modulo scheduling; compare Fig. 4.29) of mapping a loop onto a TCPA to compile time and at run time only perform the steps that require knowledge of the concrete loop bound parameters and number of allocated processing elements. A polyhedral syntax tree, as shown and explained in Fig. 4.30, is a data structure that connects these two

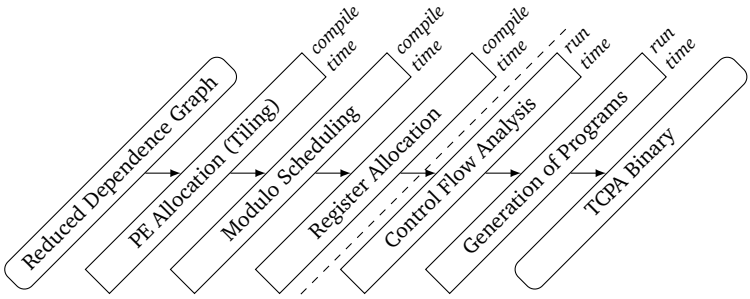


Figure 4.29: In our hybrid compile flow for symbolic loop compilation, all intractable problems (in particular modulo scheduling and register allocation) are solved at compile time. At run time, only efficient (that is, polynomial in time and space) problems remain.

phases in an elegant and efficient manner and from which all potential programs can be assembled at run time.

In [WHT19], we showed how to systemically generate polyhedral syntax trees from a loop description and that using them is both space- and time-efficient: they require polynomial space to store—whereas storing all possibly generated programs is non-polynomial—and poly-

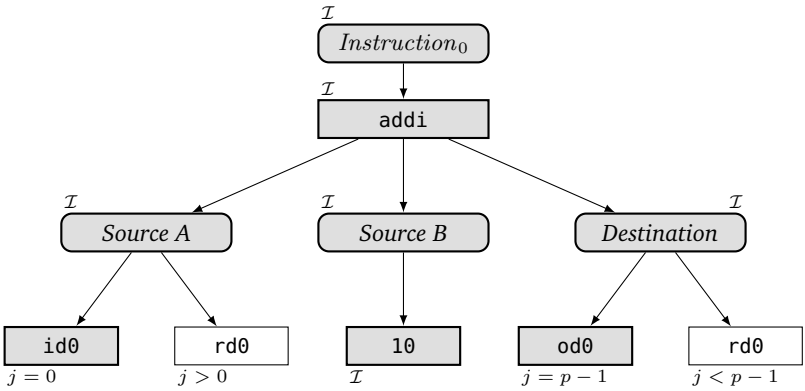


Figure 4.30: Partial polyhedral syntax tree for a symbolic instruction. Each node v is annotated with its *domain* $\mathcal{I}_v$, represented as a polyhedral set of integers. To evaluate the tree—i. e. to obtain a concrete instruction—at an iteration $I = (j)$, all nodes where $I \notin \mathcal{I}_v$ are discarded. The resulting sub-tree represents a syntactically correct concrete instruction. In the figure, the resulting instruction for the edge case where the tile is only one iteration wide ($p = 1$) and $I = 0$ is marked with a shaded background. It results in the single instruction `addi od0 10 id0` to be executed at $I = 0$.

nomial time to evaluate—whereas just-in-time compilation requires solving NP-hard problems. For example, in a case study, we showed for a representative loop program that using a polyhedral syntax tree saves 98.88% of space compared to storing all program variants.

Synthesis of *i*-Core Special Instructions

We have, extended our compiler libFIRM with a new VHDL back end, which will be able to support more optimisations than the old ad-hoc code generation approach. We are confident to see first results in this area in 2020.

Near-memory Computing

The InvasIC MPSoC is a distributed-memory system, so the CPU cores on the tiles are located "far from" the main memory: Main memory accesses need to traverse the NoC, which incurs a latency of at least 100 cycles. We therefore expect large performance gains by running some recurring computations "near memory", i. e. on dedicated hardware positioned close to the main memory controller.

We concentrate our efforts on system software, since that is used by every application, and programs need not be rewritten in order to benefit from improved system software. Moreover, our X10-based PGAS run-time system offers a high-level programming model, which needs to be well-optimised to avoid overhead in every program.

Therefore, we have developed NEMESYS, Near-Memory Graph Copy Enhanced System-Software in cooperation with Project B5 [Rhe+19a]. With NEMESYS, we tackle the problem that our PGAS programming model requires complete *object graphs* to be transferred between tiles. For example, in the code snippet

```
val x = computeX();
at (tile2) { val y = x.getY(); }
```

`x` (the *root*) and all objects transitively reachable from it need to be copied to `tile2`. A classical DMA unit is insufficient for this task as the objects are not necessarily allocated in a contiguous block of memory.

The core component of NEMESYS is a *near-memory accelerator* which uses a hardware implementation of the Deutsch/Schorr/Waite algorithm to traverse the source object graph and copy the objects to the destination. This algorithm is particularly suited for hardware implementation since it can do without a recursion stack by storing reverse pointers in the objects themselves. Figure 4.31 shows the basic steps of

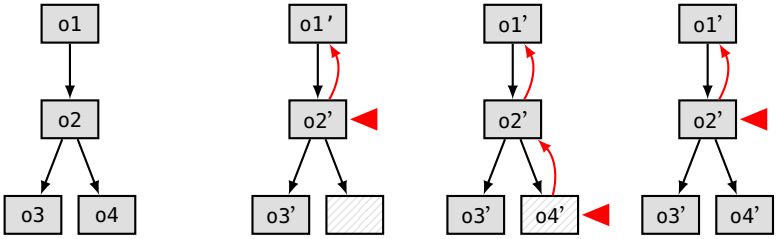


Figure 4.31: Object graph traversal in NEMESYS. On the left, the source graph is shown, and to its right the steps needed to copy o4 to o4'. The red triangle marks the object currently under consideration by the NMA. First, the NMA allocates a new object and sets o2 to point to it. It then moves to the new object, setting the reverse pointer, and copies o4 to it, recursively following its pointers if necessary. Finally, it returns to o2' by following the reverse pointer.

the algorithm: allocating a new object, copying it recursively, and going back to the parent object using the reverse pointer.

In addition, the NMA needs to detect and re-use objects it has already copied in order to preserve the graph structure. Otherwise, the destination graph would contain two copies of an object which is reachable via two different paths from the root.

NEMESYS accomplishes this with a hardware hash table using the classical universal hash function H_3 due to Carter and Wegman. To hash n bits of input down to k bits, H_3 defines a set of functions h_M parameterised by a $k \times n$ bit matrix M. The hash $h_M(x)$ is then given by $h_M(x) = \bigoplus_{i=0}^{n-1} M_i x_i$, where M_i is the i-th column of M, x_i is the i-th bit of x, and $\oplus$ is the exclusive or operation. This function is easily and cheaply implemented in hardware, only requiring nk XOR gates for a fixed matrix.

Evaluating the performance of NEMESYS on the X10-imsuite benchmarks, we find that we achieve speedups between $1.35\times$ and $3.85\times$ compared to Mohr's PEGASUS technique.

Further Results

Faramarz Khosravi has successfully defended his PhD thesis [Kho19] with important results on the reliability analysis of non-redundant and replicated loop programs on TCPAs.

Moreover, we are investigating deep learning applications and their mapping on TCPAs. In [Hei+19a], we have shown that they could enable the layer-parallel (resp. pipelined) processing of multiple layers of CNNs quite efficiently concerning energy and memory requirements.

With respect to GPUs, we expect an order of magnitude lower energy consumption for executing the loop nests of such nets.

Publications

[Bra+19b] M. Brand, M. Witterauf, É. Sousa, A. Tanase, F. Hannig, and J. Teich. "*-Predictable MPSoC Execution of Real-Time Control Applications Using Invasive Computing". In: *Concurrency and Computation: Practice and Experience* (Feb. 2019). DOI: 10.1002/cpe.5149.

[Hei+19a] C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. "Efficient Mapping of CNNs onto Tightly Coupled Processor Arrays". In: *Journal of Computers (JCP)* 14.8 (Aug. 2019), pp. 541–556. DOI: 10.17706/jcp.14.8.541-556.

[Kho19] F. Khosravi. "System-Level Reliability Analysis and Optimization in the Presence of Uncertainty". Dissertation. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Aug. 5, 2019.

[Rhe+19a] S. Rheindt, A. Fried, O. Lenke, L. Nolte, T. Wild, and A. Herkersdorf. "NEMESYS: Near-Memory Graph Copy Enhanced System-Software". In: *MEMSYS 19: The International Symposium on Memory Systems*. Washington DC, 2019.

[WHT19] M. Witterauf, F. Hannig, and J. Teich. "Polyhedral Fragments: An Efficient Representation for Symbolically Generating Code for Processor Arrays". In: *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)* (San Diego, CA, USA). IEEE, Oct. 9–11, 2019.

C5: Security in Invasive Computing Systems

Felix C. Freiling, Wolfgang Schröder-Preikschat, Gregor Snelting, Ingrid Verbauwhede (Mercator Fellow)

Simon Bischof, Franziska Schirrmacher, Pieter Maene, Furkan Turan

Project C5 explores security aspects of invasive computing and resource-aware programming. Invasive MPSoC architectures will only be accepted if basic security properties are supported. The final goal is to ensure confidentiality, integrity, and availability in the presence of untrustworthy programs that compete for resources and/or can contain malicious functionality. This requires a comprehensive approach, addressing both hardware and software mechanisms.

In the current third funding phase, we focus on integrating information flow control (IFC) with control flow attestation (CFA) mechanisms that can reliably and provably assess the information leakage from an executed code. In addition, we aim at enforcing security properties that have been requested by applications at run time even if attacker assumptions are violated, thereby increasing the assumption coverage.

The year 2019 saw the successful PhD defence of the long-time Project C5 collaborator Pieter Maene of KU Leuven for his thesis entitled "Lightweight Roots of Trust for Modern Systems-on-Chip" [Mae19]. Furthermore, we made progress in several work packages. For example, at the end of 2019, we have already finished a prototype implementation for yielding a mutual attestation mechanism to check and (re-)enforce the integrity of computations on remote tiles (in collaboration with the Mercator fellow, work package C5.6). The progress in work packages C5.2 and C5.5 is explained below.

Attacker Model

Our attacker model consists of four hierarchy levels at which an attacker can operate and execute software. An *X10-attacker* corresponds to an attacker who can run programs written in X10. These programs can be statically checked through a trusted X10 compiler, strongly inhibiting malicious behaviour, e. g. by the type system of X10. In

contrast, the *binary attacker* may execute arbitrary (binary) code that runs with privileges associated with normal applications (usually user-level privileges). With the *OS-level attacker*, we allow an attacker to take over control of the operating system. Finally, *physical attacks* are the most powerful ones which are considered to be technically difficult to perform, but also to defend against. Physical attacks are not considered in the project.

Results

Traditionally, information flow control (IFC) checks for non-interference as security property. This results in a binary answer: Either the secret inputs of the program cannot have any influence on the program output or the security analysis finds that there may be such an influence, however slight. In the latter case, the program is rejected as insecure. In certain cases, however, such influence cannot be prevented since it is part of that program's functionality. With quantitative IFC, one can deduce bounds on how many bits a program can reveal, giving a much more fine-grained answer than traditional IFC. As a part of a master's thesis, a quantitative IFC algorithm has been integrated into JOANA. It uses a data flow analysis to calculate constant bits and dependency relations between different bits in the program. This results in a dependency graph on the bit level. A minimal cut is then computed to obtain a minimal number of bits which determine the output of the program (see Figure 4.32). Thus, we get a sound approximation of the leakage. For if-statements, we increase precision in the then- and else-branch by using guarantees we get from knowing that the if-condition must have been true or false, respectively. By using techniques based on summary edges, the analysis is interprocedural, including arbitrary recursion. While traditional quantitative IFC only works for batch programs, our approach has been extended to deal with interactive programs, allowing user input and output at arbitrary program points.

C5

Besides security within a tile using CFI, we worked on extending the trusted execution environment from one tile to another [TV19b]. For this specific purpose, we introduced the idea of mutual attestation, which allows the tiles to directly evaluate each other, and manage their own cooperation. They start with attesting the measurements of their resources to each other. In this context, the resources refer to code and data files, and predefined memory regions. Next, the tiles evaluate the attested resources of each other. The goal of this evaluation is to compare these resources and decide which one is trusted compared to the other. To aid this decision, we propose extending the resources

```
x = (h1 & 1 == 1) ? h1 : 3;
y = (h2 & 1 == 1) ? h2 : 3;
s = (x == y)
o = s ? x : 3
p = h2
```

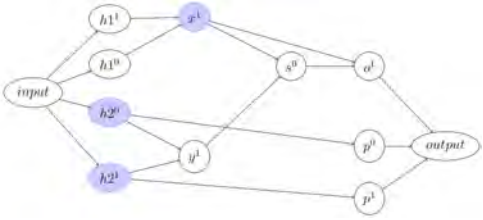


Figure 4.32: An example program and its bit dependence graph. A minimal cut is marked in blue. Since the cut has size 3, the program leaks at most 3 bits.

with a metadata, defining them specifically. This definition involves a version information, for the assumption that a recent version of a resource is expected to be trusted compared to its older versions, as it should address the known vulnerabilities. After the decision is made using the comparison with the metadata, the attestation responses take place. These responses aim at updating the untrusted resources of a tile with the trusted ones found in the neighbouring tiles. Furthermore, the responses aim at preventing even the Denial-of-Service (DoS) attacks by recovering a tile even when it is exploited.

For executing the above mentioned attestation, evaluation, and response mechanisms reliably, we designed a new trusted computing base as a hardware Root-of-Trust (RoT) attached to each tile. The RoT modules are supported with a link to the network adapter for direct communication with each other, and a cryptography module. These prevent the RoT from relying on the system software, hence avoids any disruption of its operation by potentially exploited software. For minimising the cost of equipping each tile with the proposed RoT module, we designed a compact [TV19a] elliptic curve implementation.

C5

Considering the case that we have a confidential app we want to secure, additional hardware solutions are beneficial. Relying on a Dynamic Root of Trust, Secure Encrypted Virtualization (SEV) guarantees the confidentiality of virtual machines (VM). The VM is placed in the host memory and the SEV firmware attests the integrity of the initial VM state. We extended this idea by introducing SEVGuard [PNG19], which allows to secure one application instead of VMs. SEVGuard is a minimal virtual execution environment that protects the confidentiality of applications based on AMD's Secure Encrypted Virtualization. Therefore, we move the App into a minimal VM that leverages plain KVM API of the Linux kernel. Using the SEV firmware, the application is injected in the VM and then the VM is encrypted. The role of SevGuard is to

encrypt the guest's initial memory and instructs the hypervisor to manage and control the VM. To keep the VM minimal without consuming a high amount of memory, system calls and libraries are not included in the VM. By implementing an advanced trapping mechanism, these functionalities can be used from host.

Summary and Outlook

The invasive computing paradigm offers applications the possibility of dynamically spreading their computation in a multicore/multiprocessor system in a resource-aware way. If applications are assumed to act maliciously, many security problems arise. Our solutions for isolating applications as well as for guaranteeing software and control flow integrity are able to enforce security on invasive computing architectures.

Testing for the existence of potential side channels in the distributed monitoring system will be part of the upcoming research. We will focus on the temperature sensor from the monitoring system for the first investigations. Furthermore, we plan to extend intra-tile security to inter-tile for integrity and confidentiality and make further progress on integrating IFC and CFA.

Publications

[Mae19] P. Maene. "Lightweight Roots of Trust for Modern Systems-on-Chip". Dissertation. Faculty of Engineering Science, KU Leuven, Belgium, Oct. 2019.

[PNG19] R. Palutke, A. Neubaum, and J. Götzfried. "SEVGuard: Protecting User Mode Applications using Secure Encrypted Virtualization". In: *SecureComm 2019 Proceedings* (Orlando). Springer, Oct. 24, 2019.

[TV19a] F. Turan and I. Verbauwhede. "Compact and Flexible FPGA Implementation of Ed25519 and X25519". In: *ACM Transactions on Embedded Computing Systems (TECS)* 18.3 (2019), p. 24.

[TV19b] F. Turan and I. Verbauwhede. "Propagating Trusted Execution through Mutual Attestation". In: *4th Workshop on System Software for Trusted Execution (SysTEX)*. Huntsville, Ontario, Canada: ACM, 2019.

D1: Invasive Software–Hardware Architectures for Robotics

Tamim Asfour, Walter Stechele

Dirk Gabriel, Fabian Paus

Project D1 investigates the benefits and limitations of invasive computing in robotic applications, which combine methods and algorithms from the area of computer vision and motion generation with concurrent processes and timely varying resource demands.

The research activities in Project D1 focus on building an invasive memory system—as integral part of a robot control architecture—which encodes prior world and task knowledge, sensorimotor experience and application resource requirements associated with robot actions and scene context. Such memory system will allow 1) encoding, storing and retrieving information about resource requirements, 2) learning prediction models of robot actions and their consequences (world state) as well as their computational resources, and 3) implementing speculative resource management system based on resource-aware prediction models learned from experience. The approach will be validated in the context of a comprehensive *perception and affordance pipeline* which combines a wide variety of robot vision algorithms for scene understanding and action selection.

Furthermore, we will address the challenges posed by the legacy software of existing robotic systems as currently the code of used algorithms would require complete re-engineering in order to benefit from invasive computing. We will investigate methods for automatic porting of the non-invasive code of the perception and affordance to invasive multicore platforms.

Resource-Aware Parameter Tuning for Real-Time Applications

In the last years we adapted robotic applications like, e. g. the object detection pipeline to run on the invasive platform. A major challenge was to extract the resource requirements for different workload scenar-

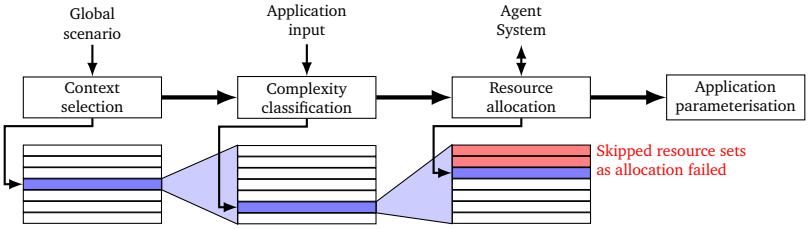


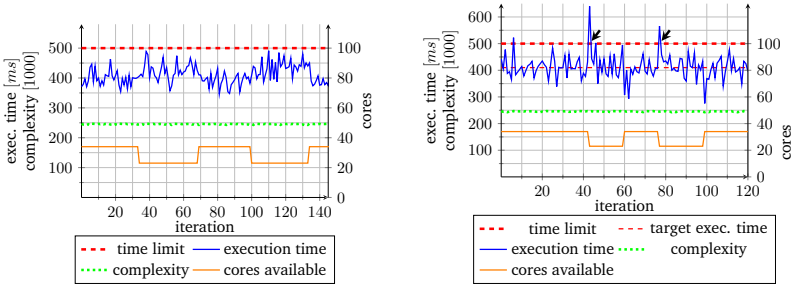
Figure 4.33: Workflow of the Resource-Aware Parameter Tuning.

ios and use the resource information provided by the operating system within the application.

In 2019, we extended this approach to a more generic method which can be applied to any application whose degree of parallelism can be parameterised and is executed in multiple iterations. The Resource-Aware Parameter Tuning (RAPT) [GSW19] relies on a set of operating points optimised during the Design Space Exploration (DSE). The operating points combine the information about application specific parameters, the set of allocated resources, the expected execution time, and the quality of the calculated result.

At run time a multi-staged lookup mechanism as shown in Figure 4.33 selects one suitable operating point for each iteration. The operating points are therefore stored in an ordered lookup tree. The first stage selects a subtree based on the context which specifies the time and power constraints. The second stage classifies the complexity of the current problem based on the application's input and steps one layer deeper into the tree. The third stage requests the resources from the system by interacting with the agent system. As it might happen that the resource demand can not be complied by the system, different configurations are stored in descending order of quality-level. RAPT steps through this list until it reaches an entry which resource demand is successfully mapped by the agent system. The last stage applies the annotated parameters to the application.

Figure 4.34 depicts the execution time of the object detection application running on an hardware platform consisting of 4 Intel®Xeon®E7-4830 CPUs once with RAPT being used and once with an feedback mechanism. With RAPT the execution time is only exceeded when the input complexity unpredictably changed whereas the feedback mechanism fails even with constant complexity class. The overhead introduced by RAPT remains below $25\,\mu s$ or 0.1% of the whole execution time and is therefor negligible.



(a) Resource-Aware Parameter Tuning guarantees timing constraint by adapting application specific parameters.

(b) Timing constraint violated by feedback mechanism.

Figure 4.34: Comparison of the Resource-Aware Parameter Tuning with a feedback mechanism.

This current workflow of RAPT still relies on modifications of the original application code. Therefore we are working on an implementation of the RAPT library compatible to the pthread-interface. It will allow the resource-aware execution of applications based on pthreads without any modifications. The major challenges are the reliable identification of different thread blocks and an automated approach for the complexity classification.

Resource-Aware Object Classification and Segmentation

We integrate cameras, robot vision and control algorithms in prosthetic hands to support semi-autonomous grasping. On these devices, power consumption determines the operating time in which the device is useful for the user. Usually, prosthetic hands use myoelectric control which relies on electromyographic (EMG) signals captured by two surface electrodes attached to the human body. Controlling the hand by the user requires long training and depends heavily on the robustness of the EMG signals. We developed a visual perception system ([HMA19]) to extract scene information for semi-autonomous hand-control that allows minimising required command complexity and leads to more intuitive and effortless control. We present methods that are optimised towards minimal resource demand to derive scene information from a camera inside the hand (see Figure 4.35). In particular, we show object classification and semantic segmentation of image data realised by convolutional neural networks (CNNs). We present a system architecture (see Figure 4.36), that takes user feedback into account and thereby

D1

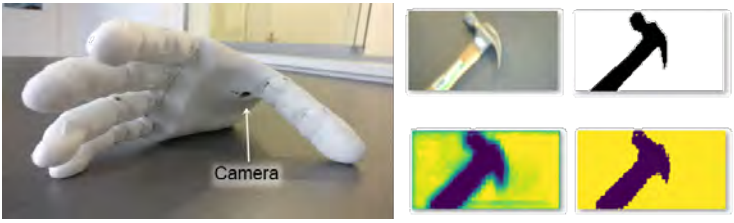


Figure 4.35: Left: In-hand camera in the palm of the *KIT Prosthetic Hand*. Right: Camera image, ground truth, predicted mask (8 bit) and binary mask.

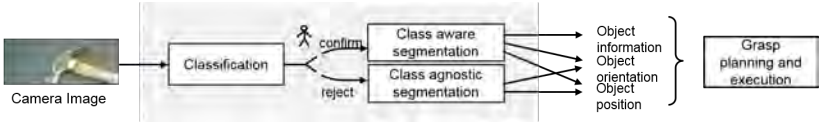


Figure 4.36: Overview of the pipeline used for semi-autonomous grasping including object classification and segmentation.

improves results. In addition, we present an evolutionary algorithm to optimise CNN architecture regarding accuracy and hardware resource demand. Our evaluation shows classification accuracy of 96.5% and segmentation accuracy of up to 89.5% on an in-hand ArmCortex-H7 microcontroller running at only 400 MHz.

Classification is realised by a deep convolutional network optimised towards inference on the embedded processor. For our application of object classification two aspects are relevant: A set of known objects must be recognised with high accuracy within given real-time constraints. We set a maximum of $\approx 150\ ms$ as an acceptable value for recognition.

To achieve real-time network inference, the classification network architecture is synthesised by using an evolutionary algorithm. The algorithm uses multiple evolution steps in which the algorithms evaluated the fitness of all networks, breeds offspring networks with crossover characteristics from two parent networks of high fitness and randomly mutates segment parameters. Since real-time constraints and given hardware resources allow a maximum number of operations at which the highest accuracy can be expected, we define the fitness function to target a given number of operations. Since the convolution layers are responsible for most of the resulting operations, only the operations resulting from convolution are regarded. The number of multiply-

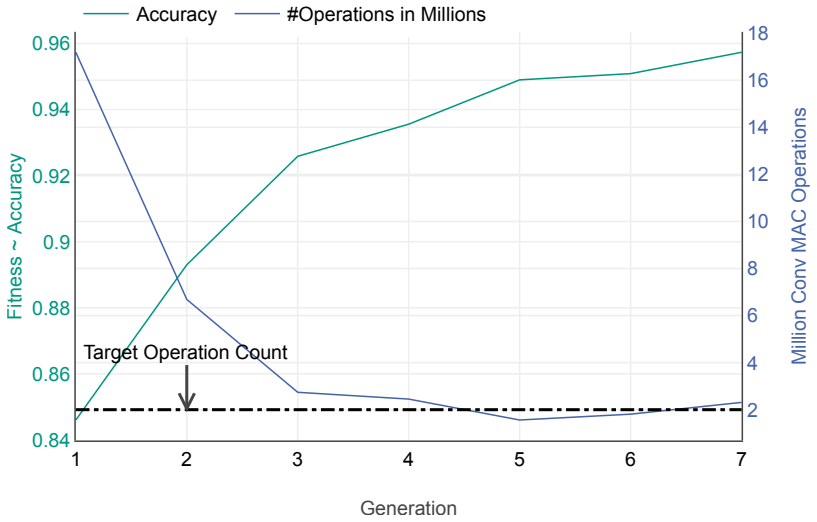


Figure 4.37: Average accuracy and MAC operations in millions per generation. The evolutionary algorithm successfully generates networks that have increased accuracy and lower operation count. The convergence of operations to 2 is a result of the chosen fitness function.

accumulate (MAC) operations per convolution layer can be calculated as

$$\beta_{\text{conv}} = \frac{I_H \times I_W \times I_C \times K_H \times K_W \times O_C}{S^2} \tag{4.3}$$

with input I, kernel K, output O as well as height, width and channels H, W, C. S is the filter stride. We derive the fitness of a network combining number of operations and accuracy by designing a fitness function using a generalised logistic function, in detail

$$\mathcal{F}(\alpha_{\text{final}}, \beta_{\text{conv}}) = \alpha_{\text{final}} + (1 + e^{\beta_{\text{conv}} - (1+\nu)})^{-\frac{1}{2+\nu}} \tag{4.4}$$

Results of the evolutionary algorithm used for the design of the optimised classification network architecture, as average accuracy and multiply-accumulate (MAC) operations per generation, are depicted in Figure 4.37. The results show, that evolution as expected decreases the operation count while increasing accuracy. The operation count, according to the target count defined in equation 4.4, converges towards the target operation count. Since accuracy and amount of operations improved evenly, the fitness function does not overrate either one and is therefore beneficial.

Publications

[GSW19] D. Gabriel, W. Stechele, and S. Wildermann. “Resource-Aware Parameter Tuning for Real-Time Applications”. In: *Architecture of Computing Systems – ARCS 2019*. Ed. by M. Schoeberl, C. Hochberger, S. Uhrig, J. Brehm, and T. Pionteck. Springer International Publishing, 2019, pp. 45–55. DOI: 10.1007/978-3-030-18656-2_4.

[HMA19] F. Hundhausen, D. Megerle, and T. Asfour. “Resource-Aware Object Classification and Segmentation for Semi-Autonomous Grasping with Prosthetic Hands”. In: *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*. 2019.

D3: Invasive Computing and HPC

Michael Bader, Hans-Joachim Bungartz, Michael Gerndt

Jophin John, Santiago Narváez Rivas

The overall goal of Project D3 in Phase III is to show that invasive computing can make a difference in High Performance Computing (HPC)—not only for improving performance and efficiency for individual applications, but also for overcoming some compelling challenges from operating large systems at supercomputing centres.

Invasion for HPC Overview

The biggest concerns of the current supercomputing centres are

- high energy consumption and lack of reproducibility,
- increasing failure rate due to growing core numbers which in turn aggravate the energy consumption problem,
- increasing complexity for efficiently handling resources due to the increase heterogeneity of the system.

Invasive computing provides remedies for these problems. Building upon the Elastic MPI (*i*MPI) infrastructure we developed in Phase II, we investigate power corridor enforcement using the mechanisms of invasive resource management. This can help to increase the predictability of the system's energy consumption. To handle hardware failure, we are developing a resource-aware checkpointing infrastructure (*i*Check) for fast adaptive data recovery. Additionally, we will investigate the invasion of memory layers with guidance by the applications. This will allow for improved execution despite the increased complexity in resources. On the application side, all these tasks require extreme-scale and realistic simulation codes [MoH19]. We are invasifying a set of selected classical HPC applications, originally not designed for resource elasticity. We will evaluate how well suited is the invasive infrastructure developed on Phase II to be used with them. Furthermore, we have

integrated data analytics framework Apache Spark with the invasive infrastructure for efficient handling of idle resources [CCG19].

Development of Invasive Infrastructure for HPC

Scaling Invasive Infrastructure: One of the main objectives of Project D3 in Phase III is to evaluate how well suited the invasive infrastructure developed in Phase II is for concrete scientific applications. To do so, a considerable amount of effort has been devoted to enable our invasive Resource Manager (*i*RM) to handle hundreds of nodes. This included a one week sprint code carried out on SuperMUC phase 2. The code was also adapted to run on SuperMUC-NG, the newest supercomputer acquired by the Leibniz Rechenzentrum (LRZ) at TUM.

Power Corridor Management: Invasive computing can be used to enforce the power corridor of a system [JNG19; Nar18]. The idea is that the power consumption should be kept in a certain range via a combination of expansions and reductions of more and less power consuming applications. Taking this idea into practice required extensive modifications to both application model (EPOP) and resource management (*i*RM), whose new interactions are depicted in Fig. 4.38. A heuristic and

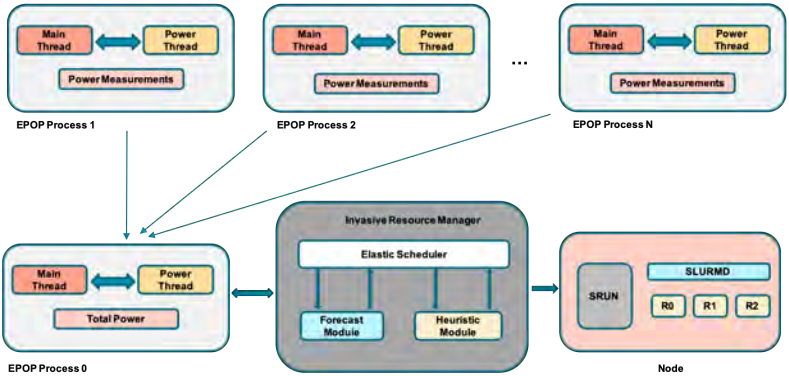


Figure 4.38: Power corridor management infrastructure.

forecast module were added to *i*RM. The latter allows the system to have a proactive approach, predicting power corridor violations before they may be happen. To do so several time series analysis techniques, such as the AutoRegressive Integrated Moving Average (ARIMA), Seasonal ARIMA with exogenous regressors (SARIMAX) and the Holt-Winters

method are used. When the infrastructure predicts that the system might go out of the power corridor, the heuristic module calculates a new resource distribution that prevents a violation.

***i*Check—Fault Tolerance in Invasive HPC:** As the peak system performance breaks the exascale barrier in HPC, the expected mean time between failures is in minutes. By utilising the application dynamism and the scheduler support, the applications can be restarted immediately with the available resources or continue the application execution with the smaller number of resources during node failures.

Our proposed system adds fault tolerance and failure recovery by exploiting this dynamism. It will be tightly coupled with *i*RM and will be able to expand and reduce the checkpointing resources according to application requirements and resource availability. It will be a multi-level application-level checkpointing system, where the checkpoints (specified application data) will be stored in compute nodes as well as in a parallel file system.

The *i*Check system has three components with three different objectives and views (see Fig. 4.39). It has one controller node (which is also a storage node) and one or more storage nodes. Storage nodes are provided to *i*Check by *i*RM. The *i*Check controller negotiates with *i*RM for more storage nodes whenever necessary. *i*RM can take back as well as add storage nodes to the system thereby making the checkpointing system itself invasive. Fig. 4.39 showcases the preliminary architecture of *i*Check System. The main components of *i*Check are:

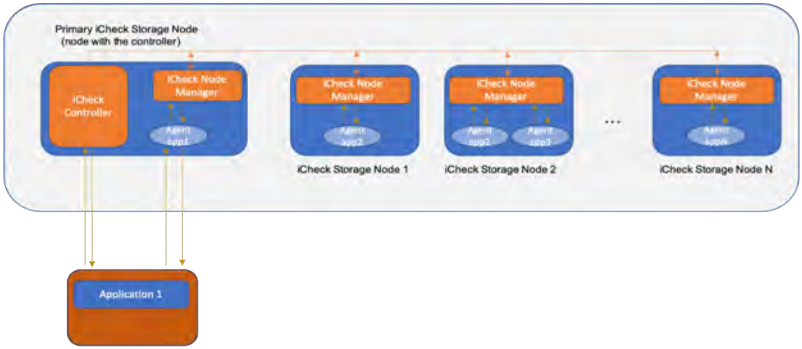


Figure 4.39: *i*Check architecture design.

controller: has a global view and is the main component of the infrastructure. Decisions regarding the checkpoints like storing the

checkpoints in parallel file system, creation of initial checkpoint, redistribution of checkpoint data, deletion of checkpoints etc. are taken by the controller.

node manager: is run on top of *i*Check storage nodes. It has the node-level view of the system. Node-level information like number of applications checkpointed, memory available, health of the node, etc. are periodically passed to the controller or passed upon request.

agent: is associated with an application. It is responsible for checkpoint management of an application.

Invasive HPC Applications

Invasive Molecular Dynamics Simulation: The molecular dynamics simulation code *ls1 mardyn* is an MPI application that focuses on the simulation of process engineering scenarios, such as the nucleation processes in large-scale chemical reactors. The invasification of the code has started. The current focus is on determining scenarios in which invasive computing could be especially useful. The following scenarios are being considered:

1. Full-invasive approach. This scenario relates to the mitigation of load imbalances in the simulation. For example, in Fig. 4.40 an heterogeneous particle distribution is depicted, with a clearly visible load imbalance. Nucleation, a process which happens in the context of condensation (i. e. when droplets are formed in an over-saturated gas phase), can cause this imbalances. *ls1 mardyn* has its own built-in functionality to deal with these cases, but invasive computing could become an alternative (see Fig. 4.40).

2. A hybrid approach. The idea here is to find the optimal amount of processors for a certain problem size. To do so, expansion and reduction operations would be used at the beginning of the simulation; the adaptations will end once the efficiency of the program is between a certain margin. Afterwards the native load imbalance functionality included in *ls1 mardyn* would be used.

Invasive Earthquake Simulation: The earthquake simulation code *SeisSol* is a large-scale application based on hybrid MPI+OpenMP parallelism. With added resource adaptivity, a large improvement in resource efficiency can be expected in rupture simulations, where most of the

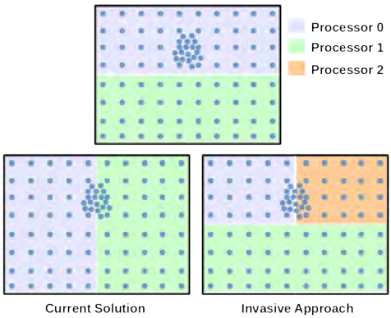


Figure 4.40: Load imbalance solution *ls1 mardyn* vs Invasive approach.

simulation time is characterised by the dynamics of the rupture process. We plan to adopt the lazy activation approach explored in Project A4 and activate parallel partitions only when seismic waves reach the partition. The invasification of *SeisSol* will begin once the one of *ls1 mardyn* is finished.

Outlook

In addition to completing the above mentioned tasks, our next steps include:

- **Hybrid Power Corridor Management:** One of the shortcomings of the invasive approach to power corridor management is that frequent resource redistributions can be expensive. Our ongoing work, a hybrid system which can use DVFS along with invasive computing to manage power requirements, addresses this shortcoming. If a power corridor violation is predicted then the scheduler selects the technique which is least expensive. An infrastructure for this hybrid system is implemented on *i*RM and EPOP. The power model of this system will be based on our work [CG19].

- **Power-aware Batch Scheduling:** Our present power corridor infrastructure manipulates the running applications to bring back the system into the power corridor. Having a power-aware batch scheduler complements this by scheduling a job based on the power requirements. The batch scheduler can make sure that new applications won't break the corridor thereby reducing the need for resource redistribution.

- ***i*Check API:** A preliminary architecture of *i*Check system has been decided and the respective API design is in progress. We will prototype the *i*Check system with a basic API and necessary design changes will be made as the development progress. Also, an invasive remote direct memory access (iRDMA) protocol is under development which will be used for checkpoint transfer and data redistribution in *i*Check.

- **Evaluation and Testing with Invasive Applications:** Performance analysis will be conducted on the invasive *SeisSol* and *ls1 mardyn* codes to evaluate the impact of resource adaptivity on the application-level. Furthermore, these codes will be used for testing the power corridor management and *i*Check's fault-tolerant functionalities. They will eventually adopt the new API provided by *i*Check for automatic data transfer during resource adaptation.

- **Investigation of memory invasion:** A newly acquired server will allow us to test OctoPOS and develop applications for it. This server has Optane DC persistent memory DIMMs, a technology from Intel that lays in between RAM and storage solutions. It can be configured as a volatile memory, as a faster storage alternative to SSD or both. This provides an opportunity to investigate, in collaboration with Project C1, memory invasion. For example, applications could be allocated memory either on RAM or on Optane DIMM depending on its performance.

Publications

[CCG19] J. A. Chacko, I. A. Comprés Ureña, and M. Gerndt. "Integration of Apache Spark with Invasive Resource Manager". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation*. Best Paper Award. 2019.

[CG19] M. Chadha and M. Gerndt. "Modelling DVFS and UFS for Region-Based Energy Aware Tuning of HPC Applications". In: *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2019.

[MoH19] A. Mo-Hellenbrand. "Resource-Aware and Elastic Parallel Software Development for Distributed-Memory HPC Systems". Dissertation. Munich: Technische Universität München, 2019. URL: `http://mediatum.ub.tum.de/?id=1471007`.

[JNG19] J. John, S. Narvaez R, and M. Gerndt. “Invasive Computing for Power Corridor Management”. In: *ParCo 2019: International Conference on Parallel Computing*. 2019.

[Nar18] S. Narvaez. “Power model for resource-elastic applications”. Master Thesis. Munich: Technische Universität München, 2018. URL: http://mediatum.ub.tum.de?id=1475095.

Z: Central Services

Jürgen Teich

Ina Derr, Frank Hannig, Jürgen Kleinöder, Stefanie Kugler,
Sandra Mattauch

The central activities and services of the CRC/Transregio 89 are coordinated and organised by Project Z. These activities can be divided into:

The first part, administrative support, the organisation of meetings (i. e. internal project meetings and workshops, Doctoral Researcher Retreats (DRR)) and assistance for visits of guest researchers and researchers travelling abroad. Project Z also provides technical support and a multitude of measures improving infrastructure and supporting data management tools for communication and collaboration (subversion repository, mailing lists, Wiki). We continue to support our researchers in facilitating equal career opportunities as well as balancing family and scientific career (e. g. cuby, kidsboxes or babysitting service during our annual meetings). In order to promote our doctoral researchers, we also organise training courses. Last but not least, the support of all management activities, financial administration and bookkeeping belong to our central service unit.

The second part, is public relations. This comprises establishing contacts with important research sites as well as the inception of an international Industrial and Scientific Board. Scientific ideas and results of the CRC/Transregio 89 are discussed regularly with guest researchers at the “InvasIC Seminar” as well as during various workshops and conferences. Among others, our website www.invasic.de, promotion material, press releases and media relations are also provided by Project Z. Eventually, we also support and organise the publishing process of central publications like the “InvasIC Annual Report”. The CRC/Transregio’s activities of 2019 are summarised in Part III of this report.

Z2: Validation and Demonstrator

Jürgen Becker, Frank Hannig, Thomas Wild

Nidhi Anantharajaiah, Marcel Brand, Joachim Falk, Fabian Lesniak, Leonard Masing, Sven Rheindt, Akshay Srivatsa

The major contribution of Project Z2 is to provide a joint environment for validating the principles of invasive computing. The contributions of the different projects across all project areas are integrated into one platform to demonstrate the advantages of invasive computing such as improved efficiency, resource utilisation, speedup, and *-predictability[12]. The currently used ProDesign proFPGA systems contain four Virtex-7 2000T FPGAs, allowing for large designs to be prototyped. This prototype has allowed us to implement tiled invasive hardware architectures containing up to 16 multicore (RISC) and manycore (TCPA) tiles, connected via the invasive NoC (*i*NoC), with room for extension and flexibility. To build a working platform, Project Z2 integrated peripherals like the transactor, SSRAM and DDR3 memory controllers into the common demonstrator and established the appropriate tool support for the integration of all the invasive components.

In the third funding phase, Project Z2 is continuing to support all projects, particularly in their work on Run-time Requirement Monitoring (RRM) and Run-time Requirement Enforcement (RRE) of non-functional properties as well as run-time verification of properties and constraints. To support the evaluation of RRM and RRE mechanisms such as the concept of leads and aides, Project Z2 will provide probes within invasive architecture prototypes that allow assessing metrics to be optimised or enforced and visualise them to an external observer.

[12] J. Teich et al. “Language and Compilation of Parallel Programs for *-Predictable MPSoC Execution using Invasive Computing”. In: *Proceedings of the 10th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. Lyon, France, Sept. 21–23, 2016, pp. 313–320. DOI: 10.1109/MCSoC.2016.30.

Continuous Integration System

At the review meeting in January 2018, two major demonstrators that showcase the advantages of invasive computing were presented. This multi-FPGA prototyping environment enabled other projects of the CRC to integrate and demonstrate the entire invasive hardware/software technology stack.

Based on this success, many new components have been developed, researched and published since then [Rhe+19b; Rhe+19a]. Project Z2 assisted their integration and further extended the memory as well as monitoring subsystem of the demonstrator platform.

In order to maintain the integrity of the system and further ease and automate the seamless integration of new components and features into our prototype, Project Z2, in close cooperation with Project C1, is establishing a continuous integration (CI) and testing environment. Continuous integration is the practice of continuously integrating changes made to a project and testing them accordingly at least on a daily basis or more frequently. An example CI flow is depicted in Fig. 4.41. As this is common practice for any software development project, we are on the way to adapt our hardware design flow to it as well. When new features are integrated, the design will be synthesised in an automated fashion and an extensive test suite consisting of microbenchmarks as well as real applications benchmarks will continuously be executed.

As important steps towards this goal, Project Z2 implemented a configurable and already semi-automated script-based synthesis flow. Project Z2 and Project C1 further provided an environment that allows performing holistic tests of project specific components and their interplay with the whole demonstrator in an automated fashion. Automated tracing of the outputs for error messages helps to categorise them and giving first hints where to focus the further testing on.

Figure 4.41: Continuous integration process.

Automating the build, test and deploy processes can alleviate much of the headaches and problems commonly happening on projects. Having a reliable method of integrating smaller changes more frequently ensures that errors can be identified and resolved easier and earlier.

The benefits of continuous integration are especially important in such a big, inter-disciplinary and collaborative project. The reduced integration risk, higher code quality and increased confidence in the proper functionality of new or modified components makes the collaboration of hardware, system-software and application researchers more comfortable and efficient. As the turnaround cycle for hardware compared to software development is significantly higher, CI is all the more important and helpful to avoid unnecessary manual testing.

Non-intrusive Monitoring

The demonstration platform provides concurrent execution of up to 80 processors allowing debugger access to each individual processor. While this debugging method enables fine-grained control and insight into single processors and their memory, it does not show the full picture. Run-time behaviour is determined by many more components, some of which are not observable in real time using the debugger. This includes effects like memory performance (e. g. cache misses, DRAM refreshing), NoC latency and load on the tile-local AHB bus. Any mechanism spanning over multiple cores like scheduling or resource allocation is hard to observe from the outside.

Several sensor values are available in the current hardware prototype, but they are distributed over different hardware components. While some of these sensors can be accessed using the tile-local bus, it is not feasible to collect that data in software. That is because access to the sensors will influence the overall system performance and lead to imprecise measurement values. This effect intensifies with increasing the number of observed sensors. To be able to monitor the system on a very detailed level and not influencing it at the same time, a non-intrusive monitoring mechanism is required.

Additionally, access to internal sensors as a memory mapped resource would be beneficial for some applications. By providing a standardised interface for any kind of sensors, applications can easily access monitoring data in a uniform way. For example, there is a plan to integrate the power and temperature monitoring system developed by Project B4 to that generic interface.

These two usage scenarios require the following basic functions:

- Collecting run-time monitoring data from several hardware resources (CPUs, Caches, AHB, Network Adapter, NoC Routers)
- Continuously streaming selected real-time monitoring values to an external capture device using Ethernet
- Providing memory-mapped access to sensor data values on the tile-internal bus

Project Z2 is implementing the task of data collection and the local interface in monitoring cores inside of each tile. They allow the monitoring system to scale with the number of tiles in the design, while still being accessible from inside the tile to provide access to local sensor data to software. To be able to gather monitoring data without internal bus access or similar intrusive mechanisms, a monitoring core has direct connection to all monitored components of its respective tile.

To reach the primary goal of monitoring the system non-intrusively, sensor data must be brought from the individual monitoring cores to the outside of the system without using existing architecture components like the *i*NoC. Therefore an additional, lightweight and uni-directional monitoring network is introduced. It connects all monitoring cores with the network streaming core and handles consolidation of monitoring data as well as transportation to the network interface (Fig. 4.42).

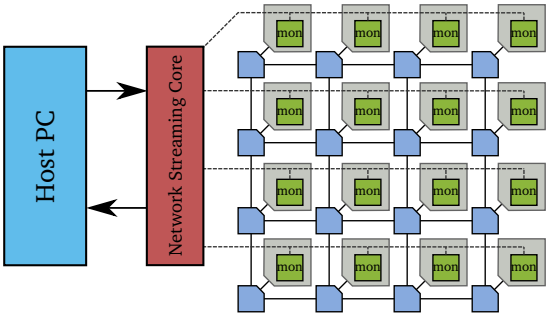


Figure 4.42: NoC with monitoring cores inside each tile.

This design allows for great insight into the system at run time while making sure not to influence the execution in any way. By simplifying debugging, optimisation and design space exploration, it will improve the evaluation capabilities of the prototyping system. Ongoing work includes collecting all sources of necessary sensor data, defining a proper sensor data interface and working on the streaming subsystem.

Validation and Exploration based on Simulation and Hybrid Prototyping

The system-level simulator InvadeSIM[13,14], which was developed by the previous Project C2, has been one significant and operative instrument in the previous funding phases. Particularly in project areas A and D, it has been used as a testbed for invasive programming concepts as well as a basis for architecture exploration and application mapping. The research achievements are subsumed in the textbook "*Modeling and Simulation of Invasive Applications and Architectures*" [RHT19] published by Springer in May 2019.

In order to keep up with the latest advances of recent operating systems (Ubuntu Linux distribution) and compiler versions (GNU Compiler Collection, GCC) as well as modern Intel processors (e. g. Haswell, Skylake, and Coffee Lake architectures), Project Z2 released a new version of simulation infrastructure InvadeSIM.

One field of InvadeSIM's application in the past year has been the testing of the first implementations of run-time requirement enforcement (RRE) techniques of program execution. In particular, image processing applications have been employed to study several RRE variants, such as central or distributed enforcement of soft as well as hard real-time requirements. For further details, we refer to Project A1 in this report.

Moreover, we have been utilising hybrid prototyping techniques that combine a virtual platform with FPGA-based prototyping, and are thus a powerful tool for design validation and exploration, e. g. of NoC features in manycore architectures [MLB19]. We refer to Project B5 in this report for additional details.

Public Dissemination

Project Z2 vividly contributed to the dissemination of the CRC by presenting the accomplished innovations and research achievements. Particular highlights of the last year were a booth at the exhibition of the Conference on Design, Automation and Test in Europe (DATE) in Florence in March 2019 as well as an appearance at the Long Night of Sciences ("Lange Nacht der Wissenschaften" in Erlangen in October 2019, see

[13] S. Roloff, D. Schafhauser, F. Hannig, and J. Teich. "Execution-driven Parallel Simulation of PGAS Applications on Heterogeneous Tiled Architectures". In: *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (San Francisco, CA, USA). ACM, June 7–11, 2015, 44:1–44:6. DOI: 10.1145/2744769.2744840.

[14] S. Roloff, F. Hannig, and J. Teich. "High Performance Network-on-Chip Simulation by Interval-based Timing Predictions". In: *Proceedings of the 15th IEEE/ACM Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia)* (Seoul, Republic of Korea). ACM, Oct. 15–20, 2017, pp. 2–11. DOI: 10.1145/3139315.3139320.

Fig. 4.43. For more information and photographs see Chapter 8). On both occasions, our joint demonstrator—the invasive inverted pendulum—was showcased, however, with a different focus. While in the case of the Long Night of Sciences, the visitors of various age and educational background got fundamental insights into tomorrow's manycore architectures and the concepts of invasive computing, in the case of the DATE 2019 conference, the visitors were mainly experts in the field.

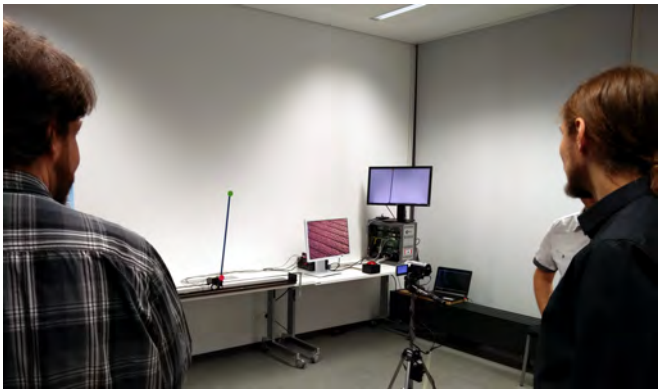


Figure 4.43: *-Predictable inverted pendulum demonstrator at the Long Night of Sciences 2019.

Publications

[MLB19] L. Masing, F. Lesniak, and J. Becker. "Hybrid Prototyping for Manycore Design and Validation". In: *Applied Reconfigurable Computing*. Springer International Publishing, 2019, pp. 319–333.

[Rhe+19a] S. Rheindt, A. Fried, O. Lenke, L. Nolte, T. Wild, and A. Herkersdorf. "NEMESYS: Near-Memory Graph Copy Enhanced System-Software". In: *MEMSYS 19: The International Symposium on Memory Systems*. Washington DC, 2019.

[Rhe+19b] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf. "SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures". In: *Proceedings of the 19th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2019.

[RHT19] S. Roloff, F. Hannig, and J. Teich. *Modeling and Simulation of Invasive Applications and Architectures*. Springer, May 2019. 168 pp. DOI: 10.1007/978-981-13-8387-8.

5 Working Groups

WG1: Run-Time Requirement Monitoring and Enforcement

Coordinators: Felix C. Freiling, Daniel Müller-Gritschneder, Timo Hönig

The focus of this working group lies on the joint investigation of run-time measures for monitoring and enforcement of requirements on non-functional properties such as timing, power, temperature, reliability and security. The enforcement of such requirements is a central research challenge of the CRC with high practical relevance. In the domain of real-time systems, strict guarantees on the execution time of tasks are required. Designers are often forced to add overly pessimistic safety margins to assure these guarantees. These margins can be relaxed if the embedded system supports the monitoring and enforcement of these properties during run-time. Other examples, for which run-time monitoring and enforcement will aid in making better use of the system resources, include systems with reliability or security requirements. Additionally, also thermal management strategies, which gain increasing attention as we move in the age of dark silicon, require accurate monitoring data from the chip. Next to embedded computing, also applications in the high-performance computing (HPC) domain can profit from run-time requirement enforcement. HPC centres often need to satisfy certain power corridors, which were negotiated with the power grid providers. The enforcement of such corridors requires constant monitoring and run-time adjustments.

Activities

The main activities of the working groups centered around two events:

- A working group meeting and tutorial on “hyperproperties” by Christopher Hahn (Universität des Saarlandes, Prof. Finkbeiner) on July 25–26, 2019 in Munich.
- A working group meeting on November 15, 2019 in Karlsruhe.

The topic of hyperproperties connected to the first working group meeting in 2018 on run-time verification using linear temporal logic (LTL). In LTL, system properties are modelled as sets of linear traces (sequences of system states) that can cover functional properties like partial correctness and mutual exclusion. It is, however, well known that certain security properties cannot be formalised in that way. This is particularly relevant for information flow properties, where an observer can infer certain secret values by observing particular system traces. Hyperproperties are therefore properties of *pairs* of traces and can be used to model information flow and therefore also to enforce information flow control.

The first meeting of the working group in 2019 consisted of a tutorial on hyperproperties and the formalism of HyperLTL that uses LTL to express relations of traces. Originally, Prof. Bernd Finkbeiner of Saarland University was to give the tutorial together with his student Christopher Hahn, but unfortunately Prof. Finkbeiner was ill and so Christopher Hahn gave the tutorial on his own. It completed the picture of property specification formalisms that form the basis for the work in this working group. Further discussions regarded the integration of run-time monitoring and run-time enforcement into the architecture of invasive computing systems.

The second meeting in 2019 in Karlsruhe aimed at gaining more practical insights into run-time monitoring. Also, possible cooperations on monitoring and enforcement between the projects were discussed. A major topic of the discussion was on how to guarantee hard real-time constraints for several applications executing in a parallel fashion within a dynamic, invasive computing environment. This challenge was identified as a major task for the Working Group. Additionally, thermal and power monitoring and enforcement aspects were discussed. Finally, it was concluded that a workshop with Dr. Ziegenbein (Bosch Research) should be organised to discuss the concepts, which were developed during the workshop, and their practical application in industrial systems.

WG2: Memory Models, Architecture and Management

Coordinators: Lars Bauer, Andreas Herkersdorf, Wolfgang Schröder-Preikschat, Gregor Snelting

WG2 is the CRC/Transregio 89-internal forum to exchange and evolve all memory-related aspects affecting the invasive computing hardware / software architecture, run-time support system, programming models, as well as formal properties of the programming language. As can be seen from Table 5.1 and the list of publications at the end of this report, the working group is really "alive and kicking". Mitigating the effects of memory access interference, e. g. by improving data-to-task locality and adaptive memory protection, were two of the central topics focused upon by WG2 members in several collaborative activities during the past 12 months.

Table 5.1 lists the ongoing, completed, already published, and also planned activities of inter-disciplinary collaborations among different

Topic	Projects	Status
OctoPOS Pthread Support	B5, C1, D1	ongoing
Region-Based Cache Coherence on Demand	B5, C1, D1	[Sri+19]
Advanced Remote Atomic Operations	B5, C1, C3	[Rhe+18]
Virtual Shared Memory (VSM)	C1, C5	ongoing
Near-Memory Support for VSM	B5, C1	planned
Near-Memory Graph Cloning	B5, C3	[Rhe+19a]
Software-Defined Hardware Managed Queues	B5, C1	[Rhe+19b] [Mai+19]
Formal X10 Memory Model	A1	ongoing
Region-Based Cache Coherence Extension for X10	B5, C1, C3	ongoing
Design Time Characterisation of Memory Accesses	A4	planned
Remote Load-Store Optimisations	B5	completed
Physically Distributed Global Memory	B5, Z2	completed
Near-Memory BLAS	B5, D3	ongoing
Invasion of Memory	B5, C1	ongoing
Consistency Models Tutorial	C1	completed

Table 5.1: Activity status on working group collaborations.

CRC/Transregio 89 projects that we consider necessary to tackle the memory architecture and management challenges. These collaborations already yielded quite a number of joint publications at well-known international conferences during the last year, see e. g. [Rhe+19a; Rhe+19b].

The working group lives from the many bilateral exchanges and collaborations. Besides the (semi-)annual meetings, the working group had an additional meeting on July 1, 2019 at TU Munich. The next meeting is planned for spring 2020. An example for a new topic, which was brought up in one of the working group meetings, Near-Memory BLAS (basic linear algebraic subroutines), has been picked up by a master thesis student at TUM.

Finally, Prof. Herkersdorf also contributed a keynote titled "Tackling the MPSoC Data Locality Challenge with Regional Coherence and Near Memory Acceleration" [Her19] at NorCAS 2019 in Helsinki.

Publications

[Her19] A. Herkersdorf. "Tackling the MPSoC Data Locality Challenge with Regional Coherence and Near Memory Acceleration". Keynote talk, 2019 IEEE Nordic Circuits and Systems Conference (NorCAS). Oct. 29, 2019.

[Rhe+19a] S. Rheindt, A. Fried, O. Lenke, L. Nolte, T. Wild, and A. Herkersdorf. "NEMESYS: Near-Memory Graph Copy Enhanced System-Software". In: *MEMSYS 19: The International Symposium on Memory Systems*. Washington DC, 2019.

[Rhe+19b] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf. "SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures". In: *Proceedings of the 19th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2019.

WG3: Benchmarking and Evaluation

Coordinators: Michael Gerndt, Walter Stechele

The goal of WG3 is to identify results from the projects and potential use of these in demonstrators. Benchmarking invasive versus state-of-the-art computing seems a major challenge.

In 2019, WG3 organised two working group meetings, on January 1, 2019 in Munich with a focus on the planned results of the projects that may be used in a demonstrator or benchmarking activity, and on June 14, 2019 in Karlsruhe with a specific focus on robotic use cases.

During the Munich Meeting in January 2019, a discussion developed around the OS concepts of a squad and an aide. These OS concepts are the basis for the implementation of run-time enforcers. The squad is a team of application *i*-lets including one or more aide *i*-lets. These aide *i*-lets implement a single enforcer or multiple enforcers. The enforcers are application-level concepts that can influence the next invade/infect operation. It is important that these enforcers can access objects within application *i*-lets. They are triggered by the OS in case a violation of a quality of service requirement is detected. The concept of squad-based enforcers has been discussed further during the InvasIC annual meeting in February 2019.

The specific focus of the June 2019 Meeting in Karlsruhe was on two robotic use cases for invasive computing, the perception pipeline and the prosthetic hand.

The perception and affordance extraction pipeline is a sequence of algorithms on a robot, which uses RGB-D data as input to build primitive-based geometric scene representations and assign whole-body interaction possibilities, so-called affordances, such as push, lift, lean, support, etc. to the primitives. The proposal was to focus on a special part of the pipeline, namely Lidar Point Cloud clustering based on a RANSAC algorithm. This is planned as a demonstrator for the invasification of legacy code. Relevant topics include legacy application wrapping (D1), Design Space Exploration (A4), (Agent-based) resource management

(C1), and run-time & power enforcement (A1). The goal is to demonstrate more efficient resource usage and energy reduction compared to a non-invasive implementation, the trade-off between resource usage, energy, run-time, and perception quality enforcement.

The hand prosthesis includes a camera and an ARM microcontroller. It simplifies the control of the prosthesis for the human, by detecting objects via a convolutional neural network (CNN) and automatically grasping the object. A critical aspect is the energy consumption since the hand is battery powered. Currently, the only hardware control knob for energy saving is the processor frequency. Future extensions might include multicore processors and FPGA-based accelerators. Invasive methods might be based on the use of TCPA (Project B2) and *i*-Core (Project B1) for acceleration. Dynamic precision adaptation (FloatTCPAs) might be exploited to meet the 150 ms reaction time requirement via multiversioning or anytime precision support. Different CNNs can be seen as operating points. The goal is to trade classification accuracy for run-time and energy enforcement.

Beginning of 2020, WG3 organises the workshop on the computing continuum, as part of the HiPEAC conference in Bologna, January 20, 2020. The term "computing continuum" describes an infrastructure comprised of high performance computing centres connected to the servers of clouds, computers at the edge of the network, and embedded computing resources within distributed IoT (Internet-of-Things) devices. Currently, invasive computing is exploring resource invasion within embedded MPSoC and within HPC. The future challenge of the computing continuum might be related to resource invasion on the overall infrastructure over the boundaries between HPC, edge, and embedded MPSoC.

WG4: Power and Thermal Aspects

Coordinators: Jörg Henkel, Nicole Megow, Stefan Wildermann

A limiting factor for high performance has been, is, and will be power consumption. There are various aspects for this limit:

1. The power density represents a physical limit as the amount of power that can be dissipated at a certain chip area is limited by the maximum temperature a circuit can stand without the risk of accelerated circuit degradation or even immediate irreversible damage.
2. The energy efficiency determines how much computation (or communication) can be accomplished with a certain amount of energy. Especially in energy-limited embedded applications, it is the goal to make as much as possible use of a limited amount of energy.
3. Power and energy under real-time constraints: while reasons 1 and 2 are already hard to accomplish, the problem grows more complex when real-time comes into play. For example, if a real-time task needs to complete at a certain time, boosting might be a preferable means. That, however, will increase peak temperature and put the circuits under non-sustainable high stress. A multi-objective optimisation strategy may be needed. In general, a thorough investigation of the trade-offs is a primary goal.
4. Investigating how various scheduling and allocation algorithms match or can be adapted to invasive computer architectures in order to achieve a high efficiency.

InvasIC has various projects (among them Project A5, Project B2, Project B3, Project C1) that focus on one or more aspects of power and energy with respect to aspects 1, 2, 3 and 4. The goal of WG4 is to bring these various goals under one umbrella in order to:

- Coordinate these various aspects such that in various phases during execution on an invasive multicore architecture, the applied

power and energy means at different components (OS, architecture, application software, etc.) are targeted towards the same goals and contradictory control loops are avoided.

- Develop power and energy models that can be used by all projects that deal with the topic.
- Identify which aspects of power, energy and temperature analysis and modelling should be addressed at design time and which should be modelled as uncertainties when making scheduling and application mapping decisions (at run time). Of particular interest is to study how far dynamic resource reservation using invasive computing and novel ideas on run-time requirement enforcement and run-time verification can help to reduce such uncertainties.

From an organisational point of view, it is the goal to target two working group meetings per year. It is also the goal to organise a session at an international event to present and discuss the very goals of this WG4 with international experts working on similar topics.

Second Meeting – Machine Learning Workshop In our first meeting in 2018, we have identified that Machine Learning was considered in several projects. As a result of this meeting, we decided to plan a tutorial-style workshop on Machine Learning for 2019. Together with Project Z, we could win Prof. Dr. Alexander Waibel together with Dr. Thanh-Le Ha and Ngoc Quan Pham to give a two-day workshop.

It took place September 19–20, 2019 at the Technologiefabrik Karlsruhe. The considered topics were an introduction to machine learning and deep neural networks in particular, as well as overviews on feature generation and extraction, clustering techniques, and neural networks for classification and recurrent neural networks for time series analysis.

A second result of the meeting in 2018 was to establish collaborations between groups on the topics of thermal-aware application mapping, power and energy in HPC, ageing monitoring, power models, as well as how to model and benchmark our result. Since then, multiple cooperations are ongoing. One successful publication in 2019 stemming already from this working group is [Pou+19a].

Publications

[Pou+19a] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel. “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems”. In: *40th IEEE Real-Time Systems Symposium (RTSS)*. 2019.

III

Events and Activities

Summary

The central activities and services in InvasIC are coordinated and conducted by Project Z.

Figure 5.1: From left to right: Stefanie Kugler (Public Relations), Prof. Dr.-Ing. Jürgen Teich (Coordinator), Dr. Sandra Mattauch (Coordination and Public Relations), Dr.-Ing. Jürgen Kleinöder (Managing Director) and Ina Derr (Financial Services).

In the following, we summarise the major events Project Z organised like Internal Meetings (Section 6), Training Courses (Section 7) as well as further InvasIC Activities (Section 8) and Awards (Section 9). Last but not least, we present the current composition of the Industrial and Scientific Board in Section 10.

Figure 5.2: At the annual meeting in Kloster Irsee, February 2019.

6 Internal Meetings

Collaboration between the researchers of the three sites Karlsruhe, Munich, and Erlangen is essential for the success of the CRC/Transregio 89 – Invasive Computing. In 2019, researchers met at the following opportunities (list not being exhaustive):

Event	Date	
WG Evaluation and Benchmarking	Jan. 28, 2019, Munich	The goal of the meeting was to identify potential demonstration scenarios for invasive computing considering the enforcement of non-functional properties.
Semi-Annual Meeting	Feb. 18/19, 2019, Kloster Irsee	At the semi-annual meeting, the status quo of the projects and the working groups were summarised.
Doctoral Researcher Retreat	Feb. 26–28, 2019, Bad Boll	The doctoral researchers met in Bad Boll to discuss progress and further challenges of the third funding phase.
WG Evaluation and Benchmarking	June 14, 2019, Karlsruhe	Members of the working group met in Karlsruhe to identify methods for evaluation and to determine benchmarks.

Figure 6.1: Impressions from the semi-annual meeting.

Event	Date	
WG Memory Models, Architecture and Management	July 1, 2019, Munich	The agenda of the meeting was to discuss WG2 work items (especially those not started yet), to present “Virtual Shared Memory – Current State and Future Steps” and how to demonstrate merits of results.
WG Runtime Requirement Monitoring and Enforcement	July 26, 2019, Munich	The main part of the meeting was a tutorial on “Hyperproperties” by Bernd Finkbeiner and Christopher Hahn from Saarland University.
Annual Meeting 2019	Oct. 7/8, 2019, Dinkelsbühl	At the annual meeting all researchers met. In short talks all projects and working groups presented their progress.
Doctoral Researcher Retreat	Oct. 9–11, 2019, Dinkelsbühl	The 14th InvasIC DRR took place at the Designhotel Meiser in Dinkelsbühl in conjunction with the annual meeting.
WG Runtime Requirement Monitoring and Enforcement	Nov 15, 2019, Munich	Researchers from different groups used the meeting to collaborate across projects.

Figure 6.2: At the annual meeting in Dinkelsbühl 2019.

7 Training Courses

The following internal workshops and training courses were organised by Project Z to give the doctoral researchers of InvasIC the opportunity to strengthen their soft skills, train their key qualifications, and improve their knowledge on topics related to invasive computing. Doctoral researchers could choose their preferred workshop topics by a poll beforehand.

Workshop "Machine Learning"

Because of artificial intelligence (AI) and machine learning (ML) being hyped topics in computer science, which makes trainers for related topics hard to be found nowadays, we were delighted that Professor Dr. Alexander Waibel, Professor of Computer Science at the Karlsruhe Institute of Technology (KIT), Germany and at Carnegie Mellon University, Pittsburgh agreed to be our referee for this workshop. His research centres around Machine Learning algorithms that improve human-machine and human-human interaction. His early pioneering work on the Time-Delay Neural Network was the first "convolutional" neural network.

Prof. Waibel, accompanied by Dr. Thanh-Le Ha and Ngoc Quan Pham, brought some specific topics of machine learning closer to our researchers. This tutorial-styled workshop was initiated by WG4 and included: Feature Generation & Extraction, Clustering, Classification, Neural Networks, Time Series Analysis, and Deep Learning.

The workshop was completed by a homework assignment to deepen the understanding of machine learning.

Figure 7.1: Workshop on machine learning, September 19–20, 2019.

Workshop "My Project Management Skills"

The "Research Associations of the Friedrich-Alexander University Erlangen-Nürnberg for the Promotion of Gender Equality" (F^3G) are an association of various DFG projects at the FAU, with our CRC/Transregio being one of its founding members. In addition to better networking, the goal is the appropriate and directive use of DFG gender equality funds. Since 2019, the F^3G offers the possibility for doctoral researchers of DFG funded projects to participate in soft skill workshops at FAU. These workshops are open to all CRC/Transregio 89 doctoral researchers notwithstanding they work at FAU or not. Some of our doctoral researchers took advantage of this new opportunity and attended the course "My Project Management Skills" on November 23, 2019. The course was given by Dr. Silke Oehrlein-Karpi.

Organisation, coordination and effective leadership have become a must for growing in one's career, in Academia as well as in other sectors. Not only scientist's day-to-day work, but also acquiring external funding requires accurate timing, financial structuring and planning of scientific work. If scientists have little awareness of these skills, this can hinder their attitude and confidence in achieving their goals. In the workshop self-awareness was raised to show the participants how to hone these project-management related skills. More specifically, these characteristics of project management were compiled by

- recognising how much experience someone has already gained in life regarding project management,
- reflecting upon personal typical approaches, preferences, and strategies that have been applied during previous projects,
- figuring out typical role/s in a project (i.e. expert, leader, organiser, coordinator, conflict manager, facilitator, generalist, specialist, team player etc.),
- realising someone's particular strengths in project work and project teams,
- learning how to integrate the new findings into day-to-day work and long-term projects,
- improving self-efficacy regarding upcoming projects, and
- communicating project management experience in application processes.

8 InvasIC Activities

To promote the ideas and results of InvasIC and discuss them with leading experts from industry and academia, international guest speakers were invited to the “InvasIC Seminar”. In addition, members of InvasIC gave talks and seminars at important research sites and conferences (“Invited Talks”) or organised conferences and workshops (“Organised Conferences and Workshops”) on the topics of invasive computing.
The InvasIC Seminar is a series of talks given at one of the three sites (“InvasIC Seminar”). Videos of the respective talks are provided at our website `http://www.invasic.de`.

Figure 8.1: Dr. Silvia Melitta Mueller, IBM visiting our CRC/Transregio in Erlangen

Figure 8.2: Prof. Alberto Bosio (on the right) and Prof. Jürgen Teich (left).

InvasIC Seminar

Place and Date	Title	Speaker
Munich, May 17, 2019	On the road to Self-Driving IC Design Tools and Flows	Prof. Andrew B. Kahng (University of California, San Diego)
Erlangen, May 17, 2019	Optimizing Enterprise Servers across the Hardware and Software Stack	Dr. Silvia Melitta Mueller (IBM, Germany)
Erlangen, July 19, 2019	Approximate Computing: Design and Test for Integrated Circuits	Prof. Alberto Bosio (Ecole Centrale de Lyon, France)
Erlangen, Sept. 27, 2019	Design of Decentralized Embedded IoT Systems	Prof. Sebastian Steinhorst (Technische Universität München, Germany)
Erlangen, Nov. 29, 2019	Scalable Data Management on Modern Networks	Prof. Carsten Binnig (Technische Universität Darmstadt, Germany)

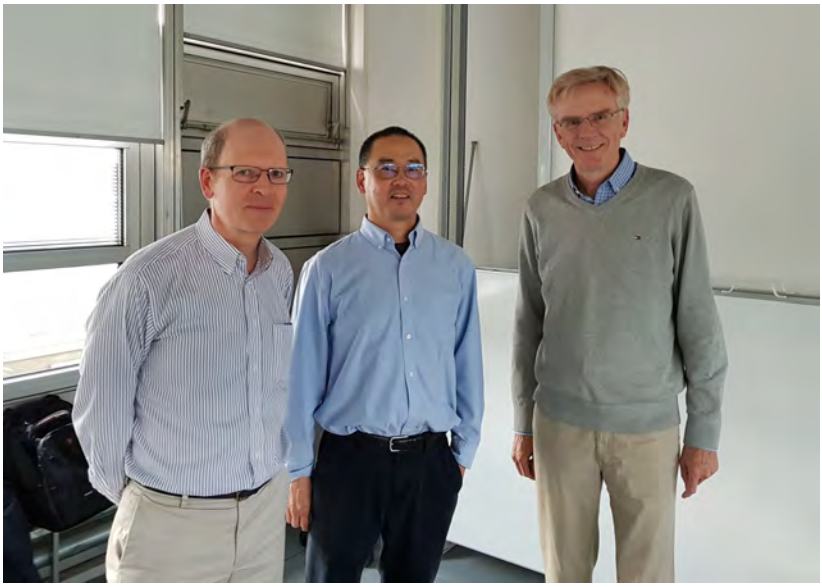


Figure 8.3: Prof. Andrew B. Kahng (middle) together with Prof. Ulf Schlichtmann (left) and Prof. Andreas Herkersdorf (right).

Invited Talks

Place and Date	Title	Speaker
Hawaii, USA, Jan. 4, 2019 ASECOLab	Talk: Adaptive Memory Protection for Many-Core Systems	Prof. W. Schröder-Preikschat (FAU)
Darmstadt, Germany, April 10, 2019 International Symposium on Applied Reconfigurable Computing	Talk: Hybrid Prototyping for Manycore Design and Validation	Prof. J. Becker, L. Masing (KIT)
Tokyo, Japan, April 18, 2019 University of Tokyo	Talk: Multi-Core Computing with Timing, Reliability, and Security Guarantees	Prof. J. Teich (FAU)
Alghero, Italy, April 30, 2019 ACM International Conference on Computing Frontiers 2019	Talk: Anytime Instructions for Programmable Accuracy Floating-Point Arithmetic	M. Brand (FAU)
Copenhagen, Denmark, May 20-23, 2019 32nd International Conference on Architecture of Computing Systems	Keynote: Predictability Issues in Operating Systems	Prof. W. Schröder-Preikschat (FAU)
Xi'an, China, June 21/22, 2019 Xidian University	Talk: Machine Learning Approaches for Efficient Design Space Exploration of Application-specific NoCs	Prof. U. Schlichtmann, Dr.-Ing. L. Zhang (TUM)
Phoenix, USA, June 25, 2019 9th International Workshop on Runtime and Operating Systems for Supercomputers	Talk: Asynchronous Abstract Machines: Anti-noise System Software for Many-core Processors	Dr.-Ing. T. Hönig (FAU)
Dortmund, Germany July 4/5, 2019 Technische Universität Dortmund	Talk: Run-Time Enforcement of Non-functional Program Properties on MPSoCs	Prof. J. Teich (FAU)
	Talk: As Embedded Systems Became Serious Grown-Ups, They Decide on Their Own	Prof. A. Herkersdorf (TUM)

Place and Date	Title	Speaker
Lausanne, Switzerland July 31, 2019 International Symposium on Low Power Electronics and Design	Talk: NCFET-Aware Voltage Scaling	Sami Salamin Martin Rapp (KIT)
Wellington, New Zealand, Sept. 23, 2019 University of Wellington	Talk: Predictability Issues in Operating Systems: Time, Space, Energy	Prof. W. Schröder-Preikschat, Dr.-Ing. T. Hönig (FAU)
Otago, New Zealand, Oct. 4, 2019 University of Otago	Talk: Adaptive Memory Protection for Many-core Systems	Prof. W. Schröder-Preikschat (FAU)
Tainan, Taiwan, Oct 23, 2019 AI College of National Chiao Tung University in Tainan	Talk: Novel Ideas in Timing of Digital Circuits	Prof. U. Schlichtmann (TUM)
Helsinki, Finland, Oct 29, 2019 IEEE Nordic Circuits and Systems Conference	Keynote: Tackling the MPSoC Data Locality Challenge with Regional Coherence and Near Memory Acceleration	Prof. A. Herkersdorf (TUM)

Figure 8.4: Prof. Jürgen Teich visited the University of Tokyo giving an invited talk at the Department of Creative Informatics.

Organised Conferences and Workshops

Place and Date	Title	Organiser
Leiden, Netherlands, Jan. 4, 2019	Scheduling Meets Fixed-Parameter Tractability	Prof. N. Megow (UB), Dr. M. Mnich (University of Bonn), Prof. G. Woeginger (RWTH Aachen)
Florence, Italy, March 26, 2019 Design, Automation and Test in Europe (DATE) 2019	Exhibition Theatre Session at DATE	Prof. J. Teich (FAU)
Canmore, Canada, Sept. 3/4, 2019	1st ACM/IEEE Workshop on Machine Learning for CAD	Prof. J. Henkel (KIT), Prof. U. Schlichtmann (TUM)
Auckland, New Zealand, Sept. 16, 2019 New Zealand-Germany Research Workshop	Time predictability, energy awareness and security in embedded and real-time systems	Prof. W. Schröder-Preikschat, Dr.-Ing. T. Hönig (FAU)

DATE 2019

Professor Jürgen Teich organised as its General Chair the Design, Automation, and Test in Europe (DATE) 2019 Conference and Exhibition which took place at the Firenze Fiera in Florence, Italy from 25 to 29 March 2019 [TF19]. The conference attracted more than 1,600 registrations from over 40 countries and concluded with excellent feedback from both participants and exhibitors. DATE combines the world's favourite electronic systems design and test conference with an international exhibition for electronic design, automation, and test from system-level hardware and software implementation right down to integrated circuit design.

On Monday, the DATE week started with five in-depth technical tutorials on the main topics of DATE as well as a hands-on industry tutorial given by leading experts in their respective fields. The topics covered Machine Learning for Manufacturing and Test, OpenCL Design Flows for FPGAs, Approximate Computing, Hardware-Based Security, and Safety and Security in Automotive, while the hands-on tutorial was on Quantum Computing with IBM Q and Qiskit.

During the Opening Ceremony on Tuesday, plenary keynote lectures were given by Astrid Elbe, managing director of Intel Labs Europe, and Jürgen Bortolazzi, director driver assistance systems and highly auto-

Figure 8.5: Impressions from DATE 2019. (photographs DATE 2019 / © Cruz Garcia)

mated driving at Porsche. On the same day, the Executive Track offered a series of business panels with executive speakers from companies leading the design and automation industry, discussing hot topics. Further, a talk by Claudio Giorgione, curator of the Leonardo Department at the National Museum of Science and Technology Milano, gave insight into the life and work of Leonardo da Vinci in honour of the 500th anniversary of his death, which is celebrated in Florence in 2019.

Figure 8.6: Impressions from DATE 2019. (photographs DATE 2019 / © Cruz Garcia)

The main conference program included 58 technical sessions organised in eight parallel tracks from the following four areas:

- D – Design Methods and Tools
- A – Application Design
- T – Test and Dependability
- E – Embedded and Cyber-physical Systems

and from several special sessions on hot topics such as Emerging Design Technologies, Design and Test of Secure Systems, IoT Security, Embedded Systems for Deep Learning, Augmented Living and Personalized Healthcare, Robotics and Industry 4.0, as well as results and lessons

learned from European projects. In addition, numerous interactive presentations were given during five IP sessions. The technical program was composed of 834 submitted papers with an acceptance rate of 24%.

Two special days in the program focused on areas bringing new challenges to the system design community, including Embedded Meets Hyperscale and HPC and Model-Based Design of Intelligent Systems. Each of the special days has had a full program of panels, tutorials and technical presentations, and a lunchtime keynote.

On Wednesday, the keynote on the topic of heterogeneous, high-scale computing in the era of cloud-connected devices by David Pellerin, Amazon US, was the highlight of the special day on Embedded Meets Hyperscale and HPC. During the special day on Model-Based Design of Intelligent Systems on Thursday, Edward Lee from UC Berkeley took "A Fundamental Look at Models and Intelligence" in his keynote.

One of the highlights of the DATE week was the DATE Party as one of the main networking opportunities. The party took place in the Palazzo Borghese, which is a beautiful example of neoclassic architecture in the heart of Florence. Local delights, entertaining music, and a visit by Leonardo da Vinci made this evening a memorable event on its own!

Figure 8.7: Further impressions from DATE 2019. (photographs DATE 2019 / © Cruz Garcia)

On Friday, ten workshops covered several hot topics from areas such as Open Source and Machine Learning in EDA; Emerging Techniques for Memories, Interconnections, and Quantum Computing; Hardware Design, Synthesis, and Approximate Computing; as well as EDA in application domains such as Autonomous Systems and IoT. Furthermore, an International F1/10 Autonomous Racing Demo took place, supported by the IEEE Council on Electronic Design Automation. This presentation of open-source, affordable, and high-performance 1/10 scale autonomous vehicles was a particular highlight on the last day of DATE 2019.

Exhibition Theatre Session at DATE 2019

Prof. Teich also organised an Exhibition Theatre Session on "DFG Collaborative Funding Instruments" on 26 March with an associated exhibition of selected currently funded collaborative research initiatives. The exhibition ran for three days (Tuesday to Thursday). The session was chaired by German Research Foundation (DFG) program director Dr. Andreas Raabe who started with an introduction of which types of funding instruments are offered in Germany, but also funding opportunities for international cooperations. After this introduction, concrete initiatives in the scope of topics of DATE were shortly introduced and summarised by representatives with a majority of these initiatives also exhibiting during

Figure 8.8: Impressions from the Exhibition Theatre Session at DATE 2019. (photographs DATE 2019 / © Cruz Garcia)

the conference. Two Priority Programmes (SPP 1648 Software for Exascale Computing and SPP 2037 Scalable Data Management for Future Hardware), three Collaborative Research Centres (SFB 901 On-the-fly Computing, SFB 912 Highly Adaptive Energy Efficient Computing and SFB 876 Providing Information by Resource-Constrained Data Analysis) and the CRC/Transregio 89 Invasive Computing, as well as the Research Unit FOR 1800 (Controlling Concurrent Change – Towards Self-Aware Automotive And Space Vehicles) and a Bi-National Research Project (Conquering MPSoC Complexity with Principles of a Self-Aware Information) used the opportunity to present newest ideas, work-in-progress and lessons learned from the project.

Public Relations

Long Night of the Sciences 2019

Every other year, the Long Night of the Sciences electrifies the metropolitan region around Nuremberg, Fürth and Erlangen. This kind of event is the biggest in Germany with around 20,000 visitors. Researchers take this terrific possibility to showcase their research work to the broad audience. Also we took the chance to present our CRC. On one hand, we demonstrated research on real-time multicore computing to visitors who are already familiar with computer science. In turn, the demonstrations opened up for interesting discussions about the characteristics of invasive computing.

Figure 8.9: Impressions from the Long Night of the Sciences 2019 at FAU.

On the other hand, we introduced “InvasiTrax”, a game based on the GraviTrax construction kit being very popular among children as a perfect way to explain invasive computing to people who have never heard of it before. Interactively visitors experience the differences between a “normal” multicore system and a system built upon invasive computing. InvasiTrax was particularly well received by children. We are very pleased that also a lot of girls came around to get to know InvasiTrax and learn something about invasive computing. Maybe we could contribute to more female students deciding for a STEM study program in the future.

Moreover, our self-made movie “InvasIC for Dummies”[15] was shown, adding a touch of multimedia to our presentation. Furthermore, we portrayed general information about the CRC/Transregio at a poster. All in all, we are very satisfied with the Long Night of the Sciences 2019. We have been able to introduce invasive computing to a large number of visitors enlarging the comprehension for this young research field and raise awareness for computer science in general.

LZE Tech Day

The Leistungszentrum Elektroniksysteme (LZE) is a joint initiative of the Fraunhofer-Gesellschaft, its Institutes IIS and IISB and FAU, together with other non-university research institutions and associated industrial partners. The LZE is breaking new ground here. With novel structures

Figure 8.10: Impressions from the LZE Tech Day 2019.

[15] https://www.youtube.com/watch?v=4kOQYHhnZW0

and cooperation models between science and industry, the successful transfer of research results has been initiated.

At the LZE Tech Day, examples from different stages of the innovation chain were shown ranging from new wide-range wireless communication technology as a best practice for comprehensive, successful market development, to completed research projects that are in the early stages of exploitation, to technological development that is just beginning. We presented an invasive parallel Shallow Water application demonstration to raise awareness for the research work of the CRC/Transregio 89.

B2Run in Nuremberg

"Run for Fun" was the theme under which our team participated at this year's B2Run in Nuremberg in July. The B2Run is a company racing series that is held at 17 locations in Germany. In Nuremberg the course leads around the great and the small Dutzendteich with the finish in the Max-Morlock-Stadium. Despite the very hot weather conditions all ten runners reached the finish of the almost 6 km long running track and enjoyed the great atmosphere. Inspired by the cheering spectators, the team proved that they can achieve a great deal together, whether on the PC, in the lab or in sports.

Figure 8.11: InvasIC runners at the B2Run in Nuremberg.

9 Awards

Emerging Talents Initiative

Dr.-Ing. Timo Hönig received a research grant by the Emerging Talents Initiative (ETI) of FAU for his project “Energy-Aware Gearing of System Software for Adaptive Leverage of Renewable Energies”. The project investigates the effects of the digitisation of power grids on the system-software design for complex computing systems (e.g. operating systems, workload management systems). In particular, usage and utilisation patterns of high-performance computing systems (e.g. HPC clusters) are considered. Following on from this, it is examined how operating patterns are applied by suitable models in control applications for the operation of complex computer systems. This serves to reduce or increase the electrical power demand of systems. Increasing the electrical power demand is necessary in situations where power grids have to be relieved by key consumers due to large quantities of renewable energies within the power grid.

PhD Forum Best Poster Prize at DATE 2019 for Tobias Schwarzer

Tobias Schwarzer (FAU) received the PhD Forum Best Poster Prize at DATE 2019 for his poster titled “System-Level Mapping and Synthesis of Data Flow-Oriented Applications on MPSoCs” [Sch19]. The prize is supported by EDAA, ACM Sigda and IEEE CEDA.

Figure 9.1: Tobias Schwarzer receiving the PhD Forum Best Poster Prize at DATE 2019 in Florence.

Best Presentation Award at the RTAS 2019 for Simon Schuster

Simon Schuster (FAU) received the Best Presentation Award for his presentation on the paper “Proving Real-Time Capability of Generic Operating Systems by System-Aware Timing Analysis” [Sch+19a] contributed by Simon Schuster, Peter Wägemann, Peter Ulbrich and Prof. Wolfgang Schröder-Preikschat (FAU) at RTAS 2019.

VDE Award 2019 for Nael Fasfous

Nael Fasfous (TUM) has been awarded the VDE Award 2019, in recognition of his Master Thesis on the topic of “Compact Directories with Hybrid Architecture Aware Eviction Policies for Distributed Shared Memory MPSoCs”. In his thesis, Nael proposed innovative cache eviction strategies for Multiprocessor System-on-Chip architectures. WithinProject B5, Akshay Srivatsa was the advisor of this thesis. The VDE Award has been given to Nael Fasfous from the Association of German Engineers (Verein Deutscher Ingenieure, VDE Südbayern) on November 24, 2019.

Figure 9.2: Simon Schuster (on the left) and Nael Fasfous (on the right).

Best Poster and Presentation Award at MEMSYS 2019 for Sven Rheindt

Sven Rheindt (TUM) received the Best Poster and Presentation Award for his presentation on the paper “Near-Memory Graph Copy Enhanced System-Software” [Rhe+19a] contributed by Sven Rheindt, Andreas Fried, Oliver Lenke, Lars Nolte, Thomas Wild, and Andreas Herkersdorf (TUM) at International Symposium on Memory Systems (MEMSYS 19).

IEEE CS TTTC Outstanding Service Award at the DATE 2019 for Prof. Jürgen Teich

Prof. Teich was honoured with the IEEE CS TTTC Outstanding Service Award at the Design, Automation and Test in Europe 2019 (DATE) in recognition of significant service as DATE 2019 General Chair.

Figure 9.3: Prof. Jürgen Teich at DATE 2019.

Best Paper Award at SAMOS XIX Conference for Akshay Srivatsa

The paper “CoD: Coherence-on-Demand – Runtime Adaptable Working Set Coherence for DSM-based Manycore Architectures” [Sri+19] authored by Akshay Srivatsa, Sven Rheindt, Dirk Gabriel, Thomas Wild and Andreas Herkersdorf and presented at this year’s SAMOS Conference (International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation) received the best paper award. It extends the already published region-based cache coherence (RBCC) approach, which provides dynamically changeable coherence support in a tiled manycore processor architecture, and shows design and implementation details of the Coherency Region Manager, the major component to realise this property.

10 Industrial and Scientific Board

For the promotion of our ideas to the industrial community and for the discussion with peer colleagues world-wide, we established the InvasIC Industrial and Scientific Board. Members of the board in its current constitution are 8 experts from 7 institutions in industry and university:

IBM

Dr. Peter-Hans Roth (IBM Böblingen)

Dr. Patricia Sagmeister (IBM Rüschlikon)

Intel

Hans-Christian Hoppe (Intel Director of ExaCluster Lab Jülich)

Siemens

Urs Gleim (Head of Research Group Parallel Systems Germany, Siemens Corporate Technology)

University of Edinburgh

Prof. Dr. Michael O'Boyle
(Director Institute for Computing Systems Architecture)

BETTEN & RESCH Patent & Trademark Attorneys

Prof. Dr. Christoph von Praun
(European Trademark Attorney)

IAV – Automotive Engineering

Elmar Maas (IAV, Gifhorn)

Xilinx

Michaela Blott (Xilinx, Dublin)

11 Publications

[AKH19] H. Amrouch, H. Khdr, and J. Henkel. “Aging Effects: From Physics to CAD”. In: *Harnessing Performance Variability in Embedded and High-performance Many/Multi-core Platforms*. Springer, 2019, pp. 43–69.

[Ana+19] N. Anantharajaiah, F. Kempf, L. Masing, F. M. Lesniak, and J. Becker. “Dynamic and Scalable Runtime Block-based Multicast Routing for Networks on Chips”. In: *Proceedings of the 12th International Workshop on Network on Chip Architectures*. NoCArc. Columbus, Ohio: ACM, 2019, 10:1–10:6. DOI: 10.1145/3356045.3360718.

[Bau+19] L. Bauer et al. “Analyses and Architectures for Mixed-Critical Systems: Industry Trends and Research Perspective”. In: *International Conference on Embedded Software (EMSOFT)*. Invited Special Session Extended Abstract. New York City, NY, USA, Oct. 2019, 13:1–13:2.

[Bra+19a] M. Brand, M. Witterauf, F. Hannig, and J. Teich. “Anytime Instructions for Programmable Accuracy Floating-Point Arithmetic”. In: *Proceedings of the ACM International Conference on Computing Frontiers (CF)* (Alghero, Sardinia, Italy). ACM, Apr. 30–May 2, 2019, pp. 215–219. DOI: 10.1145/3310273.3322833.

[Bra+19b] M. Brand, M. Witterauf, É. Sousa, A. Tanase, F. Hannig, and J. Teich. “*-Predictable MPSoC Execution of Real-Time Control Applications Using Invasive Computing”. In: *Concurrency and Computation: Practice and Experience* (Feb. 2019). DOI: 10.1002/cpe.5149.

[CCG19] J. A. Chacko, I. A. Comprés Ureña, and M. Gerndt. “Integration of Apache Spark with Invasive Resource Manager”. In: *2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation*. Best Paper Award. 2019.

[CG19] M. Chadha and M. Gerndt. “Modelling DVFS and UFS for Region-Based Energy Aware Tuning of HPC Applications”. In: *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2019.

[Che+19] J.-J. Chen, T. Hahn, R. Hoeksma, N. Megow, and G. von der Brüggen. “Scheduling Self-Suspending Tasks: New and Old Results”. In: *31st Euromicro Conference on Real-Time Systems (ECRTS)*. 2019.

[Dam19] M. Damschen. “Worst-Case Execution Time Guarantees for Runtime-Reconfigurable Architectures”. Dissertation. Chair of Embedded Systems (CES), Department of Informatics, Karlsruhe Institute of Technology, Germany, 2019.

[DBH19a] M. Damschen, L. Bauer, and J. Henkel. “WCET Guarantees for Opportunistic Runtime Reconfiguration”. In: *IEEE/ACM 38th International Conference on Computer-Aided Design (ICCAD)*. Westminster, CO, USA, Nov. 2019.

[DBH19b] M. Damschen, L. Bauer, and J. Henkel. “Worst-Case Execution Time Guarantees for Runtime-Reconfigurable Architectures”. Ph.D. Forum at IEEE/ACM 22nd Design, Automation and Test in Europe Conference (DATE). Florence, Italy, Mar. 2019.

[Dam+19] M. Damschen, M. Rapp, L. Bauer, and J. Henkel. “i-Core: A runtime-reconfigurable processor platform for cyber-physical systems”. In: *Embedded, Cyber-Physical, and IoT Systems*. Ed. by S. S. Bhattacharyya, M. Potkonjak, and S. Velipasalar. Springer International Publishing, 2019.

[GSW19] D. Gabriel, W. Stechele, and S. Wildermann. “Resource-Aware Parameter Tuning for Real-Time Applications”. In: *Architecture of Computing Systems – ARCS 2019*. Ed. by M. Schoeberl, C. Hochberger, S. Uhrig, J. Brehm, and T. Pionteck. Springer International Publishing, 2019, pp. 45–55. DOI: 10.1007/978-3-030-18656-2_4.

[Har19] T. Harbaum. “Dynamisch adaptive Mikroarchitekturen mit optimierten Speicherstrukturen und variablen Befehlssätzen”. Dissertation. Institut für Technik der Informationsverarbeitung (ITIV), Fakultät für Elektrotechnik und Informationstechnik, Karlsruher Institut für Technologie (KIT), June 25, 2019.

[Hei+19a] C. Heidorn, M. Witterauf, F. Hannig, and J. Teich. “Efficient Mapping of CNNs onto Tightly Coupled Processor Arrays”. In: *Journal of Computers (JCP)* 14.8 (Aug. 2019), pp. 541–556. DOI: 10.17706/jcp.14.8.541-556.

[Hei+19b] B. Heinloth, M. Ammon, D. Nguyen, T. Hönig, V. Sieh, and W. Schröder-Preikschat. “Cocoon: Custom-Fitted Kernel Compiled on Demand”. In: *Proceedings of the 10th Workshop on Programming Languages and Operating Systems (PLOS)*. ACM. New York, NY, USA: ACM Digital Library, 2019, pp. 1–7. DOI: 10.1145/3365137.3365398.

[MoH19] A. Mo-Hellenbrand. “Resource-Aware and Elastic Parallel Software Development for Distributed-Memory HPC Systems”. Dissertation. Munich: Technische Universität München, 2019. URL: http://mediatum.ub.tum.de/?id=1471007.

[HKR19] J. Henkel, H. Khdr, and M. Rapp. “Smart Thermal Management for Heterogeneous Multicores”. In: *Design, Automation & Test in Europe (DATE)*. IEEE. 2019, pp. 132–137.

[Her19] A. Herkersdorf. “Tackling the MPSoC Data Locality Challenge with Regional Coherence and Near Memory Acceleration”. Keynote talk, 2019 IEEE Nordic Circuits and Systems Conference (NorCAS). Oct. 29, 2019.

[HHS19] T. Hönig, B. Herzog, and W. Schröder-Preikschat. “Energy-Demand Estimation of Embedded Devices Using Deep Artificial Neural Networks”. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC)*. ACM Digital Library, 2019, pp. 617–624. DOI: 10.1145/3297280.3297338.

[HMA19] F. Hundhausen, D. Megerle, and T. Asfour. “Resource-Aware Object Classification and Segmentation for Semi-Autonomous Grasping with Prosthetic Hands”. In: *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*. 2019.

[JNG19] J. John, S. Narvaez R, and M. Gerndt. “Invasive Computing for Power Corridor Management”. In: *ParCo 2019: International Conference on Parallel Computing*. 2019.

[Kes+20] O. Keszocze, M. König, M. Brand, and J. Teich. “Error Analysis for Loop Programs Using Anytime Instructions in Approximate Computing”. In: *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*. Stuttgart, Germany, 2020.

[KAH19] H. Khdr, H. Amrouch, and J. Henkel. “Dynamic Guardband Selection: Thermal-Aware Optimization for Unreliable Multi-Core Systems”. In: *Transactions on Computers (TC)* (2019).

[Kho19] F. Khosravi. “System-Level Reliability Analysis and Optimization in the Presence of Uncertainty”. Dissertation. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Aug. 5, 2019.

[Kön19] M. König. *Approximative Schleifen mit Anytime Instruktionen in der Simulationsumgebung Daisy*. Master Thesis. Friedrich-Alexander University Erlangen-Nürnberg (FAU). Sept. 1, 2019.

[LMS19] A. Listl, D. Mueller-Gritschneder, and U. Schlichtmann. “MAGIC: A Wear-leveling Circuitry to Mitigate Aging Effects in Sense Amplifiers of SRAMs”. In: *2019 IEEE 17th International New Circuits and Systems Conference (NEWCAS)*. July 2019.

[Lis+19] A. Listl, D. Mueller-Gritschneder, U. Schlichtmann, and S. Nassif. “SRAM Design Exploration with Integrated Application-Aware Aging Analysis”. In: *Design, Automation, and Test in Europe (DATE)*. Mar. 2019, pp. 1249–1252.

[Mae19] P. Maene. “Lightweight Roots of Trust for Modern Systems-on-Chip”. Dissertation. Faculty of Engineering Science, KU Leuven, Belgium, Oct. 2019.

[Mai+19] S. Maier, T. Hönig, P. Wägemann, and W. Schröder-Preikschat. “Asynchronous Abstract Machines: Anti-noise System Software for Many-core Processors”. In: *Proceedings of the 9th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)* (Phoenix, AZ, USA). ACM, 2019, pp. 19–26. DOI: 10.1145/3322789.3328744.

[Mar+20] A. Marchetti-Spaccamela, N. Megow, J. Schlöter, M. Skutella, and L. Stougie. “On the Complexity of Conditional DAG Scheduling in Multiprocessor Systems”. In: *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2020.

[MLB19] L. Masing, F. Lesniak, and J. Becker. “Hybrid Prototyping for Manycore Design and Validation”. In: *Applied Reconfigurable Computing*. Springer International Publishing, 2019, pp. 319–333.

[MMS20] M. Mettler, D. Mueller-Gritschneder, and U. Schlichtmann. “Runtime Monitoring of Inter- and Intra-Thread Requirements on Embedded MPSoCs”. In: 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID) (Jan. 2020).

[Mue19] D. Mueller-Gritschneder. *Advanced Virtual Prototyping and Communication Synthesis for Integrated System Design at Electronic System Level*. 2019.

[Nar18] S. Narvaez. “Power model for resource-elastic applications”. Master Thesis. Munich: Technische Universität München, 2018. URL: http://mediatum.ub.tum.de?id=1475095.

[PNG19] R. Palutke, A. Neubaum, and J. Götzfried. “SEVGuard: Protecting User Mode Applications using Secure Encrypted Virtualization”. In: *SecureComm 2019 Proceedings* (Orlando). Springer, Oct. 24, 2019.

[PH19] A. Pathania and J. Henkel. “HotSniper: Sniper-Based Toolchain for Many-Core Thermal Simulations in Open Systems”. In: *Embedded Systems Letters (ESL)* (2019).

[PBB19] A. Pöppl, S. Baden, and M. Bader. “A UPC++ Actor Library and Its Evaluation on a Shallow Water Proxy Application”. In: *Parallel Applications Workshop, Alternatives To MPI+X*. IEEE. Denver, Colorado, United States of America: IEEE, Nov. 2019.

[Pou+19a] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich, and J. Henkel. “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems”. In: *40th IEEE Real-Time Systems Symposium (RTSS)*. 2019.

[Pou+19b] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich. “Isolation-Aware Timing Analysis and Design Space Exploration for Predictable and Composable Many-Core Systems”. In: *31th Euromicro Conference on Real-Time Systems (ECRTS)*. Stuttgart, Germany, 2019, 12:1–12:24. DOI: 10.4230/LIPIcs.ECRTS.2019.12.

[Pou+20] B. Pourmohseni, F. Smirnov, S. Wildermann, and J. Teich. “Real-Time Task Migration for Dynamic Resource Management in Many-Core Systems”. In: *Workshop on Next Generation Real-Time Embedded Systems (NG-RES)*. 2020.

[Pou+19c] B. Pourmohseni, S. Wildermann, M. Glaß, and J. Teich. “Hard Real-Time Application Mapping Reconfiguration for NoC-Based Many-Core Systems”. In: *Real-Time Systems* (2019), pp. 1–37. DOI: 10.1007/s11241-019-09326-y.

[Rap+19a] M. Rapp, A. Pathania, T. Mitra, and J. Henkel. “Prediction-Based Task Migration on S-NUCA Many-Cores”. In: *Design, Automation & Test in Europe (DATE)*. IEEE. 2019, pp. 1579–1582.

[Rap+19b] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel. “Power-and Cache-Aware Task Mapping with Dynamic Power Budgeting for Many-Cores”. In: *IEEE Transactions on Computers* (2019).

[Rap+19c] M. Rapp, S. Salamin, H. Amrouch, G. Pahwa, Y. Chauhan, and J. Henkel. “Performance, Power and Cooling Trade-Offs with NCFET-based Many-Cores”. In: *Design Automation Conference (DAC)*. ACM. 2019, p. 41.

[Rei+19] S. Reif, P. Raffeck, H. Janker, L. Gerhorst, T. Hönig, and W. Schröder-Preikschat. “Earl: Energy-Aware Reconfigurable Locks”. In: *Proceedings of the 9th Embedded Operating Systems Workshop (EWiLi)*. Forthcoming. ACM. New York, NY, USA: ACM SIGBED Review, 2019.

[Rhe+19a] S. Rheindt, A. Fried, O. Lenke, L. Nolte, T. Wild, and A. Herkersdorf. “NEMESYS: Near-Memory Graph Copy Enhanced System-Software”. In: *MEMSYS 19: The International Symposium on Memory Systems*. Washington DC, 2019.

[Rhe+19b] S. Rheindt, S. Maier, F. Schmaus, T. Wild, W. Schröder-Preikschat, and A. Herkersdorf. “SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures”. In: *Proceedings of the 19th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*. 2019.

[RHT19] S. Roloff, F. Hannig, and J. Teich. *Modeling and Simulation of Invasive Applications and Architectures*. Springer, May 2019. 168 pp. DOI: 10.1007/978-981-13-8387-8.

[Ros+18] E. Rossi, M. Damschen, L. Bauer, G. Buttazzo, and J. Henkel. “Preemption of the Partial Reconfiguration Process to Enable Real-Time Computing with FPGAs”. In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 11.2 (Nov. 2018), 10:1–10:24. DOI: 10.1145/3182183.

[Sag+19] M. Sagi, N. A. V. Doan, T. Wild, and A. Herkersdorf. “Multi-core Power Estimation using Independent Component Analysis based Modeling”. In: *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. Oct. 2019.

[Sal+19] S. Salamin, M. Rapp, H. Amrouch, G. Pahwa, Y. Chauhan, and J. Henkel. “NCFET-Aware Voltage Scaling”. In: *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE. 2019.

[Sch+19a] S. Schuster, P. Wägemann, P. Ulbrich, and W. Schröder-Preikschat. “Proving Real-Time Capability of Generic Operating Systems by System-Aware Timing Analysis”. In: *Proceedings of the 25th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE Computer Society, 2019, pp. 313–330. DOI: 10.1109/RTAS.2019.00034.

[Sch19] T. Schwarzer. “System-Level Mapping and Synthesis of Data Flow-Oriented Applications on MPSoCs”. Ph.D. Forum at the Design, Automation and Test in Europe Conference (DATE). Ph.D. Forum Best Poster Award. Mar. 2019.

[Sch+19b] T. Schwarzer et al. “Compilation of Dataflow Applications for Multi-Cores Using Adaptive Multi-Objective Optimization”. In: *ACM Transactions on Design Automation of Electronic Systems* 24.3 (Mar. 2019), 29:1–29:23. DOI: 10.1145/3310249.

[Sie+19] V. Sieh et al. “Combining Automated Measurement-Based Cost Modeling With Static Worst-Case Execution-Time and Energy-Consumption Analyses”. In: *IEEE Embedded Systems Letters* 11.2 (June 2019), pp. 38–41.

[Sim+20] B. Simon, J. Falk, N. Megow, and J. Teich. “Energy Minimization in DAG Scheduling on MPSoCs at Run-Time: Theory and Practice”. In: *Workshop on Next Generation Real-Time Embedded Systems*. 2020.

[Spi+19] J. Spieck, S. Wildermann, T. Schwarzer, J. Teich, and M. Glaß. “Data-Driven Scenario-based Application Mapping for Heterogeneous Many-Core Systems”. In: *Multicore/Many-core Systems-on-Chip (MCSoC)* (Singapore). Oct. 1–4, 2019.

[SWT19] J. Spieck, S. Wildermann, and J. Teich. “Run-Time Scenario-Based MPSoC Mapping Reconfiguration Using Machine Learning Models”. In: *1st ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. 2019.

[Sri+19] A. Srivatsa, S. Rheindt, D. Gabriel, T. Wild, and A. Herkersdorf. “CoD: Coherence-on-Demand – Runtime Adaptable Working Set Coherence for DSM-Based Manycore Architectures”. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by D. N. Pnevmatikatos, M. Pelcat, and M. Jung. Cham: Springer International Publishing, 2019, pp. 18–33.

[TF19] J. Teich and F. Fummi. “Conference Reports: Recap of DATE 2019 in Florence, Italy”. In: *IEEE Design & Test* 36.4 (2019), pp. 59–61. DOI: 10.1109/MDAT.2019.2915112.

[Tei+20a] J. Teich, P. Mahmoody, B. Pourmohseni, S. Roloff, W. Schröder-Preikschat, and S. Wildermann. “Run-Time Enforcement of Non-functional Program Properties on MPSoCs”. In: *A Journey of Embedded and Cyber-Physical Systems*. Ed. by J.-J. Chen. Springer, 2020.

[Tei+20b] J. Teich, B. Pourmohseni, O. Keszocze, J. Spieck, and S. Wildermann. “Run-Time Enforcement of Non-Functional Application Requirements in Heterogeneous Many-Core Systems”. In: *Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2020, pp. 629–636.

[TV19a] F. Turan and I. Verbauwhede. “Compact and Flexible FPGA Implementation of Ed25519 and X25519”. In: *ACM Transactions on Embedded Computing Systems (TECS)* 18.3 (2019), p. 24.

[TV19b] F. Turan and I. Verbauwhede. “Propagating Trusted Execution through Mutual Attestation”. In: *4th Workshop on System Software for Trusted Execution (SysTEX)*. Huntsville, Ontario, Canada: ACM, 2019.

[WHT19] M. Witterauf, F. Hannig, and J. Teich. “Polyhedral Fragments: An Efficient Representation for Symbolically Generating Code for Processor Arrays”. In: *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)* (San Diego, CA, USA). IEEE, Oct. 9–11, 2019.

[WS19] A. Würstlein and W. Schröder-Preikschat. “T-IBE-T: Identity-Based Encryption for Inter-Tile Communication”. In: *Proceedings of the 12th European Workshop on Systems Security (EuroSec)*. ACM Digital Library, 2019, pp. 1–6. DOI: 10.1145/3301417.3312500.

[Zha+20] G. L. Zhang, M. Brunner, B. Li, G. Sigl, and U. Schlichtmann. “Timing Resilience for Efficient and Secure Circuits”. In: *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2020.