Sonderforschungsbereich/Transregio 89
Transregional Collaborative Research Center 89

Invasive Computing
Annual Report 2016

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 2016

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

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

Deputy Managing Director
Dr. Katja Lohmann
Lehrstuhl für Informatik 12
Universität Erlangen-Nürnberg
Cauerstraße 11, 91058 Erlangen

Public Relations
Dr. Sandra Mattauch
Lehrstuhl für Informatik 12
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 2016.

The CRC/Transregio InvasIC is funded by the Deutsche Forschungsgemeinschaft in its second funding phase from July 2014 – July 2018. 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, 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.

Exciting events in 2016 certainly were two Dagstuhl Seminars co-organised by members of our CRC/Transregio. In January, Prof. Dr. Michael Gerndt (TUM) and Prof. Dr.-Ing. Michael Glaß (FAU) organised and coordinated the Seminar 16052 on “Dark Silicon: From Embedded to HPC Systems”. Prof. Dr.-Ing. Jürgen Teich (FAU) and Prof. Dr. Ir. Ingrid Verbauwhede (KU Leuven, BE) were the organisers of the Seminar 16441 “Adaptive Isolation for Predictability and Security”, which took place in October.

2016 also was an outstanding year concerning the professional success of three members of the CRC/Transregio: Prof. Dr.-Ing. Michael Glaß (FAU), Dr.-Ing. Muhammad Shafique (KIT) and PD Dr.-Ing. habil. Daniel Lohmann (FAU) were appointed as professors at the highly reputable universities of Ulm, Vienna, and Hanover, respectively.

We would like to thank all members of the CRC/Transregio InvasIC 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 achieved in 2016, as well as about our research planned for the following years.

Jürgen Teich
Coordinator

Contents

I

Invasive Computing

1 About InvasIC

The Idea of Invasive Computing

The main idea and novelty of *invasive computing* is to introduce resource-aware programming support in the sense that a given program gets the ability to explore and dynamically spread its computations to processors similar to a phase of invasion, then to execute portions of code of high parallelism degree in parallel based on the available (invasible) region on a given multi-processor architecture. Afterwards, once the program terminates or if the degree of parallelism should be lower again, the program may enter a *retreat phase*, deallocate resources and resume execution again, for example, sequentially on a single processor. To support this idea of self-adaptive and resource-aware programming, new programming concepts, languages, compilers and operating systems are necessary as well as architectural changes in the design of MPSoCs (Multi-Processor Systems-on-a-Chip) to efficiently support invasion, infection and retreat operations by involving concepts for dynamic processor, interconnect and memory reconfiguration. Decreasing feature sizes have also led to a rethinking in the design of multi-million transistor system-on-chip (SoC) architectures, envisioning dramatically increasing rates of temporary and permanent faults and feature variations.

As we can foresee SoCs with 1000 or more processors on a single chip in the year 2020, static and central management concepts to control the execution of all resources might have met their limits long before and are therefore not appropriate. Invasion might provide the required *self-organising* behaviour to conventional programs for being able to provide scalability, higher resource utilisation, required fault-tolerance and, of course, also performance gains by adjusting the amount of allocated resources to the temporal needs of a running application. This thought opens a new way of thinking about *parallel algorithm design*. Based on algorithms utilising invasion and negotiating resources with others, we can imagine that corresponding programs become *personalised* objects, competing with other applications running simultaneously on an MPSoC.

First Achievements

A Transregional Collaborative Research Centre aggregating the best researchers from three excellent sites in Germany provides an ideal base to investigate the above revolutionary ideas. Starting off at basically zero in terms of invasive processor hardware, language, compiler, and operating-system availability, we have truly fostered the *fundamentals of invasive computing* in our first funding phase: These include the definition of required programming language elements for the specification of invasion operations as well as a set of constraints to argue about number, types, and state of resources that may be invaded defining the *invasive command space* (Project Area A). A first invasive language based on the language X10 by IBM as well as a compiler for translation of invasive X10 programs (Project Area C) onto a heterogeneous invasive multi-tile architecture that has also been successfully jointly architected (Project Area B) is meanwhile ready for experimentation on an FPGA-based prototype (Project Z2). The compiler interfaces to the invasive runtime support system *i*RTSS that provides for dedicated operating-system support for invasive computing. First invasive applications exploiting different types of processor and communication resources of an invasive network-on-chip (*i*NoC) are running successfully and have shown considerable gains in resource utilisation and computational efficiencies in comparison with their non-invasive counterparts.

Current Scientific Goals

A unique jewel of invasive computing, however, has not been exploited at all so far: By the fact that resources are temporally claimed (by default) in an exclusive manner, interferences due to multiple applications sharing the same resources being the reality on today's multicore systems may be reduced if not avoided completely. Moreover, *run-to-completion* is the default mode of thread execution. Finally, memory reconfiguration and isolation as well as bandwidth guarantees on the designed network-on-chip allow us also to provide predictable QoS also for communication.

In the current funding phase, we play out this ace systematically by tackling (a) *predictability* of (b) *multi-objective execution qualities* of parallel invasive programs and including their (c) *optimisation* and *exploration of design space*. Our joint investigations include new language constructs to define so-called *requirements* on desired, respectively amended qualities of execution. Application-specified qualities may not only be of type performance (e. g. execution time, throughput, etc.),

but do also include aspects of *security* and *fault tolerance*. Through analysis of application requirements from different domains including stream processing and malleable task applications, not only efficiency but also predictable execution qualities shall be demonstrated for applications stemming from robotics, imaging, as well as HPC. As another new yet very important facet of invasive computing, a special focus in the current funding phase is devoted to the problem of *dark silicon* and *energy-efficient computing*.

Long Term Vision

With the aforementioned fundamental investigations in mind, we intend to demonstrate that invasive computing will be a—if not the—vehicle for solving many current problems of multicore computing today by providing *resource awareness* for a mixture of *best-effort* applications and applications with *predictable quality*. We do expect that a huge application and business field in embedded system applications might be accessed through the foundations of invasive computing.

2 Participating University Groups

Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Hardware-Software-Co-Design

- Prof. Dr.-Ing. Jürgen Teich
- Prof. Dr.-Ing. Michael Glaß
- 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
- PD Dr.-Ing. Daniel Lohmann

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

Lehrstuhl für integrierte Systeme

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

Lehrstuhl für Rechnertechnik und Rechnerorganisation

- Prof. Dr. Michael Gerndt

Lehrstuhl für Technische Elektronik

- Prof. Dr. rer. nat. Doris Schmitt-Landsiedel

Lehrstuhl für Wissenschaftliches Rechnen

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

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 at the end of the second project phase and to decide on the further roadmap of specific hardware for invasive computing.

The four working groups **Predictability**, **Memory Hierarchy**, **Benchmarking and Evaluation** and **Power Efficiency and Dark Silicon** 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. J. Teich, Prof. Dr.-Ing. G. Snelting, Dr.-Ing. S. Wildermann*	A1
	Design-Time Characterisation and Analysis of Invasive Algorithmic Patterns *Prof. Dr.-Ing. M. Glaß, Prof. Dr. M. Bader*	A4
B: Architectural Research	Adaptive Application-Specific Invasive Microarchitecture *Prof. Dr.-Ing. J. Henkel, Dr.-Ing. L. Bauer, Prof. Dr.-Ing. J. Becker*	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
	Hardware Monitoring System and Design Optimisation for Invasive Architectures *Prof. Dr. rer. nat. D. Schmitt-Landsiedel, Prof. Dr.-Ing. U. Schlichtmann*	B4
	Invasive NoCs –– Autonomous, Self-Optimising Communication Infrastructures for MPSoCs *Prof. Dr.-Ing. J. Becker, Prof. Dr. sc. techn. A. Herkersdorf, Prof. Dr.-Ing. J. Teich*	B5
C: Compiler, Simulation and Run-Time Support	Invasive Run-Time Support System (*i*RTSS) *Prof. Dr.-Ing. W. Schröder-Preikschat, PD Dr.-Ing. D. Lohmann, Prof. Dr.-Ing. J. Henkel, Dr.-Ing. L. Bauer*	C1
	Simulative Design Space Exploration *Dr.-Ing. F. Hannig*	C2
	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*	C5
D: Applications	Invasive Software-Hardware Architectures for Robotics *Prof. Dr.-Ing. T. Asfour, Prof. Dr.-Ing. W. Stechele*	D1
	Invasion for High-Performance Computing *Prof. Dr. H.-J. Bungartz, Prof. Dr. M. Bader, Prof. Dr. M. Gerndt*	D3
Z: Administration	Validation and Demonstrator *Prof. Dr.-Ing. J. Becker, Dr.-Ing. F. Hannig, Dr.-Ing. T. Wild*	Z2
	Central Services *Prof. Dr.-Ing. J. Teich*	Z
WG: Working Groups	Predictability *Prof. Dr. M. Gerndt, Prof. Dr.-Ing. M. Glaß*	WG1
	Memory Hierarchy *Dr.-Ing. L. Bauer, Prof. Dr.-Ing. G. Snelting*	WG2
	Benchmarking and Evaluation *Prof. Dr. M. Bader, Prof. Dr.-Ing. W. Stechele*	WG3
	Power Efficiency and Dark Silicon *Dr.-Ing. F. Hannig, Prof. Dr.-Ing. J. Henkel*	WG4

4 Research Projects

A1: Basics of Invasive Computing

Jürgen Teich, Gregor Snelting, Stefan Wildermann

Andreas Zwinkau, Andreas Weichslgartner

The goal of Project A1 is to develop a programming model and the theoretical foundations for enforcing *predictability* of invasive program execution with multiple non-functional requirements. Research focuses on (a) a formal semantics and a nonstandard (dependent) type system of the invasive core language to provide resource usage guarantees and a memory model for invasive architectures, (b) programming extensions to express typical invasive programming patterns and non-functional requirements and to alleviate formal program analysis, and (c) run-time management strategies for feasible and optimal program execution. We describe our results obtained in these research areas during 2016 in the following.

*-Predictability

In [Tei+16], we have defined the notion of **-predictability*. It is driven by the fact that in many application domains, not a single but multiple non-functional requirements have to be enforced at the same time like, e. g. a timing deadline and a maximum power budget. The definition of *-predictability for a program implementation p under predictability analysis is given as follows.

Definition *Let o denote a non-functional property of a program (implementation) p, which has the input space given by I and state space of the execution environment given by Q. The predictability (marker) of objective o for program p is described by the interval*

$$o(p, Q, I) = [\inf_o(p, Q, I), \sup_o(p, Q, I)]$$

where $\inf$ *and* $\sup$ *denote the infimum and supremum under variation of all states $q \in Q$ and inputs $i \in I$, respectively.*

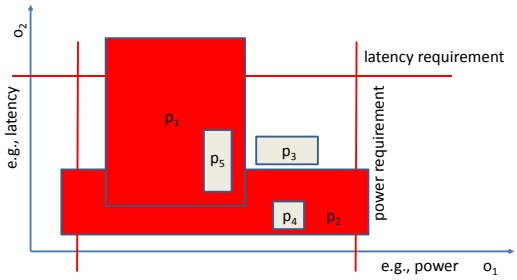


Figure 4.1: Pareto-front showing multiple program implementations p_i with two objectives each being uncertain but bounded by an interval. Also shown constraints (bounds) that must not be violated (from [Tei+16]).

As an example, a program p that under the uncertainty of dynamic power management and input leads to a total power consumption P between 1 and 2 W is described by $P(p, Q, I) = [1, 2]$. Moreover, this interval could be refined to be either assumed as a uniform distribution or by any other discrete or continuous distribution. However, in our work, we only assume upper and lower bounds on such non-functional properties.

The result of a *-predictability analysis is a Pareto-front of multiple implementations p_i of an invasive program under analysis, each being uncertain but bounded by an interval in its objectives as depicted in Figure 4.1. Implementations are characterised by intervals in the objective space. Implementations which intervals exceed upper and/or lower bounds violate requirements and are infeasible (implementations with red uncertainty intervals in the figure).

Invasive Programming for *-Predictability

Invasive computing propagates resource-aware programming. However, it might be difficult or even impossible for a programmer to specify by hand constraints on number and type of resources in order to achieve a desired *-predictability of program execution. For (automatic) analysis of complex programs, appropriate higher level abstractions on computation and communication are necessary for performance analysis and optimisation. In [Rol+16], Project C2, Project C4, and Project A1 have presented *actorX10*, an X10 library of a formally specified actor model based on the APGAS principles. The realised actor model explicitly exposes communication paths and decouples these from the control

```
1   public static def main(args:Rail[String]) {
2   /* Declare actors */
3   val v1 = new SourceActor("v1");
4   val v2 = new Filter1_1Actor("v2");
5   val v3 = new Filter2Actor("v3");
6   // ...
7
8   /* Declare actor graph with requirements */
9   // Performance Requirements
10  @REQUIRE("ag", new Latency(0,110,"ms","hard"))
11  @REQUIRE("ag", new Throughput(20,40,"fps","soft"))
12  // Reliability Requirement
13  @REQUIRE("ag", new PFH(0.001, 0.0000001))
14  // Power Requirement
15  @REQUIRE("ag", new Power(1,2,"W","soft"))
16  // Security Requirement
17  @REQUIRE("ag", new Confidentiality(50))
18  val ag = new ActorGraph("ag");
19
20  // Add actors and connect them
21  ag.addActor(v1);
22  ag.addActor(v2);
23  // ...
24  ag.connectPorts(v1.outPort1, v2.in);
25  ag.connectPorts(v1.outPort2, v3.in);
26  ag.connectPorts(v2.out, v4.in);
27  // ...
28
29  /* This statement is replaced by the design flow */
30  ag.start();
31 }
```

Figure 4.2: Example of an actor graph generation and execution in actorX10 as well as annotations regarding requirements on objectives (from [Wil+16]).

flow of the concurrently executed application components. Furthermore, we have enriched the invasive programming language with the option to specify so-called *requirements* on objectives of execution to express allowable uncertainty intervals rather than specifying constraints on resources [Wil+16].

Figure 4.2 shows an example of how to construct and execute an actor graph in actorX10, as well on how to specify requirements on non-functional execution properties by means of the **@REQUIRE** annotation. The figure illustrates requirements regarding lower and upper bounds on end-to-end latency and throughput (lines 10 and 11), intervals on the allowed probability of failures per hour (line 13), and on power

consumption (line 15), as well as a security requirement (line 17). Soft requirements should be satisfied but infrequent violations are tolerated. Whereas, hard requirements must never be violated.

A1

Hybrid Application Mapping

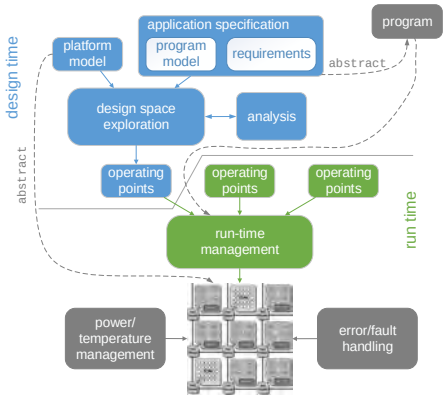


Figure 4.3: Overview of the methodology for predictable hybrid application mapping of applications with predictable timing and other non-functional requirements onto tile-based MPSoCs (from [Wil+16]).

For predictable multicore program execution, we propose a design flow which automatically determines resource constraints that will fulfil a set of given requirements. Subsequently, the requirement pragmas in the X10 source code are automatically replaced by the respective set of claim constraints. The flow implements a *hybrid application mapping (HAM)* approach for achieving run-time predictability by combining design-time analysis of application mappings with their run-time management, see Figure 4.3. In an invasive architecture, constituting an MPSoC of heterogeneous compute tiles connected by a network-on-chip (NoC), various mappings onto different resource constellations are possible. The idea of HAM is to identify those constellations that fulfil given timing constraints and possible other user requirements by applying a design space exploration (DSE) of resource allocations (resource claims) and task mappings. Each such constellation is called an *operating point* of the program, representing a program implementation with intervals $[\inf_o(p,Q,I), \sup_o(p,Q,I)]$ not exceeding the bounds specified in the user requirements.

We have elaborated different analysis techniques: A first approach for timing analysis of invasive architectures is presented in [Wil+16]. Analysis for enhancing security, particularly by avoiding side-channels on shared resources, is presented in [Wei+16; Dre+16]. Furthermore, [WT16; Lar+16] present HAM techniques for increasing fault tolerance on invasive network-on-chip architectures. Finally, analysis can also be performed by evaluation via simulation [Tei+16]. However, in this case, only average case results can be analysed.

The mapping constellation information explored in DSE is then used by the run-time management to find a concrete application mapping at run time. Each time an application with *-predictability requirements is activated, the run-time management checks whether a feasible application mapping exists that corresponds to one of the operating points with verified requirement bounds. Only in this case, the application is started according to the resource constellation corresponding to that point.

Combat Demonstration

We invested a lot of manpower into advancing the stability of our platform, to demonstrate the benefits of invasive computing in a full-stack scenario. In cooperation with Project C1 and other projects, several fixes were developed for the compiler runtime, agent system, and the operating system. In September, we demonstrated for the first time a realistic *combat scenario* using our full software stack to our industry board.

The visualisation in Figure 4.4 shows one moment in time during such a combat. The two applications are “Multigrid” and “Integrate”. In the figure scenario Integrate is started twice to enforce a competition, since the architecture is big enough to satisfy one Multigrid and one Integrate completely. Multigrid is a heat dissipation simulation of a laser engraving a metal plate. It uses a multigrid solver for the linear equations involved. The multigrid approach implies a v-cycle, where periodically fewer resources are required. Integrate computes an integral numerically using a *job-queue framework*. The hardest part about this is the load balancing, since it is unpredictable, which parts of the function are steeper, which requires more precise evaluation. We developed a generic job-queue framework, which is used here. It does load balancing within tiles and across tiles.

For the visualisation of the resource needs over the whole execution time, see Figure 4.5. Both applications have a very dynamic resource need. In the environment, there were enough resources for both at all

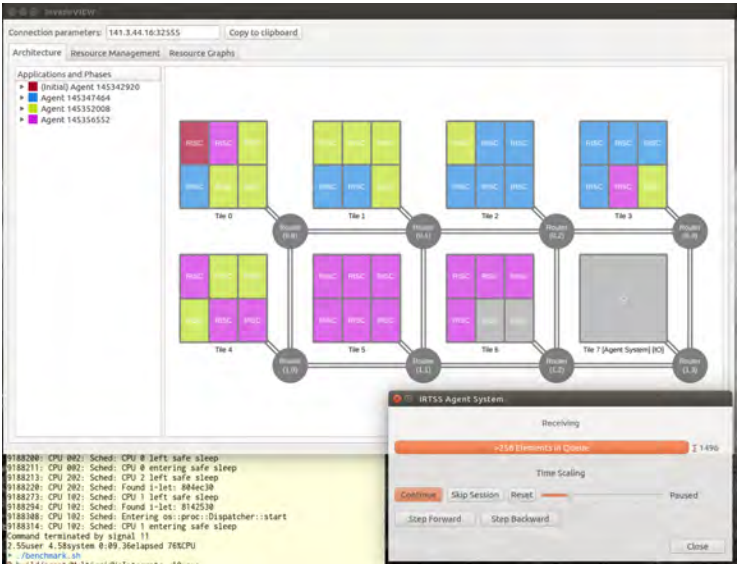


Figure 4.4: This image shows a graphical representation of an invasive MPSoC architecture, with 8 tiles, 6 cores each (except one I/O tile in the bottom left). Three applications (blue, pink and green) bargain about the resources. Tile 5 is completely owned by the pink application, while the others are shared. The red core in tile 0 is the "main" application, which spawns the other three.[1]

times. By adding a third application or using a smaller architecture, an competition on resources arises.

The next steps are to demonstrate on not just the software stack, which runs on a Linux host system. We want to exploit the features of the invasive hardware projects. Unfortunately, the reliability of a house cards exponentially decreases when you add more layers. Likewise, various errors in hard- and software currently prevent us from benchmarking the whole stack.

X10 Memory Model

As a foundation for further formalisation we developed a memory model for the X10 programming language [Zwi16]. The memory model is an important part of the specification for the compiler, which must correctly map the language's memory model to the target architecture's memory model.

[1] Visualisation is based on tool InvadeVIEW developed and provided by Project C2.

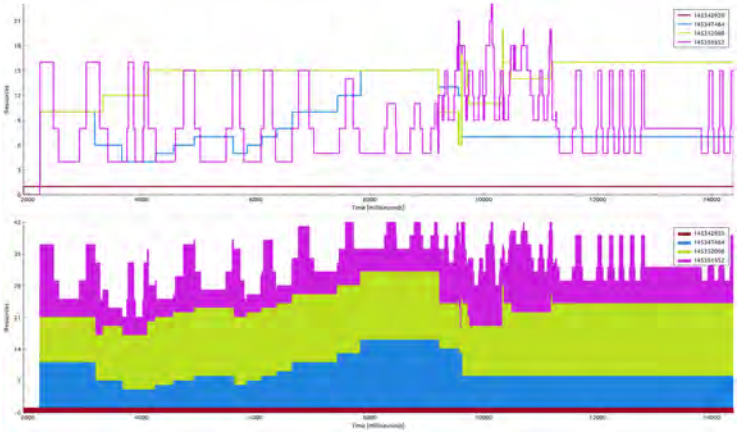


Figure 4.5: A resource plot of an execution like in Figure 4.4. The upper plot shows the cores used per application, the lower plot is the same in a stacked format. The Multigrid application (pink) features a cyclic resource need, while the Integrate applications (blue and green) has an unpredictable changing resource need. The period of Multigrid changes depending an the amount of available resources and how its cores are distributed across the architecture. More tiles with the same total amount of cores requires more communication and time.

Since X10 is a language initially designed for High-Performance Computing, performance has a high priority for its design. Therefore, the memory model must not burden the compiler with restrictions, such that fast execution becomes virtually impossible. On the other hand, safety is also important, so users can trust the results of the computations. There are three back ends for the compiler: Java, C++, and assembly. The C++ mapping requires two noteworthy design choices: a) data races are undefined behaviour and b) termination can be assumed. If the X10 memory model would define the behaviour for data races (like the Java memory model does), the compiler would have to remove all data races, when compiling to C++. This removal would significantly hurt performance, since many barrier instructions would have be inserted. Likewise, C++ assumes that non-trivial loops terminate, so X10 must do as well. An example is shown in Figure 4.6.

The X10 memory model we developed is simpler than the Java and the C++ memory model, thanks to the design of X10. For example, X10 has no object finaliser, which arguably were a mistake in Java. Also, there are no threads to join and no reflection in X10, compared to Java. C++ requires to elaborate the model to provide relaxed operations of atomics. This might eventually be desirable for X10 to improve

```
1 def foo(y:int,n:int):void {
2   var x:int = 0;
3     while (x < y) { x += n; }
4     }
```

Figure 4.6: We store the local variables x, y, n in registers, so there is no memory access within the loop. During the execution there is no "action" (see below) with respect to the memory model, so we consider the loop empty. Additionally, x is not used after the loop, so we do not care about its value. We cannot guarantee termination, since n might be zero. Still, the compiler can remove the loop.

performance, but is not available in the current version.

Figure 4.7 gives an impression of the memory model. It shows an activity being created and finishing, which corresponds to thread creation and termination in Java/C++. The arrows are "happens before" relations, of which there are two kinds. Program order for intra-activity relations and synchronises-with for inter-activity relations. The memory model is built on traces of execution, which means only one specific interleaving is shown here, where load of x happens after 42 is stored.

We also discovered that the `@Volatile` annotation and the `Fences` class have broken semantics and/or behaviour. Our proposal to remove them has been discussed with the language authors.

X10 is an APGAS language and provides features explicitly for distributed computing. For the memory model, this was not relevant. Since the only mechanism for distributed communication `at` is subject to program order, so there is no additional ordering necessary within the memory model. We only identified one case, where it can matter, which is the `Rail.asyncCopy` method. However, such exceptional cases should be handled in the method documentation, but not in the core memory model. This specific method uses internal runtime features, which are not accessible to application code.

Abstract Core Calculus

Our work on the abstract core calculus (WP A1.4) has been delayed, to provide additional man power to the development of the integrated demonstration platform partially shown in the visualization above.

A variation of this topic is developed in the dissertation of Andreas Zwinkau, but not yet published. Instead of a calculus with precise semantics, he proposes a model, which specifies valid behaviour of resource management according to resource constraints. This approach

```
1 finish {
2   async {
3     atomic { r = x; }
4   }
5   atomic { x = 42; }
6 }
```

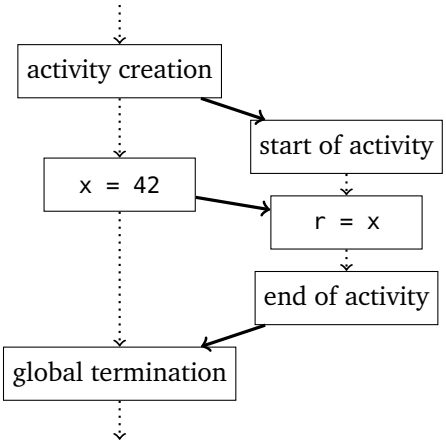


Figure 4.7: An example execution of an activity life cycle as written in the code on the left. Each box is an action. The thick arrows are synchronise-with edges and the dotted arrows show program order. The figure demonstrates the difference between "activity creation" and "start of activity" actions.

avoids the problem that a calculus must specify the behaviour of the agent system.

Invasive Programming Patterns

Last year, we worked on the job queue pattern. The actor pattern, described above, became the focus now. Since implementation is not yet finished, we will continue with this work package in 2017 although the working plan allocates no resources there. This is possible, since we finished the memory model work package early.

Publications

[Dre+16] G. Drescher, C. Erhardt, F. Freiling, J. Götzfried, D. Lohmann, P. Maene, T. Müller, I. Verbauwhede, A. Weichslgartner, and S. Wildermann. "Providing Security on Demand Using Invasive Computing". In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 281–295. DOI: 10.1515/itit-2016-0032.

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. “Providing Fault Tolerance Through Invasive Computing”. In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[Rol+16] S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. “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.

[Tei+16] J. Teich, M. Glaß, S. Roloff, W. Schröder-Preikschat, G. Snelting, A. Weichslgartner, and S. Wildermann. “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.

[WT16] A. Weichslgartner and J. Teich. “Position Paper: Towards Redundant Communication through Hybrid Application Mapping”. In: *Proceedings of the third International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: IEEE, Apr. 4, 2016, 4 pp.

[Wei+16] A. Weichslgartner, S. Wildermann, J. Götzfried, F. Freiling, M. Glaß, and J. Teich. “Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs”. In: *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*. (Sankt Goar, Germany). ACM, May 23, 2016, pp. 153–162. DOI: 10.1145/2906363.2906370.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

[Zwi16] A. Zwinkau. “An X10 Memory Model”. In: *Proceedings of the sixth ACM SIGPLAN X10 Workshop*. X10 ’16. June 2016.

A4: Design-Time Characterisation and Analysis of Invasive Algorithmic Patterns

Michael Glaß, Michael Bader

Tobias Schwarzer, Behnaz Pourmohseni, Alexander Pöppl

Project A4 investigates existing, develops novel, and analyses invasive algorithmic patterns w. r. t. diverse qualities of execution to exploit the resource awareness of invasive computing. The research focuses on (a) stencil computation and non-regular tree traversals as invasive algorithmic patterns inspired by the invasive applications from Projects D1 and D3 and (b) design-time characterisation techniques for the derivation of sets of optimised and diverse operating points by considering symmetries and system services of a given heterogeneous invasive many-core architecture. Figure 4.8 depicts the core topics and design flow investigated in Project A4.

In 2016, Project A4 focussed on two main topics, which are detailed in the following sections: (I) An extension of the X10 language to support actor-based design was developed together with Project C2 and Project A1, and a respective version of the shallow water waves simulation was developed for actorX10. (II) A novel technique for the distillation of operating points, i. e. the selection of a subset of the operating points delivered by the design-time characterisation to be passed to the *i*RTSS.

SWE-X10 – An actor-based proxy application

SWE-X10 is a proxy application for the simulation of shallow water waves [PB16; PBSG16]. In collaboration with Project A1 and Project C2, we developed actorX10 [Rol+16], an actor library that enables the creation of actor-based applications with a finite-state-machine-based control flow. Our earlier task-based version of SWE-X10 was amended to use actorX10, and the semantics of the execution was changed to resemble the semantics of the finite state machine (FSM). We demonstrated that FSMs, as part of the actor-oriented parallel programming model, can be used to identify parts of the computational domain that are not updated: following a *lazy activation* paradigm, respective com-

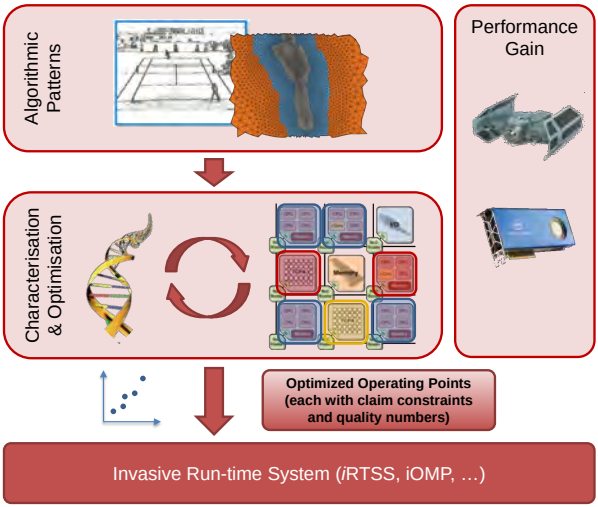


Figure 4.8: Project A4 in a nutshell: The co-design of invasive algorithmic patterns and a design-time characterisation as novel contribution to invasive computing shall result in performance gains and enable predictability by providing statically analysed operating points (tuples of claim constraints and quality numbers) to the invasive run-time system for dynamic resource allocation.

pute resources could be invaded only when they are actually needed. The actor approach also allows Cartesian grid patches to propagate in time without global control.

As a key feature, the actor approach prepares the use of methods for design-time characterisation: using analysis techniques or short sample simulations to evaluate performance numbers of simulation phases, this approach not only allows for optimisation of various parameters such as patch sizes, distributions, or load balancing, but also for exploring trade-offs between performance, the number of used components, energy consumption etc.

Lazy Activation via actorX10 To realise actor-based lazy activation, we model the communication between patches of the discretisation grid via actors that communicate via channels – Figure 4.9 shows such an actor graph with 3×3 actors, illustrating all channels and their capacities.

With lazy activation, simulation patches will remain in an initial steady state until the propagating wave enters the respective patch. We avoid superfluous computations on such patches by assigning each actor an attribute that stores its activity status. Initially, the status of each

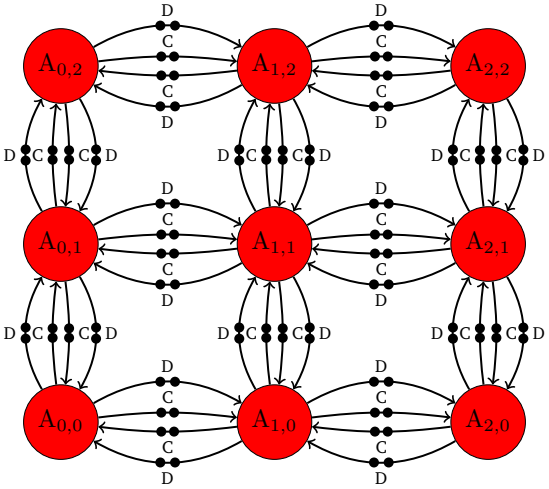


Figure 4.9: Actor graph for a simulation with 3×3 actors/patches. Each actor (node of the graph) controls a patch of the simulations domain (compare Figure 4.10). Each edge in the graph represents a channel between actors. The number of black circles on a channel reflects its capacity. The data type is denoted by C (control information) and D (ghost layer data).

actor is set based on the scenario's initial condition – usually only a few patches will be activated from the start. Figure 4.10 shows four snapshots from a simulation run with enabled lazy activation, where more and more patches are activated as the simulation progresses.

Figure 4.11 illustrates the FSM-based coordination. As a first step, all actors in the simulation send their activity status to their neighbours using the control channels (sendActivation()). Actors with an initial perturbation of the water level are set to the state *propagating-wave*, while the rest is set to *lake-at-rest*. Then, active actors perform the simulation steps (computeStep()) once they receive all necessary updates ($recvActive()$). During computation of the update, the patch determines for each of its copy layers whether the update actually changes any values. If changes occurred or if the neighbouring actor is already active, the updated layer will be sent. If the neighbour is still inactive, the actor will also send a control message stating that updates are now available (sendActivation()). Upon receiving the message ($recvActivation()$), a neighbour in the *lake-at-rest* state will set itself to *propagating-wave* and announce its new activity status to all neighbours (sendStatus()). Finally, once an actor has reached the termination condition, it will

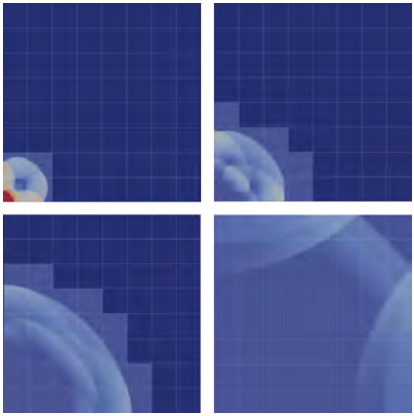


Figure 4.10: Simulation run with 8×8 actors using lazy activation. As the wave propagates (starting in the lower-left corner), more and more actors become active. Dark blue patches represent *lazy* patches; lighter patches are active.

send a termination signal (sendTerm()) to other actors, which will be propagated until no more actors are active ($recvTerm()$).

Benefit due to lazy activation We evaluated the benefit of lazy activation of patches via the actor approach by determining the accumulated CPU time ('CPU hours'). Assuming that inactive actors could remain in an idle state, we accumulated the total time spent for *useful* computing on each CPU. For a scenario similar to that in Figure 4.10 on 8 CPUs, we found significant reduction in CPU hours (approx. 55 % compared to without lazy activation). Turning this theoretical gain in CPU hours into actual reduced execution time (or reduced amount of resources in general) will require invasive techniques, for example migration of actors based on the computation of best-possible operation points.

Outlook on Adaptivity and Local Time Stepping Our goal for the project is to extend the approach to block-adaptive meshes with local time stepping, as described by LeVeque[2]. Here, fine-level meshes would overlay with coarse-level meshes, would only be activated on demand and be controlled by the actors' finite state machines, triggering load-balancing or migration based on a performance characterisation.

[2] R. LeVeque, D. George, and M. Berger. "Tsunami modelling with adaptively refined finite volume methods". In: *Acta Numerica* 20 (May 2011), pp. 211–289. DOI: 10.1017/S0962492911000043.

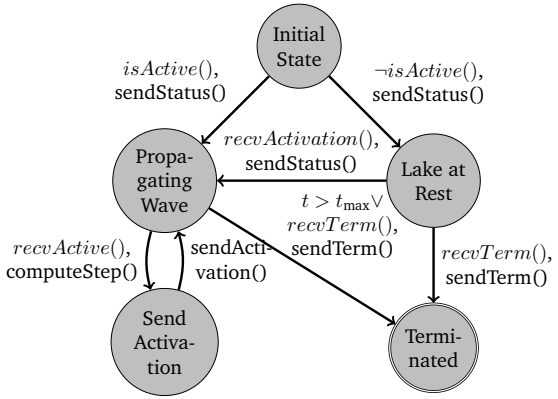


Figure 4.11: Finite State machine for the simulation actor. Methods written in italics are guard functions, and functions written in normal letters are actions.

Local time stepping will exploit the FSM semantics provided by the actor model. For example, actors communicating with neighbouring actors running at only half of the time step size (e. g. due to higher resolution) might use two separate states to deal with ghost layer data: one state for steps where a communication is needed and another state when the previously received data needs to be interpolated.

Dynamic adaptivity will especially make load balancing more complicated. Actors with a highly refined grid will have a significantly higher computational load. Hence, we will either need to allow splitting of actors, and allow actors to have more than one neighbour in each direction. Or we could make the assignment of actors to patches and cores more flexible and, for example, assign several cores (maybe even on accelerators) to the computation of refined patches.

Operating Point Distillation

Given multiple *quality objectives* (e. g. performance, energy efficiency, or reliability) and a number of *resource objectives* (i. e. number of required processing resource of each type), the DSE (Design Space Exploration) used in *hybrid application mapping* (HAM) techniques typically delivers a substantially large set of operating points, handling of which may impose an intolerable overhead to the RM (Run-time Manager). Therefore, it is necessary to reduce the number of the points before providing them to the RM, a reduction process Project A4 calls *operating point distillation*.

In [PGT17], we proposed an automatic operating point distillation

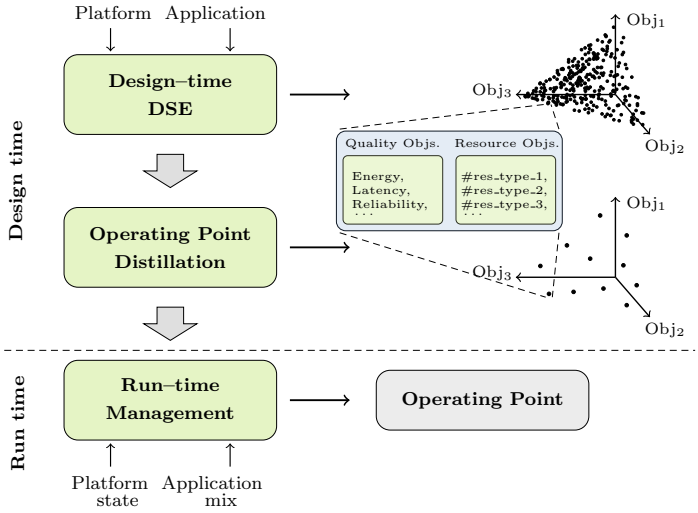


Figure 4.12: Integration of a design-time operating point distillation step in the standard hybrid application mapping flow.

approach that can be seamlessly integrated into the standard hybrid application mapping flow as illustrated in Figure 4.12. Given a set of Pareto-optimal operating points in the space of quality objectives and resource objectives (i. e. one objective per resource type), the proposed mechanism distils a subset—configurable in size—of operating points. It is distilled in such a way that (1) a *diverse* set of trade-off alternatives in the quality objectives is delivered to meet the—at design time unknown—run-time quality requirements, while (2) distilled points exhibit *modest* yet *diverse* resource requirements to enhance the *embeddability* of the application in view of a dynamic run-time resource availability. To achieve this, we separated the quality objectives from the resource objectives and employed a novel two-level distillation mechanism as follows: In the first step, the space of quality objectives is subdivided into regions of comparable quality trade-offs. To obtain the regions, an exponentially-spaced *hyper-grid* is placed in the normalised space of quality objectives where the number of divisions per objective is adaptive to the intended number of operating points and each grid cell represents one quality region. This is illustrated in Figure 4.13 (left). In the second step, from each quality region with efficient quality trade-offs (highlighted cells in Figure 4.13), one representative operating point

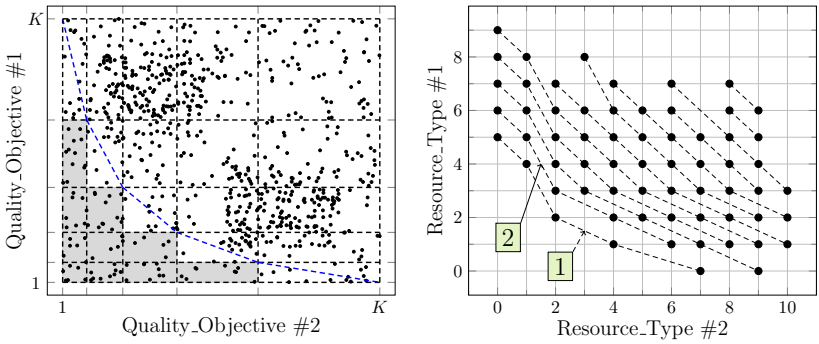


Figure 4.13: (Left) Quality space subdivision with 5 divisions per objective in the space of two quality objectives. Highlighted cells exhibit efficient quality trade-offs. (Right) Operating point Pareto-ranking in the space of two resource objectives where points belonging to the same cluster (with the same ranking) are connected with dashed lines.

is selected based on its resource requirements to enhance the run-time embeddability. For this purpose, we Pareto-rank the operating points in the space of resource objectives, as illustrated in Figure 4.13 (right), and exploit the ranking in distillation of representative points.

Experimental evaluations show that, compared to three standard multi-objective truncation techniques, the proposed distillation mechanism improves the embedding success rate by 8 % up to 45 % and the embeddability rate by 16 % up to 45 %[3]. Furthermore, it delivers a higher diversity in the quality trade-offs of the distilled points, indicated by an ϵ-dominance improvement of 35 % up to 42 % in the quality objectives, compared to the other approaches.

Outlook on operating point transition As the next step, we will focus on design-time investigation of run-time *operating point transition*. For this purpose, additional evaluation techniques will be investigated to compare the operating points obtained from the DSE with respect to the expected costs of run-time transition. The respective costs shall be back-annotated, e. g. to edges between operating points.

[3] Given a set of available platform resources and a set of distilled operating points, success rate indicates whether *at least one* of the points can be feasibly embedded, whereas embeddability rate denotes the *proportion* of the embeddable points in the set.

Publications

[PB16] A. Pöppl and M. Bader. “SWE-X10: An Actor-based and Locally Coordinated Solver for the Shallow Water Equations”. In: *Proceedings of the 6th ACM SIGPLAN Workshop on X10*. (Santa Barbara, CA, USA). X10 2016. ACM, 2016, pp. 30–31. DOI: 10.1145/2931028.2931034.

[PBSG16] A. Pöppl, M. Bader, T. Schwarzer, and M. Glaß. “SWE-X10: Simulating shallow water waves with lazy activation of patches using ActorX10”. In: *Proceedings of the 2nd International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*. IEEE, Nov. 2016, pp. 32–39. URL: http://conferences.computer.org/espm2/2016/papers/3858a032.pdf.

[PGT17] B. Pourmohseni, M. Glaß, and J. Teich. “Automatic Operating Point Distillation for Hybrid Mapping Methodologies”. In: *Proceedings of Design, Automation and Test in Europe Conference Exhibition (DATE)*. Lausanne, Switzerland: IEEE, 2017. forthcoming.

[Rol+16] S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. “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.

[Tei+16] J. Teich, M. Glaß, S. Roloff, W. Schröder-Preikschat, G. Snelting, A. Weichslgartner, and S. Wildermann. “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.

[Wei+16] A. Weichslgartner, S. Wildermann, J. Götzfried, F. Freiling, M. Glaß, and J. Teich. “Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs”. In: *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*. (Sankt Goar, Germany). ACM, May 23, 2016, pp. 153–162. DOI: 10.1145/2906363.2906370.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

B1: Adaptive Application-Specific Invasive Microarchitecture

Jörg Henkel, Jürgen Becker, Lars Bauer

Marvin Damschen, Tanja Harbaum, Srinivas Rao Kerekare, Carsten Tradowsky

Project B1 investigates mechanisms that provide run-time adaptivity: in the microarchitecture (μArch) and by using a run-time–reconfigurable fabric. We propose concepts and methods that allow invading the reconfigurable fabric and μArch within the invasive core (i-Core). The i-Core is an integral part of the InvasIC hardware. Therefore, we integrated our i-Core prototype as well as support hardware into the new demonstration platform (proFPGA) and provided a tool flow for partial run-time reconfiguration. In the following, we briefly describe our recent activities of the second funding phase and results on dynamic cache architectures, Auto-SI, predictability and the InvasIC hardware prototype.

Dynamic Cache Architecture

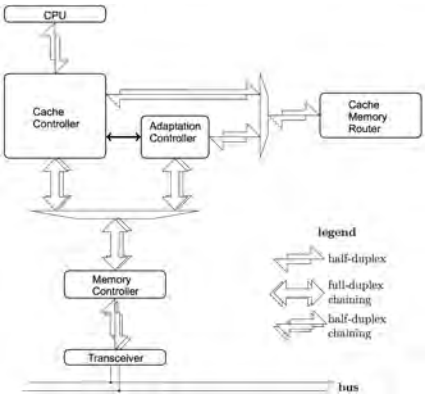


Figure 4.14: Overview of the dynamic cache Architecture

In future, most computing systems will evolve to heterogeneous processing systems with several diverse processing units and multiple applications will run concurrently. Therefore, these applications compete for computational resources and thus processing power. Especially in terms of caches, it is necessary to use all available resources efficiently, because the bandwidth and the available resources strongly bound computation times. For example, if streaming-based algorithms run

concurrently with block-based computations, then this could lead to an inefficient allocation of cache resources.

To tackle these challenges, we developed a dynamic cache architecture that enables the parametrisation and the resource allocation of cache memory resources between cores during run-time. Figure 4.14 shows the structure of the implemented design.

B1

The design is carefully weighted and a method with as little overhead as possible is chosen without degrading the performance of the cache architecture to be applicable for the strict timing constraints of an L1 cache. The L1 cache design is integrated into the CPU microarchitecture and is evaluated on reconfigurable hardware. In our evaluation we showed that already a slight hardware overhead of less than 10% enables our dynamic run-time cache architecture [Tra+16b]. In addition, we showed that it is possible to achieve an optimally-utilised cache memory tile according to the used on-chip memory target architecture.

Non-Cache-Coherent architectures for Manycore systems

In the case of manycore systems the cache coherence is often only ensured within single tiles. In cooperation with Project C3 a new technique to transfer object-oriented data structures on non-cache-coherent shared memory systems has been designed. The novel cloning approach avoids serialisation by managing cache coherence in software at object granularity. A compiler-assisted implementation for PGAS languages has been implemented, which runs fully automatic, save and has zero overhead. The experimental results using a distributed-kernel benchmark suite show that using our technique reduces communication time by up to 39.8%. Additionally, the results show that cache operations on address ranges are desirable on non-cache-coherent architectures. An overhead of 15% additional hardware resources can extend an existing cache controller with an efficient implementation of non-blocking range-based cache operations [MT17].

Auto-SI

Nowadays modern computer systems demand more and more performance without increasing power dissipation or chip area too much. To achieve this, we propose to speed up loops and to load miscellaneous accelerators during run-time. Several steps are necessary to accelerate a loop transparently, dynamically, and automatically: i) monitor instructions, ii) prepare configuration, iii) configure hardware accelerator, iv) use accelerator on *i*-Core fabric. The design of the *i*-Core with its recon-

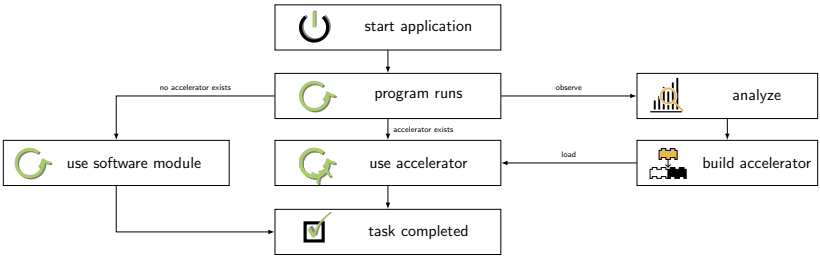


Figure 4.15: Work flow Auto-SI

figurable fabric has been expanded. Figure 4.15 shows the expanded work flow. If there is no Special Instruction provided for the *i*-Core, a suitable hardware accelerator will be loaded during run-time. The approach of Auto-SI has been prototyped on a Xilinx Virtex-7 platform.

Timing Analysis on *i*-Core Tiles

In the last years of InvasIC we demonstrated that the inherent adaptivity of applications executing on the *i*-Core provides remarkable average-case performance improvements. This year, a focus of our work was to make the performance benefits of the *i*-Core available for predictable execution on the InvasIC hardware (one of the main topics of the second funding phase). To achieve this, we developed models for static timing analysis of applications running on the *i*-Core that utilise run-time reconfiguration of SIs [DBH17; Wil+16].

Static timing analysis is performed on the control flow graph (CFG) of the application binary and aims to provide a precise upper bound of the worst-case execution time (WCET). In our model, reconfiguration of SIs for speeding up an upcoming kernel is initiated in a basic block immediately before entering the kernel. This basic block contains commands for a reconfiguration controller. Analysing these commands yields the reconfiguration delay per SI, i. e. the time it takes until an SI can be executed in hardware. A simple technique to perform analysable reconfiguration is to stall execution for the whole reconfiguration delay of all SIs and only then enter the kernel. However, this slow but analysable *stalling* affects the performance, as the reconfiguration delay of an SI is in the range of milliseconds. Instead of stalling, the pipeline of the *i*-Core can be used to execute functionally-equivalent software for each SI during the reconfiguration. We achieve this by using a conditional branch that either executes the SI on the reconfigurable fabric (if

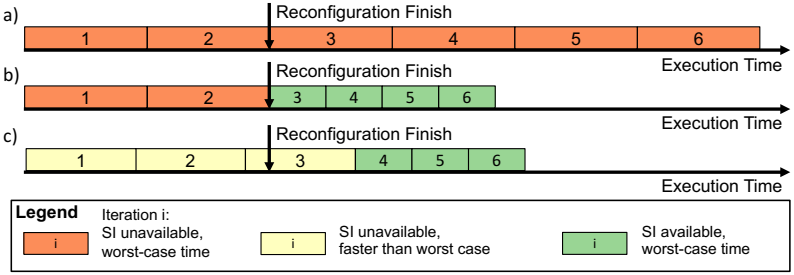


Figure 4.16: Timing analysis of a kernel with parallel reconfiguration of an SI.

reconfiguration has finished) or software on the pipeline. This enables *parallel reconfiguration*.

A safe, but imprecise kernel execution time bound for parallel reconfiguration can be obtained by assuming that every time an SI should be executed, execution branches to the unaccelerated software (see Figure 4.16 a)). While the actual execution time would benefit from the hardware accelerators, the guaranteeable WCET bound would still be as high as not using SIs at all. The challenge in analysing parallel reconfiguration, is to statically determine the worst-case iteration i of the kernel at which the reconfiguration for an SI can safely be assumed to be completed. Naively, one could assume that the worst-case iteration i is obtained by executing all iterations during the reconfiguration in WCET (see Figure 4.16 b)). However as shown in Figure 4.16 c), executing these iterations slightly faster than in WCET can lead to a longer overall execution time, e. g. because the SI in iteration 3 is executed just before the reconfiguration finishes and thus one more iteration is executed without hardware acceleration (such an effect is also known as a *timing anomaly*).

We propose models that enable estimation of precise upper bounds for execution times of tasks that use run-time reconfiguration in presence of the timing anomaly [DBH17]. Figure 4.17 shows how stalling compares to parallel reconfiguration in the guaranteed WCET (obtained using our models) and maximum observable execution time with worst-case input (simulated) of the Loop Filter Kernel from the H.264 Encoder. The results were obtained with a frequency of the reconfigurable fabric f_{fabric} of 100 MHz and several values for the CPU frequency f_{CPU} between 100 and 1600 MHz ($f_{\text{CPU}}/f_{fabric} \in [1:16]$). We could show that parallel reconfiguration is always considerably faster than stalling

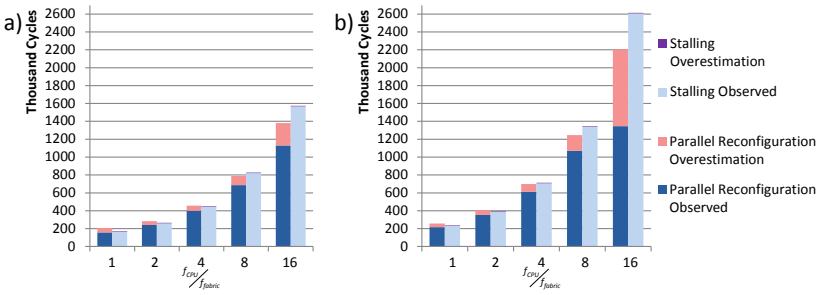


Figure 4.17: WCET results for analysing an H.264 encoder that utilises run-time reconfiguration on the *i*-Core using our proposed models. Results in a) were obtained using a reconfiguration bandwidth of 200 MB/s, results in b) using 100 MB/s.

in the maximum observable execution time. The guaranteed WCET is also lower for parallel reconfiguration despite a higher overestimation than stalling, because the speedup outweighs the overestimation. Furthermore, we could show that – in addition to a considerable speedup – overestimation compared to execution without reconfigurable SIs is reduced. SIs typically implement functionality that corresponds to several hundred instructions on the CPU pipeline. While analysing instructions for worst-case latency may introduce overestimation, the latency of SIs – executed on the *i*-Core's reconfigurable fabric – is precisely known.

Prototyping and Integration

We are continuously extending and improving the *i*-Core prototype and integrate new features into the joint InvasIC hardware prototype. During the last months, we migrated the *i*-Core to the new version of the Gaisler IP Library[4], which was necessary to move to the new ProDesign proFPGA prototyping platform. We now provide a tool flow to obtain partial bitstreams and perform run-time reconfiguration for the InvasIC hardware prototype on the proFPGA system.

Next Steps in Plan

Together with Projects A4, C1, and C3 we are currently working on an *i*-Core-accelerated actorX10-based version of the 'Shallow Water Equations' application that will serve as a comprehensive example to demonstrate the benefits of InvasIC computing.

[4] `http://www.gaisler.com`

Furthermore, we are continuing our efforts to provide an adaptable as well as predictable processor architecture.

Publications

[DBH17] M. Damschen, L. Bauer, and J. Henkel. “Timing Analysis of Tasks on Runtime Reconfigurable Processors”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.1 (Jan. 2017), pp. 294–307. DOI: 10.1109/TVLSI.2016.2572304.

[GBH17] A. Grudnitsky, L. Bauer, and J. Henkel. “Efficient Partial Online Synthesis of Special Instructions for Reconfigurable Processors”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.2 (Feb. 2017), pp. 594–607. DOI: 10.1109/TVLSI.2016.2585603.

[MT17] M. Mohr and C. Tradowsky. “Pegasus: Efficient Data Transfers for PGAS Languages on Non-Cache-Coherent Many-Cores”. In: *Design, Automation Test in Europe Conference Exhibition (DATE)*. 2017. forthcoming.

[THMB16] C. Tradowsky, T. Harbaum, L. Masing, and J. Becker. “A Novel ADL-based Approach to Design Adaptive Application-Specific Processors”. In: *Best of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2016. forthcoming.

[Tra+16a] C. Tradowsky, E. Cordero, C. Orsinger, M. Vesper, and J. Becker. *A Dynamic Cache Architecture for Efficient Memory Resource Allocation in Many-Core Systems*. Cham: Springer International Publishing, 2016, pp. 343–351. DOI: 10.1007/978-3-319-30481-6_29.

[Tra+16b] C. Tradowsky, E. Cordero, C. Orsinger, M. Vesper, and J. Becker. *Adaptive Cache Structures*. Cham: Springer International Publishing, 2016, pp. 87–99. DOI: 10.1007/978-3-319-30695-7_7.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

B2: Invasive Tightly-coupled Processor Arrays

Jürgen Teich

Marcel Brand, Vahid Lari, Éricles Sousa, Frank Hannig

Project B2 investigates invasive computing on tightly-coupled processor arrays (TCPAs). These have been shown to provide a highly energy-efficient and, at the same time, timing-predictable acceleration for many computationally intensive applications that may be expressed by nested loops from diverse areas such as scientific computing, image and signal processing, to name a few.

In the first funding phase, concepts for hardware-controlled invasion through a cycle-wise propagation of invasion control signals between neighbouring processing elements (PEs) have been investigated. Such decentralised parallel invasion strategies may not only reduce the invasion overhead by two orders of magnitude w. r. t. a centralised software-based approach. Even bounds on the invasion time of invading N processing elements in $\mathcal{O}(N)$ clock cycles have been shown to be achievable. Moreover, the self-adaptive nature of invasive computing was also exploited for the purpose of dynamic power management by controlling the wake up as well as the power down of regions of processors directly by the invade and retreat signals, respectively [Lar15; Lar16a].

Within the last year, we continued our work on providing guarantees for multiple non-functional properties such as *fault tolerance* and *energy consumption* as the major focus of research for Project B2 in the current 2nd funding phase. In the following, we will describe our results obtained in these research areas.

Safe(r) Loops – Fault-Tolerant Parallel Loop Processing

The high integration density of future multicore systems will inevitably lead also to more and more vulnerability of the circuits to malfunction due to thermal effects, circuitry wear-outs, or cosmic radiation. However, instead of analysing error and fault effects on single cores, lifting well-

known fault tolerance schemes such as dual (DMR) and triple modular redundancy (TMR) to the level of loop programs and their parallel processing on multicores is our unique solution for supporting fault-tolerant loop execution. Here, we are investigating approaches in which based on application *requirements* on reliable execution, an invasive loop program may request to switch on and off fault tolerance schemes for error detection and/or correction of certain parts or a parallel loop application as a whole [Lar+16]. We have already shown in [Lar+15], how the regular structure of TCPAs does ideally offer an application to claim (a) a non-redundant, (b) a dual-replicated, or even (c) a triple-replicated array instance for computing the parallel program in lock-step mode. At the programming level, the usage of these mechanisms may be requested through adding fault tolerance constraints:

B2

```
1 constraints.add(new FaultTolerance(TMR))
```

Within a tight collaboration with Project C1 and Project C3, we have developed hardware/software signalling mechanisms to provide feedbacks on loop executions on TCPAs, e. g. execution failures, at the level of InvadeX10 programming [Lar+16]. Such feedbacks are provided as the return value of an infect request and describes whether the executions on the invaded TCPA has failed, e. g. due to a computation failure, and which component was influenced by faults. In order to realise such a capability, we have designed and integrated a so-called error handling unit (EHU) within each processing element (PE) that is capable of making majority votes (or comparison) over values within PE's register file (see Figure 4.18(a)). An EHU generates three outputs, a value containing the result of a vote/comparison operation, a one-bit error signal denoting whether a non-maskable error is detected, and an error diagnosis value that describes the source of the detected non-maskable error, e. g. multiple errors in different input replicas or an error in the EHU etc. As shown in Figure 4.18(b), the error and diagnosis outputs from different PEs are aggregated at the level of the processor array, constructing an error index denoting the index of the PE that has detected a non-maskable error, and an error diagnosis vector per PE row, describing the cause of the detected error. In case of a detected error, an interrupt is generated that leads to running an error handling service routine within the TCPA driver code that is executed on a local RISC processor. This is then communicated to the application level as a return value of an infect request which provides an application programmer means to error exception handling codes to adapt the employed fault tolerance mechanisms depending on the application execution

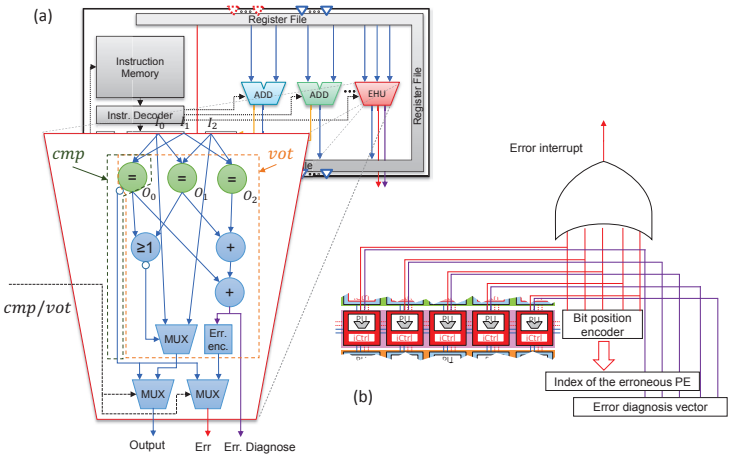


Figure 4.18: Implementation of comparison/voting instructions for DMR/TMR loop executions on TCPAs. Apart from a multiple cycle software implementation, a special functional unit called error handling unit (EHU) has been developed (shown highlighted on the left). (a) An EHU may be integrated as a new functional unit into each processing element. It votes (or compares) the content of the PE's register file (input/output ports or internal registers of the PE) and stores the result on its output port. Upon the detection of a non-maskable error, this unit generates an error signal as well as an error diagnosis value that describes the cause of the error. The corresponding instruction is implemented by a single cycle operation. The figure in (b) shows how these signals are aggregated at the PE array level, constructing error and diagnosis vectors per PE row. In case of detecting an error, an interrupt is raised that results in running an error handling service routine within the TCPA driver code.

feedbacks. An example of how an application programmer may write his/her own application-specific fault-handler is provided in the following. Here, based on a soft error rate (SER) estimated by a model such as CREME96[5] or observed using fault monitors, and also the number of detected non-maskable errors, suitable fault tolerance mechanisms are activated, i. e. dual or triple modular redundant executions.

```
1 var stop:boolean = false;
2 var errorCounter:int = 0;
3 var SER:double = 0;
4 var basicConstraints:Constraint = new AND();
5 basicConstraints.add(new TypeConstraint(PEType.TCPA));
6 basicConstraints.add(new PEQuantity(1, 8);
7
```

[5] A. J. Tylka et al. "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code". In: *IEEE Transactions on Nuclear Science* 44.6 (Dec. 1997), pp. 2150–2160. DOI: 10.1109/23.659030.

```
 8 while (!stop) {
 9   var adaptedConstraints:Constraint = basicConstraints;
10   model.update();
11   SER = model.softErrorRate();
12
13   if(errorCounter > highErrorCount){
14     stop = true;
15   } else if (errorCounter > minErrorCount){
16     adaptedConstraints.add(new FaultTolerance(TMR));
17   } else if (SER > minSER){
18     adaptedConstraints.add(new FaultTolerance(DMR));
19   }
20
21   if(!stop){
22     val claim = Claim.invade(adaptedConstraints);
23     val claimRet = claim.infect(ilet);
24     // Here, the application has
25     // terminated, either by completion
26     // or by a fault
27     if(claimRet.failed &&
28       claimRet.failureType == COMPUTATION_FAILURE){
29       errorCounter++;
30     }
31     claim.retreat();
32   }
33 }
```

In case of a low SER (smaller than a threshold `minSER`) or no detected errors on the claimed resources, an application program shall be executed without applying any fault tolerance mechanism; in case of an SER value higher than a minimum threshold (`minSER`), DMR shall be employed on the TCPA. When the number of detected errors has increased but is still within an acceptable range (between a threshold `minErrorCount` and a threshold `highErrorCount`), a higher protection is employed by running the codes on the TCPA in TMR mode. Otherwise, if the number of errors that occurred on the claim exceeds the acceptable range (bigger than a `highErrorCount`), the application execution on the claimed resources should terminate.

Ultra-Low Power/Dark Silicon

TCPAs have proven to not only offer high performance but high energy efficiency in running compute-intensive loop kernels as well compared to general purpose processors. In order to exploit this capability within heterogeneous multiprocessor system-on-chip (MPSoC) architecture, we have investigated techniques for:

Power density-aware resource management for heterogeneous tiled multicores In cooperation with Project B3, we investigated a power density-aware resource management for maximising the overall system performance on heterogeneous tiled multicores, where all cores or accelerators (i. e. TCPAs) inside a tile share the same voltage and frequency levels, without violating a predefined critical temperature [Khd+16]. Our technique considers power properties of applications as well as processors and consists of three steps defined as: (a) uniform power density constraints, (b) application assignment and mapping under power density constraints, and (c) runtime power density adaptation.

In the first step, the heterogeneity within a chip and heat transfer among cores and TCPAs are considered. A uniform power density constraint is derived that avoids any thermal violation. Second, our combined approach for massively parallel architectures takes into account the power density constraints and assigns applications to tiles and threads to cores to achieve a maximum throughput. In the last step, an adaptation scheme exploits available thermal headroom and reacts to workload changes at runtime. Here, if a power density budget assigned to a tile is underutilised, we can decrease these predefined values in order to increase the budget assigned to high power density cores/applications. Consequently, this will exploit any potential thermal headroom and maximise the utilisation of the chip cores.

In an experimental setup, a heterogeneous tiled architecture is considered. Each tile has one v/f domain and all cores and TCPAs inside a tile share the same v/f level. The TCPA power profiles were derived based on post-synthesis results from the Cadence Encounter RTL Compiler using a NanGate 45 nm low power process technology node. The accelerator can run at different frequencies from 50 to 650 MHz in steps of 50 MHz. Thus, according to the power budgets assigned to TCPA tiles, it is also possible to decide which PEs/regions of a TCPA have to become (a) dark (i. e. have to be turned off), (b) can operate at the highest clock frequency (white), or (c) will operate at a reduced voltage/frequency (gray).

Cross-layer approach for distributed dark-silicon management In cooperation with Project B1, Project B3, Project B4, Project B5, and Project C1, we presented a holistic sensing and optimisation flow for distributed dark-silicon management for tiled heterogeneous manycores under critical temperature constraints [Pag+16].

In this approach, applications are initially profiled at design time to obtain static power/performance and average/peak power consumption

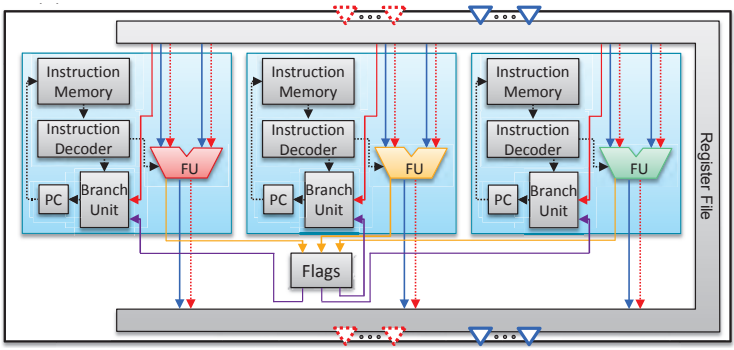


Figure 4.19: A processing element (PE) with an orthogonal instruction processing architecture including three functional units (FU). Each FU has a dedicated instruction memory along with an instruction decoder, a program counter and a branch unit. The FUs share input and output interfaces as well as the register file. Additionally, each FU can access the flags of all FUs. The figure depicts the data and control flows inside a PE by blue and red connections, respectively. The flag logic is depicted by yellow connections.

values for each application running on different types of processors. To afford runtime decisions, different monitors can be used for providing the status of the underlying hardware during application execution. Based on the status information (e. g. instantaneous power consumption, temperature, or reliability predictions), the dark-silicon management layer controls the power mode of the cores and TCPAs (active, idle, power-gated, etc.). Moreover, it decides the DVFS levels of the tiles and provides thermally safe partitioning and mapping alternatives or constraints to the agent layer, which negotiates resource allocation with respect to current system load and characteristic features of applications.

Orthogonal Instruction Processing

In general, system-on-chips such as TCPAs have only access to a limited amount of memory, which stands in contrast to the characteristic of VLIW compilers that typically produce lengthy code, especially when employing pipelining in loop programs. Therefore, we investigated a new processor architecture called *Orthogonal Instruction Processing* (OIP) that tackles this problem. This architecture is shown in Figure 4.19. Instead of a conventional VLIW processor that loads each cycle a very long instruction from a single instruction memory, each FU has its own instruction memory and branch unit. Only the register files as well as the flag distribution are shared among all FUs. We are currently

evaluating the hardware costs of this new PE architecture. We expect particularly good benefits in terms of instruction memory savings that dominate a potential small hardware overhead.

Outlook

As aforementioned, we are investigating novel architectures of TCPA processing elements using the principle of orthogonal instruction processing. In the future, we will analyse the overall cost and energy consumption of OIP.

In addition, a cost-optimised distributed interrupt architecture needs to be developed for signalling non-maskable errors. Also, for test purposes, we will extend our TCPA architecture with a fault injection module, capable of injecting temporal faults into the different registers of the processing elements. Finally, 2017 will be used to develop and showcase that invasive architectures in general and TCPAs in particular may provide *-predictability. Focus of our demonstrations will be hard real-time control applications such as the ones stemming from Project D1 (robot control).

Publications

[Khd+16] H. Khdr, S. Pagani, Éricles R. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores". In: *IEEE Transactions on Computers (TC)* (July 2016). DOI: 10.1109/TC.2016.2595560.

[Lar15] V. Lari. "Invasive Tightly Coupled Processor Arrays". Dissertation. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Nov. 18, 2015.

[Lar16a] V. Lari. *Invasive Tightly Coupled Processor Arrays*. Springer Singapore, 2016. DOI: 10.1007/978-981-10-1058-3.

[Lar16b] V. Lari. "Providing Fault Tolerance Through Invasive Computing". Talk at DTC 2016, The Munich Workshop on Design Technology Coupling, Munich, Germany. June 30, 2016.

[Lar+15] V. Lari, J. Teich, A. Tanase, M. Witterauf, F. Khosravi, and B. Meyer. "Techniques for On-Demand Structural Redundancy for Massively Parallel Processor Arrays". In: *Journal of Systems Architecture (JSA)* 61.10 (Nov. 2015), pp. 615–627. DOI: 10.1016/j.sysarc.2015.10.004.

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. “Providing Fault Tolerance Through Invasive Computing”. In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[Pag+16] S. Pagani, L. Bauer, Q. Chen, E. Glocker, F. Hannig, A. Herkersdorf, H. Khdr, A. Pathania, U. Schlichtmann, D. Schmitt-Landsiedel, M. Sagi, Éricles Sousa, P. Wagner, V. Wenzel, T. Wild, and J. Henkel. “Dark Silicon Management: An Integrated and Coordinated Cross-Layer Approach”. In: *it – Information Technology* 58.6 (Sept. 16, 2016), pp. 297–307. DOI: 10.1515/itit-2016-0028.

B3: Power-Efficient Invasive Loosely-Coupled MPSoCs

Andreas Herkersdorf, Jörg Henkel

Santiago Pagani, Mark Sagi, Anuj Pathania, Heba Khdr,
Muhammad Shafique, Philipp Wagner, Thomas Wild

The overall goal of Project B3 is to optimise power/energy efficiency considering the dark-silicon problem. Within the paradigm of invasive computing, the goal is to ensure that invaded *claims* remain thermally reliable while providing the *teams* for invading and executing *i*-lets for *infecting*. The pursued objectives are:

Objective 1: Improving power efficiency under dark-silicon constraints.

Objective 2: Developing an adaptive system for dark silicon and energy management.

Objective 3: Modelling and online estimation of dark silicon for invasive computing systems.

The related scientific challenges include the maximisation of the performance under given power density constraints or under a given energy budget, and minimising the energy consumption or peak power under a certain performance requirement. These goals require deep insight into the system. The intelligent collection and aggregation of individual data points from all over the SoC is another challenge that we address by means of powerful on-chip on-the-fly data analysis infrastructure.

Dark-Silicon Management for Performance Optimisation

The work in [Pag+16] presents an integrated and coordinated cross-layer sensing and optimisation flow for distributed dark-silicon management for tiled heterogeneous manycores under a critical temperature constraint, which summarises the main contributions of Project B3 and its interactions with Project B2, Project B4, and Project C1. In this work, we target some of the key challenges in dark silicon for manycores, such as: directly focusing on power density/temperature instead of considering simple per-chip power constraints, considering tiled heterogeneous architectures with different types of cores and accelerators, handling the

large volumes of raw sensor information, and maintaining scalability. Our solution is separated into three abstraction layers: a sensing layer (involving hardware monitors and preprocessing, i. e. Project B4 and Project B3), a dark-silicon layer (that derives thermally-safe mappings and voltage/frequency settings, i. e. Project B3), and an agent layer (used for selecting the parallelism of applications and thread-to-core mapping based on alternatives/constraints from the dark-silicon layer, i. e. Project C1).

In [Khd+16], we focus on maximising the overall system performance under a critical temperature constraint for heterogeneous clustered multicores, where all cores or accelerators inside a cluster share the same voltage/frequency levels. For such architectures, we present a resource management technique that introduces power density as system level constraint in order to avoid thermal violations. The proposed technique then assigns applications to clusters by choosing their degree of parallelism and the voltage/frequency levels of each cluster, such that the power density constraint is satisfied. Furthermore, our technique provides runtime adaptation of the power density constraint according to the characteristics of the executed applications, and reacting to workload changes at runtime. Therefore, the available thermal headroom is exploited in order to maximise the overall system performance.

Thermal Safe Power (TSP) is a novel power budgeting concept that provides safe power constraints as a function of the number of simultaneously active cores. In [Pag+17; Pag+16b], we extend the TSP concept from a power constraint to a power *density* constraint, which is suitable for heterogeneous systems with different types of cores. TSP is a fundamental new step and advancement towards dealing with the dark-silicon problem as it alleviates the pessimistic bounds of TDP and enables system designers and architects to explore new avenues for performance improvements in the dark-silicon era.

Some of these (and other) research efforts are also summarised in [Hen+16].

Dark-Silicon Management for Energy Optimisation

In [Tom+16], we propose a general-purpose middle-ware that reduces the energy consumption in embedded systems without violating the real-time constraints. The algorithms in our middle-ware adopt the computation offloading concept to reduce the workload on the processor of the embedded system by sending the most computation-intensive tasks to a powerful server. The algorithms are further combined with DVFS in order to find an execution frequency such that the energy

consumption is minimised while all real-time constraints are satisfied. Our evaluations show that our approach can reduce the average energy consumption down to nearly 60%, compared to executing all tasks locally at the maximum processor frequency.

In [Pag+16a], we focus on energy minimisation of periodic real-time tasks running on a heterogeneous multicore system clustered in multiple Voltage Frequency Islands (VFIs), where the power consumption and execution time of a task changes not only with DVFS, but also according to the task-to-island assignment. Our technique consists of the coordinated selection of the voltage/frequency levels for each island, together with a task partitioning scheme that considers the energy consumption of the task executing on different islands and at different frequencies, as well as the impact of the frequency and the core architecture on the resulting execution time. Every task is then mapped to the most energy-efficient island for the selected voltage/frequency levels, and to a core inside the island such that the workloads of the cores in the island are balanced.

Online Data Analysis to Support Dark-Silicon Management

Invasive Diagnosis on Chip (iDoC) extends the InvasIC run-time system with hardware components. These hardware components enable the observation of invasive software applications on-chip. We want to observe the application and platform metrics to identify potential run-time inefficiencies, especially power inefficiencies. Having identified the power inefficiencies, iDoC provides this inefficiency information to DaSiM. Exploiting this information, DaSiM optimises system behaviour to allow long-term power savings. Short term power savings are made possible by an in iDoC integrated low-latency hardware control loop. Performance requirements of the applications are ensured by both iDoC and DaSiM. To enable all these features, efficient and effective observations of the application behaviour are necessary. This is an interesting and novel challenge. Tremendous amounts of unprocessed data are available on the hardware level, such as CPU register contents, NoC flits etc. Potentially, all of this data might reveal power inefficiencies. However, evaluating *all* of this data *all* of the time would easily overwhelm DaSiM. Therefore, iDoC filters and post-processes the data to yield useful information for the control infrastructure on higher abstraction levels, i. e. DaSiM. Towards this goal, we created a flexible library of data analysis components. This components are coupled to a freely programmable general-purpose data analysis unit. Multiple data analysis components combined with the general-purpose data analysis unit build the iDoC infrastructure. Design-time knowledge of the system is joined

with run-time knowledge diagnosed by the data analysis components to be formalised by the data analysis unit. This formal knowledge is then sent to DaSiM. In [WWH16], we proposed a novel tracing system architecture. With this architecture, application developers can move from trace data streams towards self-contained diagnosis events. These diagnosis events can be used by developers to minimise energy inefficiencies or to debug the application. Furthermore, we presented novel ideas on application awareness and power management adaption in [SH16]. Based on these ideas, we plan to further extend the iDoC library with analysis components to enable application-aware DFS. These self-learning analysis components have the goal of automatically identifying critical application states and to boost the CPU frequencies for such critical states.

Dissertations

In November 24th 2016, Santiago Pagani received his Ph. D. (Dr.-Ing.) with *Auszeichnung* from the Faculty of Informatics, Karlsruhe Institute of Technology (KIT). The title of his dissertation is "Power, Energy, and Thermal Management for Clustered Manycores" [Pag16].

Organisation of Scientific Events and Public Dissemination

Project B3 organised an embedded tutorial on dark silicon titled "The Dark Silicon Problem: Technology to the Rescue?" at the IEEE/ACM 19th Design, Automation and Test in Europe Conference (DATE), 2016. More information at
https://www.date-conference.com/date16/conference/session/2.2.

Furthermore, Project B3 also collaborated on the Dagstuhl seminar "Dark Silicon: From Embedded to HPC Systems", 2016. More information is available at http://www.dagstuhl.de/16052.

Publications

[Hen+16] J. Henkel, S. Pagani, H. Khdr, F. Kriebel, S. Rehman, and M. Shafique. "Towards Performance and Reliability-Efficient Computing in the Dark Silicon Era". In: *Proceedings of the 19th Design, Automation and Test in Europe (DATE)*. Dresden, Germany, Mar. 14–18, 2016.

[Khd+16] H. Khdr, S. Pagani, Éricles R. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores". In: *IEEE Transactions on Computers (TC)* (July 2016). DOI: 10.1109/TC.2016.2595560.

[Pag16] S. Pagani. "Power, Energy, and Thermal Management for Clustered Manycores". Dissertation. Chair for Embedded Systems (CES), Department of Computer Science, Karlsruhe Institute of Technology (KIT), Germany, Nov. 24, 2016.

[Pag+16] S. Pagani, L. Bauer, Q. Chen, E. Glocker, F. Hannig, A. Herkersdorf, H. Khdr, A. Pathania, U. Schlichtmann, D. Schmitt-Landsiedel, M. Sagi, Éricles Sousa, P. Wagner, V. Wenzel, T. Wild, and J. Henkel. "Dark Silicon Management: An Integrated and Coordinated Cross-Layer Approach". In: *it – Information Technology* 58.6 (Sept. 16, 2016), pp. 297–307. DOI: 10.1515/itit-2016-0028.

[Pag+16a] S. Pagani, A. Pathania, M. Shafique, J.-J. Chen, and J. Henkel. "Energy Efficiency for Clustered Heterogeneous Multicores". In: *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2016).

[Pag+16b] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel. "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon". In: *IEEE Transactions on Computers (TC)* (2016). Feature Paper of the Month. DOI: 10.1109/TC.2016.2564969.

[Pag+17] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel. "Thermal Safe Power: Efficient Thermal-Aware Power Budgeting for Manycore Systems in Dark Silicon". In: *The Dark Side of Silicon*. Ed. by A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen. Springer, 2017.

[SH16] M. Sagi and A. Herkersdorf. "On-Chip Diagnosis of Multicore Platforms for Power Management". Workshop Presentation, DTC 2016, The Munich Workshop on Design Technology Coupling, Munich, Germany. June 30, 2016.

[Tom+16] A. Toma, S. Pagani, J.-J. Chen, W. Karl, and J. Henkel. "An Energy-Efficient Middleware for Computation Offloading in Real-Time Embedded Systems". In: *Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Daegu, South Korea, Aug. 17–19, 2016.

[WWH16] P. Wagner, T. Wild, and A. Herkersdorf. “DiaSys: On-Chip Trace Analysis for Multi-processor System-on-Chip”. In: *Architecture of Computing Systems (ARCS’16)*. Nuremberg, Germany: Springer, 2016.

B4: Hardware Monitoring System and Design Optimisation for Invasive Architectures

Doris Schmitt-Landsiedel, Ulf Schlichtmann

Qingqing Chen, Elisabeth Glocker, Shushanik Karapetyan,
Daniel Müller-Gritschneder, Bing Li, Cenk Yilmaz

In order to provide the invasive computing systems means to prevent or at least reduce the impact of hardware failures (including catastrophic failures as well as parametric failures such as not meeting frequency requirements or exceeding power limitations), the hardware monitoring system of Project B4 provides the system with monitoring data giving information about current hardware health. Permanent or temporal changes within the hardware can arise due to changes in the operating conditions such as supply voltage, frequency or temperature, due to increasing ageing, e. g. due to NBTI (Negative Bias Temperature Instability) or due to manufacturing variations.

Monitoring data include valuable information in order to support a flexible resource allocation adapting to these changes within the hardware and to foresee (based on current and past monitoring data) potential problems arising from these changes before they can have serious impact on the correct functioning of the system.

Project B4 is working on new monitoring concepts, especially regarding monitoring concepts for ageing, since the ageing of integrated circuits cannot be neglected in advanced process technologies.

With ageing monitors integrated into the hardware monitoring system, ageing behaviour can be analysed and ageing monitoring data can be distributed within the computing system. As for power and temperature monitoring, integration of ageing monitors into the FPGA demonstrator will be done via emulation of the behaviour of an ASIC ageing monitor and integration into the eTPMon framework. This approach gives the possibility to load and store the ageing status into the eTPMon framework easily. With that, a specific point in the system's lifetime could be loaded onto the demonstrator platform. Also timing pre-errors could be manually injected.

In [SHJL16] ageing effects are summarised and it is discussed how to

deal with them on different abstraction levels and on different levels in the design flow or to counter ageing by post-silicon tuning after design phase. Besides, an overview of the state-of-the-art is given, on how to apply techniques like on-chip timing margin monitoring and tuning, ageing analysis with high-level models and the use of flexible delay models of flip-flops in order to improve the permanent health of circuits.

[ZLS16] deals with post-silicon tuning: It proposes an efficient delay test framework (EffiTest) that uses already-existing tuning buffers in the circuit to align path delays in order to solve the post-silicon testing problem. Only representative paths are tested, since the delays of other paths are estimated by using statistical delay prediction.

In cooperation with Project B1, Project B2, Project B3, Project B5 and Project C1, in [Pag+16] an integrated and coordinated cross-layer approach on dark-silicon management with the novel hardware monitor approach from Project B4 is presented.

B4

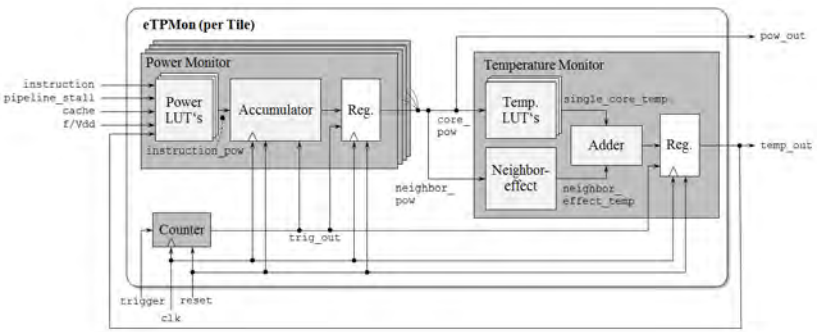


Figure 4.20: improved eTPMon framework to include a higher level of detail in the temperature and power monitoring part of eTPMon.

We worked on improving the eTPMon framework that emulates the behaviour of an ASIC power and temperature monitoring system for FPGA prototyping of MPSoC computing architectures to include a higher level of detail in the temperature and power monitoring part of eTPMon. For power monitoring, cache memories, FPU instructions, different power modes and supply voltage, temperature and frequency dependencies were included. For the temperature monitoring, the inclusion of transient temperature changes was evaluated. Also, we used accumulated monitoring data from eTPMon in order to extract information about core's ageing. In addition to the emulation of eTPMon for the FPGA demonstrator platform, a simulation framework was created in

order to visualise the output—the produced monitoring data—of eTPMon. The output data of eTPMon is usable by iDoC/DaSim in order to select suitable core(s) during resource allocation for efficient load distribution. The monitoring data produced by eTPMon can be further abstracted to the needed level of detail on higher system levels. Also the creation of per-region or per-tiles values is possible.

Besides working on new monitoring concepts, Project B4 continuously works on the optimisation of the overall monitoring system to offer an optimal trade-off between accurate monitoring data and used resources to obtain these data. The optimisation is done considering monitor types, quantities and placements, needed duty cycles for monitoring data aggregation, needed accuracy of monitoring data, the power consumption of the monitor system to alleviate problems of dark silicon and the target architecture leading to differences in the monitoring system itself.

Publications

[Pag+16] S. Pagani, L. Bauer, Q. Chen, E. Glocker, F. Hannig, A. Herkersdorf, H. Khdr, A. Pathania, U. Schlichtmann, D. Schmitt-Landsiedel, M. Sagi, Éricles Sousa, P. Wagner, V. Wenzel, T. Wild, and J. Henkel. "Dark Silicon Management: An Integrated and Coordinated Cross-Layer Approach". In: *it – Information Technology* 58.6 (Sept. 16, 2016), pp. 297–307. DOI: 10.1515/itit-2016-0028.

[SHJL16] U. Schlichtmann, M. Hashimoto, I. H.-R. Jiang, and B. Li. "Reliability, Adaptability and Flexibility in Timing: Buy a Life Insurance for Your Circuits". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE/ACM Press, Jan. 2016, pp. 705–711. DOI: 10.1109/ASPDAC.2016.7428094.

[ZLS16] G. L. Zhang, B. Li, and U. Schlichtmann. "EffiTest: Efficient Delay Test and Statistical Prediction for Configuring Post-silicon Tunable Buffers". In: *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. (Austin, TX, USA). ACM, 2016, 60:1–60:6. DOI: 10.1145/2897937.2898017.

B5: Invasive NoCs – Autonomous, Self-Optimising Communication Infrastructures for MPSoCs

Jürgen Becker, Andreas Herkersdorf, Jürgen Teich

Srivatsa Akshay, Stephanie Friederich, Leonard Masing, Sven Rheindt, Andreas Weichslgartner, Thomas Wild

Networks-on-Chip (NoCs) are the solution of choice to cope with scalability of manycore systems. Accordingly, invasive computing architectures are based on NoCs, which make up the infrastructure for communicating among the different tiles.

One part of the current research within Project B5 focuses on non-functional properties of invasive NoCs and their predictability. For this, the non-functional aspects of *realtime*, *security*, *fault tolerance*, and *energy consumption* are studied. Further investigations concentrate on NoC topologies (including 3D NoCs) and cache coherence regions that can be dynamically established among multiple tiles.

Support for Predictable Real-Time Stream Processing

The previously proposed hybrid application mapping (HAM) methodology combines the strengths of design-time analysis and run-time decision making. In [Wil+16], we present how an application, implemented in *actorX10*, can be analysed at design time. This publication also outlines how *i*NoC X10 constraints, specifying the service level and the maximal length of a guaranteed service channel, may be generated from a design-time analysis. If these constraints are fulfilled during run time, the predictable and composable nature of the *i*NoC ensures that also upper bounds for the worst-case end-to-end latencies hold.

Secure Communication for Security-Critical Applications

Sharing of *i*NoC links and routers may open possible side-channel attacks which may be used by malicious applications to extract confidential data of a security-critical application. Together with Project C5, we

propose in [Wei+16; Dre+16] a solution to provide full spatial isolation, on tile and NoC level, through hybrid application mapping (HAM) (see also Project A1 and Project A4). At design time, a design space exploration explores mappings that exclusively reserve tiles and *i*NoC communication, i. e. by assigning all available service levels of a link to an application. These mappings can be represented by so-called *shapes* (see Figure 4.21). These shapes are already determined during design space exploration and optimised for compactness to prevent fragmentation. The run-time mapping can be described as 2D placement problems where a shape of an application must not overlap with any other shape. To solve this problem, we propose fast heuristics as well as exact SAT-based methods [Wei+16].

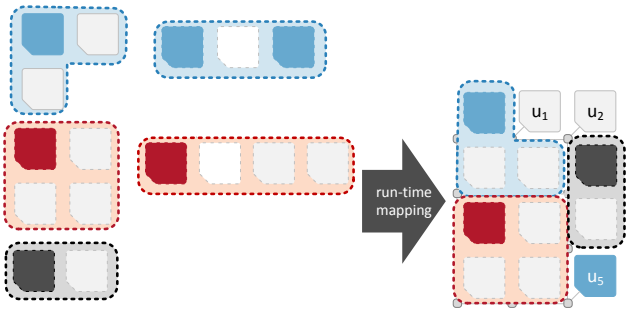


Figure 4.21: Example of spatially isolated mapping of applications. Each application can be represented by various so-called shapes which encapsulate all communication and computation (left). During run time, the shapes need to be mapped without overlap to guarantee spatial isolation.

Network Topologies

Higher dimensional networks offer shorter longest paths in the network and hence result in a lower average latency for inter-tile communication. The 3D *i*NoC architecture is partitioned across multiple layers of 3D integrated circuits. An example design implementation is shown Figure 4.22. To manage the restricted number of inter-layer connections in three-dimensional integrated circuits, we introduced a novel router implementation with heterogeneous bandwidth ports. Hence the bandwidth of through-silicon vias (TSV) is independently adjustable. In addition, the new router port includes a clock domain crossing, which makes additional synchronisation mechanisms for cross-layer connections obsolete. Figure 4.23 shows the schematic of an inter-layer

connection between two routers [FLB16].

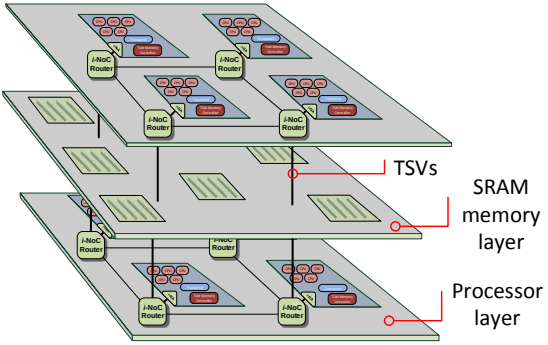


Figure 4.22: Heterogeneous multi-layer network architecture. A memory layer is placed in-between two compute layers.

B5

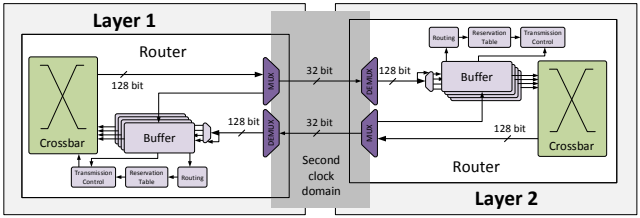


Figure 4.23: Schematic of inter-layer router connection with heterogeneous bandwidth and clock domain crossing.

Power Saving Mechanisms for *i*NoC Communication Resources

A hardware power manager unit for the network has been implemented which evaluates communication monitor data at run time to optimise the power consumption of network components without affecting the communication performance [FNB16]. The network power management employs coarse-grained power saving techniques on the one hand. Clock scaling of NoC regions and clock and power gating of complete routers is hence possible. And on the other hand, fine-grained power saving techniques minimise the power consumption in parallel. Single buffers within the router can be switched off at run time, since the biggest part of power is consumed by the virtual channel buffers.

Fault-Tolerant Communication

Manufacturing defects and aging effects are expected to cause permanent faults in future highly integrated technology nodes which are targeted by NoC-based architectures. A lightweight *second layer network* takes over the duties of defective routers. In case of no defects, the *second layer network* can be used to obtain power savings [Hei+16]. The implementation of the second layer network is now integrated into the invasive architecture to provide adaptive fault tolerance for heterogeneous MPSoCs [Lar+16]. Furthermore, disjoint-path routing can be exploited in the context of HAM to increase the reliability of NoC communication [WT16].

Support of Region-based Cache Coherence

Originally, our invasive MPSoC architectures did only support intra-tile cache coherence. This limits the maximum achievable parallelism using traditional shared memory programming to a single tile. In order to easily parallelise applications beyond tile borders without software overheads, hardware-based *inter-tile coherency* is required.

Here, Project B5 proposed a novel region-based cache coherence scheme which allows setting up and breaking down of coherency regions dynamically based on application requirements. It also aims to curb high coherence traffic and large administrative overheads associated with traditional global coherence schemes. In discussions with application programmers, it turned out that applications like DNA sequencing, sparse matrix processing, or tree traversals could profit from such inter-tile coherence schemes. Thus, inter-tile coherence support is enabled for both distributed (TLM) and shared (global) memories of the invasive architecture.

Extending hardware-based coherence support across multiple tiles is a relatively new approach especially for distributed shared-memory systems. Therefore, as a first step, we established a high-level SystemC-based simulation model of a generic tile-based architecture[6]. The generic model was refined in multiple ways to accurately match the invasive MPSoC architecture as shown in Figure 4.24. The architectural details of the model are highly configurable with respect to the number of tiles, the number of processors per tile and various cache/memory parameters. Then, we introduced a *coherency region manager (CRM)*

[6] P. Parayil et al. “Sharer Status-based Caching in tiled Multiprocessor Systems-on-Chip”. In: *HPC 2015 – 23rd High Performance Computing Symposia*. SCS, The Society for Modeling & Simulation, Apr. 2015, pp. 67–74.

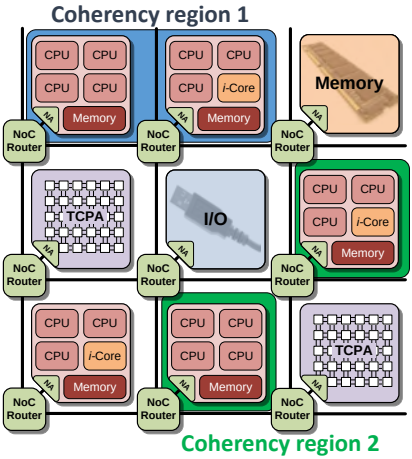


Figure 4.24: Invasive architecture with two dynamic coherency regions.

B5

module in every compute tile of the architectural model. The user can then set up coherency regions by configuring the individual CRMs. For example, the user can set up a coherency region consisting of four compute tiles where each tile shares its complete TLM every other tile in the region. Additionally, the CRM supports fine granular configurations i. e. regions can be set up where only a specific memory range is shareable. The CRM monitors sharers of these addresses and enforces appropriate invalidations for all tiles lying in the coherency region. Thus, the CRM provides a coherent view over all memory ranges present in the coherency region.

The processor module is modelled to replay benchmark traces which are generated externally using Gem5. The raw trace file has to be formatted to suit invasive computing requirements before being consumed by the trace-based processor. For example, the trace data needs to be mapped onto the invasive distributed shared-memory architecture. An inefficient mapping process may limit the performance benefits of region-based cache coherence. Therefore, several algorithms are being developed to ensure optimal placement of data.

Work is actively ongoing to evaluate the performance of region-based cache coherence using the simulation model, to investigate efficient task-data mapping techniques and to further refine the simulation model itself before finalising the hardware implementation.

Circuit switching extension

The initial NoC design in the first funding phase was developed based on the concept of a packet-switched network. As an extension to this basic architecture, circuit-switching is investigated since it promises many advantages over packet-switching, namely less buffer space, lower latency and better QoS guarantees. In our current work, a hybrid synchronous implementation is designed and furthermore a fully asynchronous implementation is currently investigated in a simulation-based on the cycle-accurate simulator iNoCsim. In this model, new dynamic approaches that will cope with the drawbacks of circuit switching are developed, tested and evaluated before a physical implementation of promising features will be tackled.

Integration Work and Demonstration

As in previous years, a lot of effort was put into implementation, debugging, and prototyping of invasive MPSoC architectures. However, this effort is mandatory to provide a hardware platform including the novel features of all the projects of research area B for the research of the projects of areas C and D. When setting up a hardware architecture with novel components, *debugging* plays a key role.

Outlook

One of the main tasks in 2017 for Project B5 will be carrying on the development of region-based cache coherency. The impact of these mechanisms on real-world applications shall be investigated in collaboration with Project C3. Additionally, Project B5 plans to explore the possibility to further enhance the benefits of region-based cache coherence with dedicated NoC support, e. g. by utilising and interfacing with the circuit switching extensions. Also, Project B5 will further contribute to the integration activities of Project Z2 to setup the demonstrator for the final review of phase two. Furthermore, Project B5 will contribute to the multi-tile demo which will showcase timing predictability through invasion of computation and NoC resources.

Publications

[Dre+16] G. Drescher, C. Erhardt, F. Freiling, J. Götzfried, D. Lohmann, P. Maene, T. Müller, I. Verbauwhede, A. Weichslgartner, and S. Wildermann. “Providing Security on Demand Using Invasive

Computing". In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 281–295. DOI: 10.1515/itit-2016-0032.

[FLB16] S. Friederich, N. Lehmann, and J. Becker. "Adaptive Bandwidth Router for 3D Network-on-Chips". In: *Applied Reconfigurable Computing*. 2016, pp. 352–360. DOI: 10.1007/978-3-319-30481-6_30.

[FNB16] S. Friederich, M. Neber, and J. Becker. "Power Management Controller for Online Power Saving in Network-on-Chips". In: *International Symposium on Embedded Multicore/Manycore SoCs (MCSoC)*. Vol. 10. Sept. 2016.

[Hei+16] J. Heisswolf, S. Friederich, L. Masing, A. Weichslgartner, A. M. Zaib, C. Stein, M. Duden, J. Teich, T. Wild, A. Herkersdorf, and J. Becker. "A Novel NoC-Architecture for Fault Tolerance and Power Saving". In: *Proceedings of the third International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: IEEE, Apr. 4, 2016, 8 pp.

B5

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. "Providing Fault Tolerance Through Invasive Computing". In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[Tei+16] J. Teich, M. Glaß, S. Roloff, W. Schröder-Preikschat, G. Snelting, A. Weichslgartner, and S. Wildermann. "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.

[WT16] A. Weichslgartner and J. Teich. "Position Paper: Towards Redundant Communication through Hybrid Application Mapping". In: *Proceedings of the third International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: IEEE, Apr. 4, 2016, 4 pp.

[Wei+16] A. Weichslgartner, S. Wildermann, J. Götzfried, F. Freiling, M. Glaß, and J. Teich. "Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs". In: *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*. (Sankt Goar, Germany). ACM, May 23, 2016, pp. 153–162. DOI: 10.1145/2906363.2906370.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

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

Wolfgang Schröder-Preikschat, Daniel Lohmann, Jörg Henkel, Lars Bauer

Gabor Drescher, Christoph Erhardt, Timo Hönig, Sebastian Maier, Florian Schmauss, Jens Schedel, Anuj Pathania, Volker Wenzel

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 multi-/manycore machines.

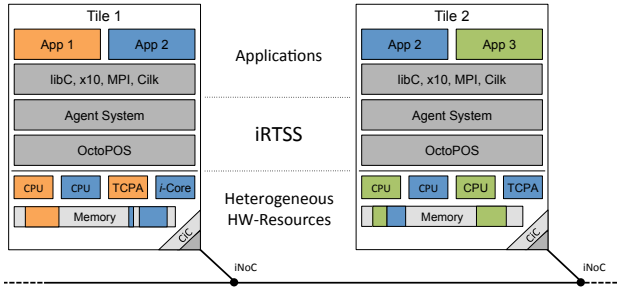


Figure 4.25: *i*RTSS Architecture on a multi-tile system. Colours indicate invaded resources.

Architectural Overview

Figure 4.25 provides a high-level view of the current *i*RTSS architecture. Key elements are OctoPOS,[7] the parallel operating system (POS)

[7] The prefix 'Octo' stems from the denotation of a nature which is highly parallel in its actions as well as adaptable to its particular 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.

that 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 runtime libraries for various kind of invasive-parallel applications and the OctoPOS kernel.

The Configurable OctoPOS Kernel

We provide OctoPOS for a variety of platforms—with and without support for dedicated hardware. 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 *unfiltrated* way.

In 2016, we could, in cooperation with many other projects, especially demonstrate the benefits of this approach for the security demands of invasive applications [Dre+16] and by reducing the energy demand at operating-system level [HHSP16; EHSP16].

Native x86_64 Port: The main purpose of the x86_64 version is comparability: In cooperation with Project C3, which provides the compiler back end for OctoPOS x86_64, application projects can use this platform to compare the invasive approach with existing (e. g. Linux-based) approaches on common, state-of-the art hardware.

In this realm, the OctoPOS team performed detailed analyses on the 4-socket Xeon E7v3 (Haswell) system with 96 logical cores that became available end of 2015. One result is a significant reduction of kernel latencies and kernel-induced noise by the decoupling of user-level and kernel-level activities on the *i*-let level. OctoPOS now supports user-level scheduling for application *i*-lets, avoiding otherwise costly syscall transitions, especially when the system runs with full protection (see below). For this purpose, we have introduced the new concept of kernel *i*-lets, which perform OS-level activities on dedicated cores. Thereby, a user-level *i*-let issuing a (conceptually) blocking syscall does not cause a complete CPU to become unavailable for user-level scheduling. Instead, the syscall execution is delegated to a *kernel* i-*let* to be processed asynchronously by a dedicated processor, while the issuing CPU picks the next request from the user-level queue. Only if no further request is available, it *temporarily* enters kernel space to assist in the execution of kernel *i*-lets or enter a sleep state (Figure 4.26). On this base, OctoPOS now provides highly efficient scheduling not only for invasive applications written in InvadeX10, but also for standard programming models, such as Cilk+ or MPI. The latter, again, facilitates comparability

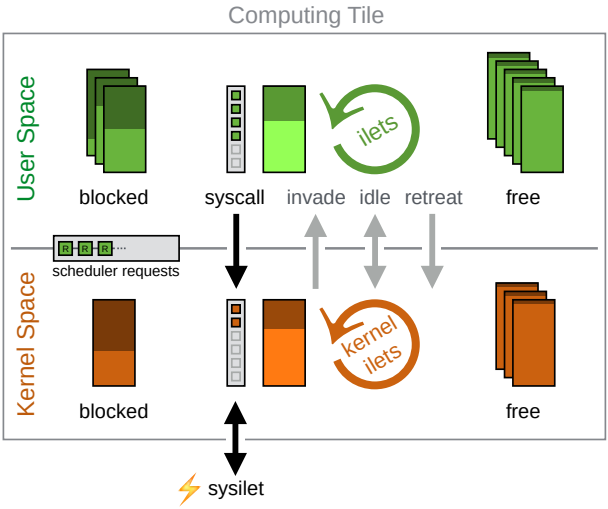


Figure 4.26: OctoPOS asynchronous kernel operation by kernel *i*-lets.

C1

of the invasive hardware/system-software stack to standard Linux-based systems.

LEON Platform Improvements: Besides work on the x86_64 port, the OctoPOS team again spent significant resources in 2016 on stability improvements for the invasive hardware platform and the transition to the new proFPGA demonstrator platform. This includes the provisioning of test cases, the development of an automatic testing infrastructure, as well as general debugging support.

Adaptive Memory Protection: Hardware-based isolation with respect to protection levels and memory access shall be a run-time configurable feature in *i*RTSS in order to support dynamic (de-)virtualisation: There is little need to use the MMU and provide the corresponding operating-system functions if, for instance, the machine is used in single-user mode or all application processes stem from programs written in a type-safe language, for which certain properties can be guaranteed ahead of time (e. g. X10). State-of-the-art operating systems statically determine which applications or application's modules run under memory protection and which do not. This assignment of protection does not change at run time, thus, all applications are charged with significant costs of isolation in the course of memory management, system call and return, interrupt handling, process dispatching and inter-process communication [DSP16].

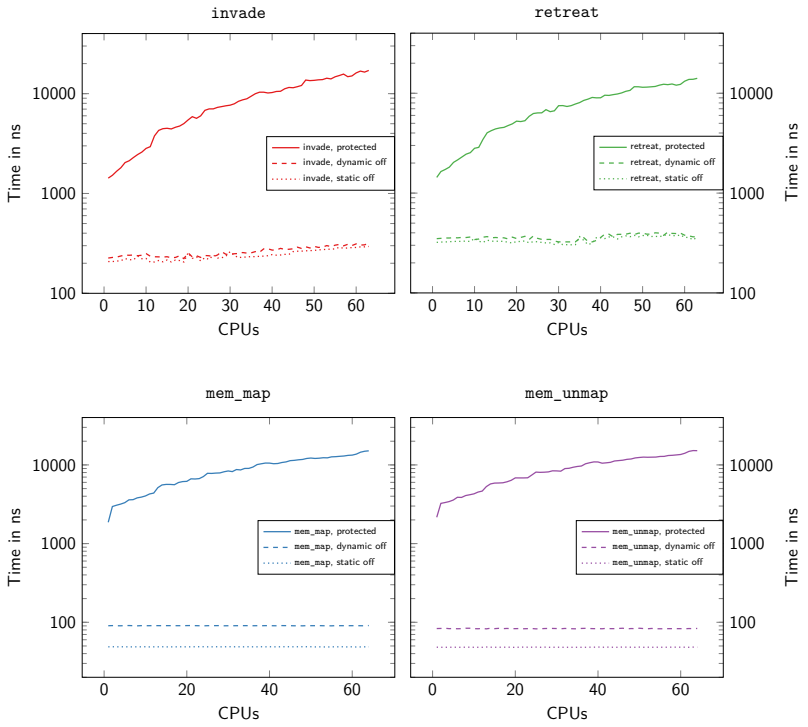


Figure 4.27: Costs and benefits of adaptive memory protection for claim operations of 1 to 60 CPUs and 512 pages of memory on a logarithmic time scale.

C1

OctoPOS, in contrast, now provides *adaptive protection*: Memory protection and privilege separation can be enabled and disabled on the fly, depending on the actual applications at run time. Figure 4.27 shows that, especially with a larger claims, leaving off protection when not needed (*dynamic off*) yields dramatic performance benefits (from 39× for the retreat operation up to 184× for the mem_unmap operation), while there are no penalties compared to an always protected system if protection is enabled later on (*protected)*. Technically, for the possibility to enable protection *later* (at runtime), also OctoPOS needs to manage logically separate address spaces by setting up the MMU and the respective page tables from the very beginning. However, the price of this flexibility, that is, the overhead in comparison to an always unprotected system (*static off*), is very low from (1.01× for retreat up to 1.86× for mem_unmap).

Constructive Measures for Low Energy: With constructive measures at operating-system level [HHSP16] and run-time mechanisms embedded into the operating-system kernel, we reduce the energy footprint of invasive applications during execution. Aligned to individual task characteristics (e. g. CPU-boundedness, memory-boundedness) the OS kernel configures and sets up hardware components specific to the current task in order to reduce the energy demand of the system during execution of the application. The task-specific settings are extracted at design time of the application and are deposited with the application, and hence, they are available to the OS kernel at run time. Figure 4.28 shows three different applications and their individual energy and time demands for execution (run-to-completion). In dependance on the set of applied energy-saving features at run time, the non-functional properties and resource-demand of the applications differ greatly. With constructive measures we reduce the energy demand of the by up to 2.9x compared to the baseline while keeping functional properties of the task intact. The constructive measures increase the resource awareness of the operating system aligned to the concept of invasive computing. Additionally to this, we further applied and extended the concepts of these constructive measures to support more additional scenarios and runtime environments, such as high-performance computing (HPC) environments.

C1

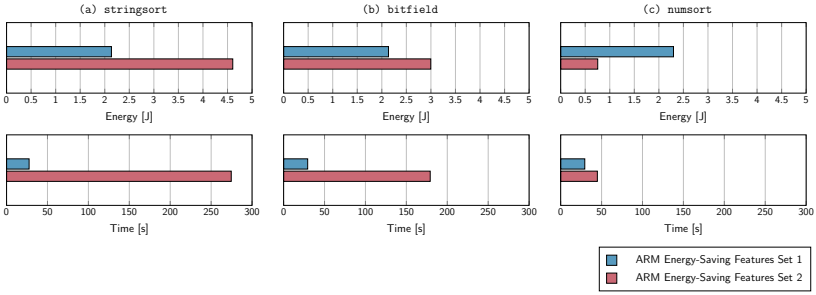


Figure 4.28: Constructive measures at OS level increase the resource awareness by processing tasks according to non-functional properties (e. g. energy demand, time demand).

Decreasing the energy-demand of workloads at OS level: To increase the energy efficiency for the execution of heterogeneous workloads, we designed an extension to our operating-system kernel which uses a decision-making process at run time [EHSP16]. At run time, processing criteria for the workloads are considered and matched by

the OS kernel-extension in order to remove jitter and unnecessary background noise from the system during execution. To exploit available energy resources to the greatest extent, we use several energy-saving features at hardware level which are independent from each other. Individual job characteristics are compared with available processing units at system-level in order to find a suitable match of *claims* and *grants*. During the execution of the workloads we monitor and refine the requirements of workloads in order to optimise the resource consumption of the invasive system, and thus, the OS kernel continuously readjusts system settings to reduce the consumption of resources. The approach [EHSP16] considers the optimisation of non-functional properties (e. g. reduction of execution time and/or energy demand) while maintaining the functional properties of the individual workloads.

The Agent System

The task of the *i*RTSS agent system is to provide a decentralised, scalable resource management for invasive applications. Per application, agents negotiate among each other for computational resources by comparing speedup curves to optimise the performance of their applications.

Agent System visualisation: There is an ongoing effort to develop an online visualisation tool to illustrate the resource allocation of the agent system on the final demonstrator. Therefore we extended the Qt-based visualisation tool *InvadeView*, that was originally developed within Project C2. The visualisation tool is divided into two parts: Our modified version of InvadeView running on a *Linux* host machine and extensions that are integrated into the agent system. The extended agent system can immediately be run on the *Invasive Demonstrator* or the guest layer and is connected to the host machine via our highly scalable push-messaging architecture based on Ethernet. System state data is generated by the agent system, which is then processed by an on-demonstrator utility program and send over to the Linux machine. An on-Linux utility then receives the data, formats it and forwards it to the extended InvadeView application for display. We augmented InvadeView with additional functionality. This includes support for stepping through monitor events forwards and backwards, highlighting of claims, support for heterogeneous hardware and statistical plotting of the agent system's resource allocations over time. Figure 4.29 shows a representative screenshot from the visualisation. The visualisation helps the application developers to debug their applications on the *i*RTSS system and it helps them to understand how their applications are parallelised and how they compete for resources.

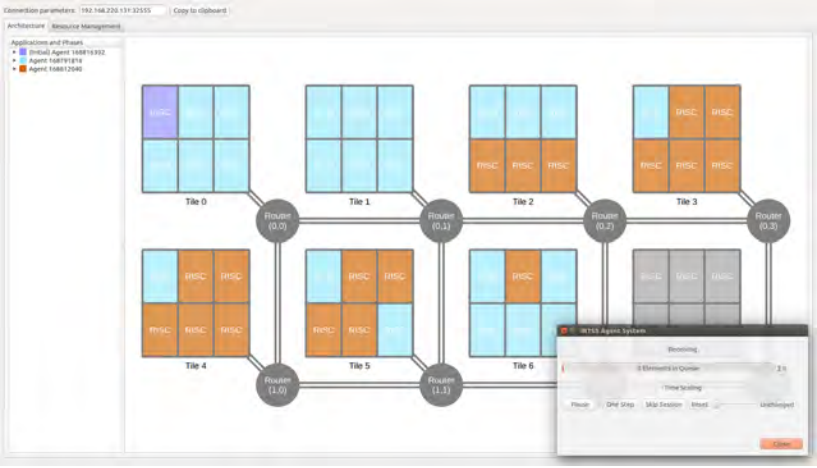


Figure 4.29: Agent System Visualization

Agent System 2.0 (*AS2.0*): Considerable effort went into a significantly improved decentralised implementation of the agent system (*AS2.0*) with a new underlying software architecture (see Figure 4.30). Agents are tile local entities that represent individual applications and negotiate the application's resource demand with other agents in the system. There is one *A*gentTileManager on every tile that stores information about all agents registered to a particular tile. The AgentClusterManager acts as a local directory service to ease the discovery of agents among each other within a defined neighbourhood. The AgentTileManager registers every agent to one of the AgentClusterManagers. Agents use the Remote Procedure Call (RPC) mechanism provided by OctoPOS to communicate. To allow for improved parallel execution and scalability of *AS2.0*'s underlying communication infrastructure, we implemented lock-free data structures for the *i*RTSS. In detail those include a lock-free hash map and a lock free list. The 'Compare and Swap' (CAS) primitive is used for the lock free data structures as it is the only one available on the InvasIC demonstrator.

Claim Proxy Mechanism: A proxy mechanism was introduced into the current agent system that allows *i*-lets the manipulation of their corresponding claim across different tiles. Applications can now make better use of claims that span multiple tiles. The proxy mechanism makes use of OctoPOS' underlying RPC functionality. The changes in the software architecture of the current agent system can be seen in Figure 4.31.

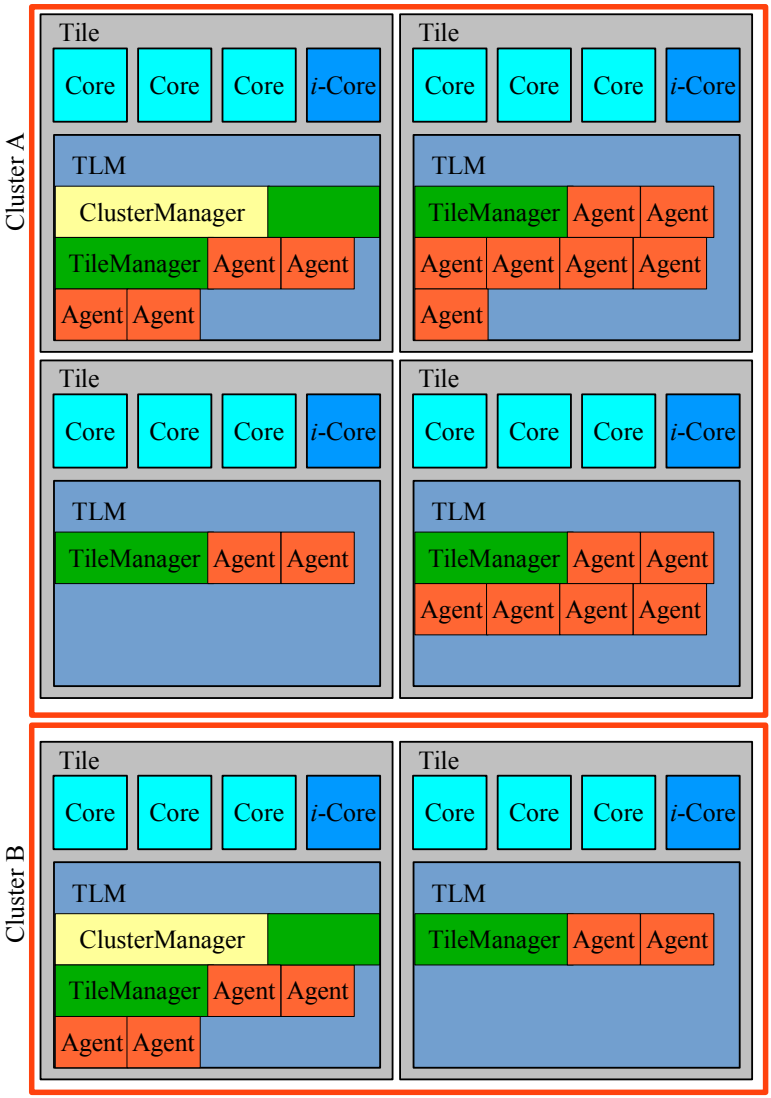


Figure 4.30: Overview of *AS2.0* architecture

Representation & Satisfaction of Quality-of-Service requirements: We introduced a sophisticated constraint hierarchy which allows the representation of complex quality of service requirements ('constraints')

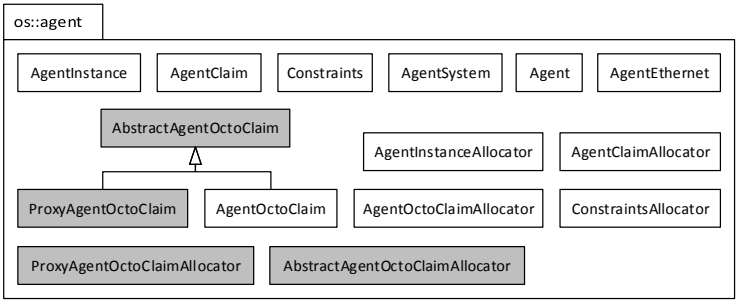


Figure 4.31: Overview of classes that were added to implement the proxy mechanism.

in the current agent system. A backtracking algorithm has been implemented to find claims for complex constraints represented in the constraint hierarchy. The hierarchy is modelled with a composite pattern as shown in Figure 4.32.

C1

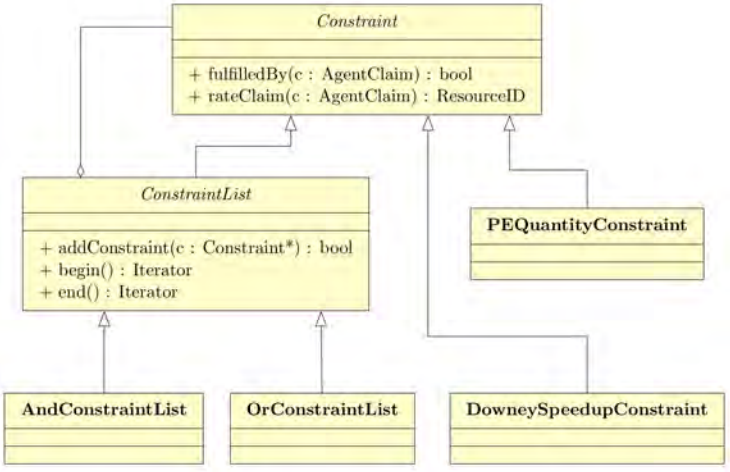


Figure 4.32: Constraint hierarchy modelled with the composite design pattern. All constraints inherit from the abstract class Constraint. Classes that inherit from ConstraintList are constraints themselves, and can store multiple other Constraints.

Outlook

We will continue the work on the low-overhead dynamic reconfiguration of *i*RTSS, which will be extended to also support dynamic reconfiguration of processor sharing and preemption points in order to facilitate on-demand migration of claims (e. g. triggered by the dark-silicon management layer). On the technical level, this requires specific code-generation support for low-overhead kernel-level dynamic reconfiguration even under high contention by multiverse functions [RDGL16], which will be integrated into OctoPOS development.

Future plans also include the quick deployment of the first *AS2.0* release with the basic functionality in place to run invasive applications. A stable interface is to be designed that can be integrated with the X10 runtime environment of Project A1. Additionally, we want to develop a concept for the distribution of the efficient satisfaction of quality of service requirements. Together with Project A4, we envision to add support for actor-based constraint graphs in extension of our current constraint solving system. Another issue that we want to tackle together with Project B5 is support for dynamic cache coherency, i. e. the agent system should support finding tile regions that can be made cache coherent in order to improve application performance.

Publications

[DSP16] G. Drescher and W. Schröder-Preikschat. “Adaptive Memory Protection for Many-Core Systems”. In: *Adaptive Isolation for Predictability and Security (Dagstuhl Seminar 16441)*. Vol. 6. 2016.

[Dre+16] G. Drescher, C. Erhardt, F. Freiling, J. Götzfried, D. Lohmann, P. Maene, T. Müller, I. Verbauwhede, A. Weichslgartner, and S. Wildermann. “Providing Security on Demand Using Invasive Computing”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 281–295. DOI: 10.1515/itit-2016-0032.

[EHSP16] C. Eibel, T. Hönig, and W. Schröder-Preikschat. “Energy Claims at Scale: Decreasing the Energy Demand of HPC Workloads at OS Level”. In: *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. May 2016, pp. 1114–1117. DOI: 10.1109/IPDPSW.2016.69.

[HHSP16] T. Hönig, B. Herzog, and W. Schröder-Preikschat. “The Narrow Way: Constructive Measures at Operating-System Level for Low Energy Use”. In: *Proceedings of the 30th Environmental Informatics Conference (EnviroInfo 2016)*. 2016, pp. 329–335.

[Pat+16a] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. “Optimal Greedy Algorithm for Many-Core Scheduling”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* PP.99 (2016), pp. 1–1. DOI: 10.1109/TCAD.2016.2618880.

[Pat+17] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. “Defragmentation of Tasks in Many-Core Architectures”. In: *ACM Transactions on Architecture and Code Optimization* (2017).

[Pat+16b] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. “Distributed Fair Scheduling for Many-Cores”. In: *Design Automation and Test in Europe (DATE)*. Dresden, Germany, 2016.

[Pat+16c] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. “Distributed scheduling for many-cores using cooperative game theory”. In: *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. ACM. Austin, TX, USA, 2016, pp. 133–139.

[RDGL16] V. Rothberg, C. Dietrich, A. Graf, and D. Lohmann. “Function Multiverses for Dynamic Variability”. In: *Foundations and Applications of Self* Systems*. Ed. by J. Andersson, R. Capilla, and H. Eichelberger. Augsburg, 2016, pp. 1–5. URL: https://www4.cs.fau.de/Publications/2016/rothberg_16_dspl.pdf.

[Taj+16] H. Tajik, B. Donyanavard, N. Dutt, J. Jahn, and J. Henkel. “SPMPool: Runtime SPM Management for Memory-Intensive Applications in Embedded Many-Cores”. In: *ACM Trans. Embed. Comput. Syst.* 16.1 (Oct. 2016), 25:1–25:27. DOI: 10.1145/2968447.

C2: Simulative Design Space Exploration

Frank Hannig

Sascha Roloff, Vahid Lari

Project C2 investigates novel simulation techniques that enable the validation and variants' exploration of architecture options as well as invasion strategies. The timed functional simulation, evaluation, and co-exploration of both invasive resource-aware programs and invasive heterogeneous tiled architectures are the major objectives of Project C2. Likewise, the architectures to be investigated are highly complex and diverse as well as the workloads to be simulated. For example, a simulation scenario could be made up of: a heterogeneous multi-tile architecture and multiple competing highly parallel applications. In order to handle this complexity, novel parallel simulation methods and environments have been developed. Parallel simulation techniques enabled us to test and evaluate the concepts of invasive computing across all project areas, especially, without the need to have full hardware or software implementations available.

In addition to this foundation, we advanced our research within the last year as follows. The first major contribution has been the modelling of parallel workloads in form of actor-based X10 programs and automatically generated task graphs based on parallel programming patterns. The second research direction has focused on simulation-based predictability analysis of timing and fault tolerance. Finally, we present our latest achievements regarding simulative design space exploration.

Modelling of Parallel Workloads

An important milestone in the context of modelling invasive applications was the development of a novel actor-based programming library in X10 [Rol+16] in close cooperation with Project A1 and Project A4. It brings together the well known concepts of the actor model and the Asynchronous Partitioned Global Address Space (APGAS) paradigm and follows a formalism in describing actor functionality and communica-

tion, which allows a clear separation of control flow and data flow. Similar to the previous task model[8], this formal actor model may be seamlessly mapped to the PGAS programming model as data processing is performed solely on locally available data, whereas all data transmissions between actors are explicitly modelled by channels.

The X10 PGAS model uses the notion of places and allows to distribute an actor graph among the available places in order to exploit the processing power of the underlying cores or for an optimal workload distribution. Each actor and each channel is mapped to a certain tile of a previously reserved claim, on which they reside during the execution of the actor graph. In case of an adaptation of the distribution at runtime, the library allows to dynamically migrate actors and channels to other tiles.

Besides modelling workloads by actorX10 applications, Project C2 also investigates the usage of task graphs based on *parallel programming patterns*. In this context, we developed a method to systematically generate task graphs, which represent the concurrent execution behaviour of common parallel programming patterns, such as *divide and conquer*, *MapReduce*, or *embarrassingly parallel*. In addition, we evaluated a benchmark set of representative parallel applications with respect to their parallel runtime behaviour. For this purpose, we instrumented the runtime library of different parallel programming environments (Cilk, Pthreads, Python) in order to extract the parallel control flow for program runs with varying input parameters. These extracted graphs are compared with our generated synthetic patterns and used as basis for *dynamic invasion graphs*. Such graphs are created at runtime based on a parallel programming pattern and the availability of resources in a simulated multi-tile architecture.

Simulation-based Predictability Analysis

In cooperation with Project A1 and Project A4, we established a case study to analyse the ability of invasive computing to isolate applications from each other in order to reduce execution time and throughput jitter of soft real-time image stream processing algorithms to a minimum on a heterogeneous NoC-based multicore architecture [Tei+16]. The case study chosen was a robot vision object detection algorithm chain, which provides a lot of pipeline parallelism in case of processing a stream

[8] S. Roloff, F. Hannig, and J. Teich. "Towards Actor-oriented Programming on PGAS-based Multicore Architectures". In: *Workshop Proceedings of the 27th International Conference on Architecture of Computing Systems (ARCS)*. Lübeck, Germany: VDE Verlag, Feb. 25–28, 2014.

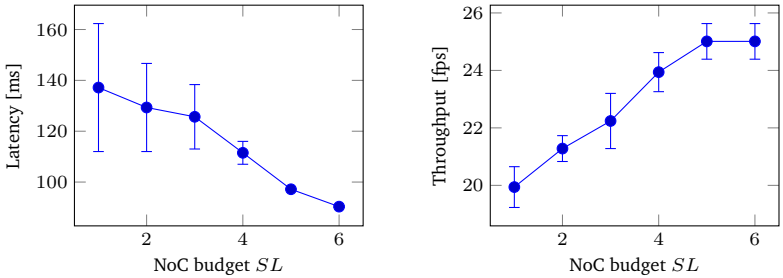


Figure 4.33: Plot of the object detection task chain latency and throughput and their min/max values in dependence of different service levels SL allocated for inter-task communications [Tei+16].

of images, e. g. delivered by a camera. We implemented this object detection chain using the *actorX10* library [Rol+16], encapsulating each of the object detection tasks in an actor.

In order to show the advantages of invasive computing, we considered an application scenario, where the object detection application is executed at the same time as a distributed Monte Carlo simulation. Both applications are mapped on the target architecture such that their communication paths are overlapping. Here, we tried to minimise the execution time and throughput jitter of the object detection task chain by applying invasive computing principles (in particular isolation). For this purpose, guaranteed service (GS) channels were reserved between the tiles on which the tasks were mapped to. In contrast, the Monte Carlo simulations only use best effort (BE) channels between communicating tasks.

C2

We simulated the described application scenario on a 4×4 NoC architecture using the simulator InvadeSIM developed in the first funding phase. We evaluated this scenario according to latency and throughput, where latency denotes the time it takes to process one image frame in the pipeline and throughput, measured in fps, denoting the rate of how many frames can be processed by the pipeline per second. Different service levels SL were reserved for the GS channels of the object detection application. As can be seen in Figure 4.33, in both cases, the objective values latency and throughput as well as their jitter significantly improve when increasing the reserved communication bandwidth. Still, the simulation-based evaluation only allows to determine observed predictability markers, and is therefore only applicable for soft user requirements. Whenever the user specifies hard requirements on objec-

tives, their predictability markers have to be determined analytically.

Together with Project B2 and Project C3, we extended the simulator for tightly-coupled processor arrays (TCPAs) in order to empirically evaluate the proposed structural redundancy mechanisms for different levels, such as dual (DMR) and triple modular redundancy (TMR) when executing loop programs on TCPAs with immediate, early, and late voting [Lar+16; Lar16a]. Thanks to the developed simulation-based fault injection environment, we were able to evaluate the reliability gains when using each of the mentioned redundancy mechanisms compared to non-redundant executions.

Architecture Exploration

Besides workload modelling and simulation-based predictability analysis, the main focus of Project C2 is the automatic *co-exploration* across all platform layers such as architectures, invasion strategies and applications. In this regard, we developed a basic infrastructure for architecture exploration[9] based on our parallel execution-driven simulator InvadeSIM. Architecture exploration is used to determine a set of optimal architecture configurations for a given mix of applications with respect to multiple objectives, including execution latency, temperature distribution, and power consumption. For this purpose, we utilised the optimisation framework Opt4J[10] to generate different architecture variants based on genetic algorithms and to evaluate them using InvadeSIM. A general overview of this tool coupling can be seen in Figure 4.34.

Here, important problems to be solved were the encoding of an architecture, the definition of appropriate operations to manipulate the architecture encoding as well as the coupling of these tools. The encoding of an architecture variant is called *genotype* and includes the number of tiles, the number of processors on that tiles as well as the configuration of these processors such as clock frequency. All parameters of an architecture are set according to a given range in order to generate feasible architecture configurations (integer genotype). A special *creator* class as part of the Opt4J framework had to be implemented, it is responsible to generate an initial set of architecture genotypes and new

[9] R. Andone. “Automatische Generierung von Architekturvarianten zur Entwurfsraumexploration von heterogenen MPSoCs”. Bachelor Thesis. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Mar. 1, 2016.

[10] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. “Opt4J: A Modular Framework for Meta-heuristic Optimization”. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. (Dublin, Ireland). ACM, 2011, pp. 1723–1730. DOI: 10.1145/2001576.2001808.

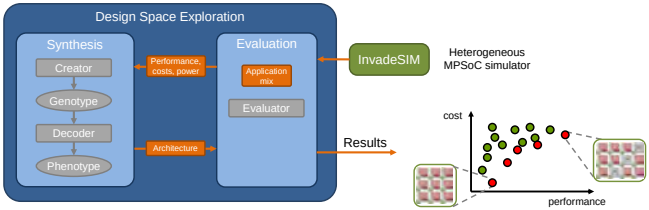


Figure 4.34: Block diagram of the design space exploration framework for finding optimal architectures with respect to given objectives for a mix of parallel applications.

architecture genotypes based on the current population by applying genetic operators. A second class called *decoder* translates the generated encoded architectures into a so-called *phenotype*, which is a format that can be used by an *evaluator*. Here, a generic XML-based architecture description format was defined to interface the heterogeneous MPSoC simulator InvadeSIM, which acts as evaluator in the context of the Opt4J optimisation process. It loads the XML architecture description, simulates the given X10 applications onto that architecture, determines the specified objectives, and feeds them back to Opt4J for determining a set of Pareto-optimal architectures.

Other Scientific Activities and Public Dissemination

In addition to the aforementioned research, Project C2 contributed also very much in disseminating the concepts of invasive computing in talks, tool presentations, and demonstrators. One highlight was this year's international trade fair HANNOVER MESSE, where we presented a demonstrator illustrating the principals of invasive computing. The exhibit was located at the booth of the FAU Interdisciplinary Centre for Embedded Systems (ESI), situated in the exhibition area "Research and Technology". The demonstrator showed how applications are able to express the demand for more or less processors while the system attempts to meet the demands. While processing, a 360° camera focussed an object that could be moved by the visitors. The more processors were used by the program, the faster and more efficient the object tracking worked. The more other programs got active in the background while simulating, the more efficiency of the image processing was reduced. This demonstrator was also showcased at DTC 2016 (The Munich Workshop on Design Technology Coupling). Also at DTC, Vahid Lari gave a talk about providing fault tolerance through invasive computing and fault simulation [Lar16b]. Furthermore, we had two tool presentations at the University Booth of DATE 2016. Both InvadeSIM [RHT16], our

parallel execution-driven simulator for functional and timing simulation of resource-aware applications on NoC-based MPSoCs, and the cycle-accurate TCPA simulator as part of the TCPA tool chain [Tan+16] were presented.

Publications

[Lar16a] V. Lari. *Invasive Tightly Coupled Processor Arrays*. Springer Singapore, 2016. DOI: 10.1007/978-981-10-1058-3.

[Lar16b] V. Lari. "Providing Fault Tolerance Through Invasive Computing". Talk at DTC 2016, The Munich Workshop on Design Technology Coupling, Munich, Germany. June 30, 2016.

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. "Providing Fault Tolerance Through Invasive Computing". In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[RHT16] S. Roloff, F. Hannig, and J. Teich. "InvadeSIM: A Simulator for Heterogeneous Multi-Processor Systems-on-Chip". Tool Presentation at the University Booth at Design, Automation and Test in Europe (DATE), Dresden, Germany. Mar. 14–18, 2016. URL: https://www.date-conference.com/system/files/file/date16/ubooth/37912.pdf.

[Rol+16] S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. "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.

[Tan+16] A. Tanase, M. Witterauf, Éricles R. Sousa, V. Lari, F. Hannig, and J. Teich. "LoopInvader: A Compiler for Tightly Coupled Processor Arrays". Tool Presentation at the University Booth at Design, Automation and Test in Europe (DATE), Dresden, Germany. Mar. 14–18, 2016. URL: https://www.date-conference.com/system/files/file/date16/ubooth/37913.pdf.

[Tei+16] J. Teich, M. Glaß, S. Roloff, W. Schröder-Preikschat, G. Snelting, A. Weichslgartner, and S. Wildermann. "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.

C3: Compilation and Code Generation for Invasive Programs

Gregor Snelting, Jürgen Teich

Sebastian Buchwald, Frank Hannig, Manuel Mohr, Tobias Schwarzer, Ericles Sousa, Alexandru Tanase, 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.

Figure 4.35 shows the structure of the developed compiler. The compiler is based on an existing X10 compiler, but has been extended with new transformation phases to support TCPAs as well as general purpose cores, such as SPARC/x86 processors as well as *i*-Cores, through libFIRM.

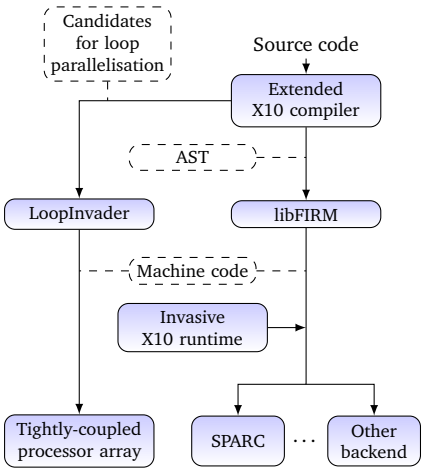


Figure 4.35: Compiler framework for invasive computing.

For general purpose cores, our research was focused on optimisations as well as on verification. First, to accelerate program execution on invasive architectures, we proposed, implemented, and evaluated a novel technique to copy complex data structures more efficiently between coherence domains. Second, we formally verified our proposed SSA construction algorithm with the proof assistant Isabelle/HOL. Third, we designed, implemented, and evaluated a system to synthesise a provably correct instruction selector from semantic descriptions of intermediate representation and target machine.

For TCPAs, we developed a mathematical framework to *symbolically* map loop programs in a hierarchical fashion to balance I/O bandwidth and memory requirements without knowing the number of processing elements actually acquired at runtime. We demonstrated our first results at the DATE 2016 university booth [Tan+16].

Symbolic Multi-Level (Hierarchical) Tiling

To map a loop program onto a TCPA independently of the number of processing elements—which, in invasive computing, we do not know at compile time—the mapping must be *symbolic*, meaning that it retains parameters until runtime. Symbolic mapping is a two-step process: First, *symbolic tiling* tessellates the iteration space of a loop into disjoint, congruent tiles of parametric size. Second, *symbolic scheduling* determines a symbolic schedule that assigns an execution time to each iteration.

We developed an algorithm for mapping loop programs symbolically on multiple levels in a *hierarchical* fashion. Because the tile size on each level correlates to different constraints such as I/O bandwidth or memory size, this allows the compiler to balance such constraints systematically. This is necessary because TCPAs only have a fixed number of I/O channels and limited memory per processing element.

So far, however, we had only considered the particular case that the iterations of the first and last tiling level are executed sequentially, while the tiles of all levels in between are executed in parallel. Because other combinations may lead to better utilisation, in 2016 we alleviated this limitation to allow each level of the hierarchical tiling to be scheduled either in parallel or sequentially. For this, we developed a rigorous mathematical framework that subsumes our previous work as special cases, making it very general.

Using this framework, we showed that it is possible to derive such symbolic, hierarchical schedules analytically at compile time.

Modulo Scheduling of Symbolically Tiled Loops

Maximising the throughput of a loop requires to overlap the execution of subsequent iterations, a concept known as *software pipelining*. Software pipelining is most commonly achieved by implementing resource-constrained modulo scheduling, which our approach to symbolic mapping described in the previous section makes impossible to use naively because the resulting constraints and objective function contain multiplicative parameters.

To circumvent this, we developed a method for symbolically tiled loops that enables the usage of existing resource-constrained modulo scheduling algorithms [WTHT16]. Our method first only considers constraints that do *not* contain any parameters and solves the resulting *reduced* modulo scheduling problem. Of course, this entails a trade-off: to ensure that the determined schedule is feasible at runtime, the tile size must be larger than a minimum tile size computed at compile time from the remaining, parametric constraints. If the schedule is infeasible, a slower fallback schedule must be used.

However, we showed that for many real-world applications, minimal tile sizes are very small or do not even exist, yet the gained speed-up is large.

Fault Tolerance in Invasive Computing and Loop Processing

Because of shrinking feature sizes, integrated circuits are increasingly susceptible to faults, necessitating fault tolerance measures. Therefore, in close collaboration with Project B2 and Project B5 [Lar+16; Tei16], we showcased how to apply invasive computing techniques to provide adaptive fault tolerance to applications.

For TCPAs in particular, we have developed a compiler transformation that introduces modular redundancy by replicating loop nests and invading redundant claims of processing elements[11]. Different frequencies of voting (once every iteration, once per tile, only once) enable a trade-off between reliability and error detection latency.

Compiler LoopInvader

The compilation flow of the loop compiler LoopInvader has advanced a lot in 2016. It is depicted in Figure 4.36: LoopInvader generates a unified configuration binary that contains configuration data for all components of the TCPA and characterises the inputs and outputs of the loop program.

In particular, LoopInvader provides a communication function that is called at runtime by the Reconfiguration and Communication Processor within the TCPA tile each time a buffer interrupt indicates an empty or full buffer. The communication function then schedules the necessary transfer to or from the buffer according to the size of the corresponding

[11] A. Tanase et al. "On-Demand Fault-Tolerant Loop Processing on Massively Parallel Processor Arrays". In: *Proceedings of the 26th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. (Toronto, Canada). IEEE, July 27–29, 2015, pp. 194–201. DOI: 10.1109/ASAP.2015.7245734.

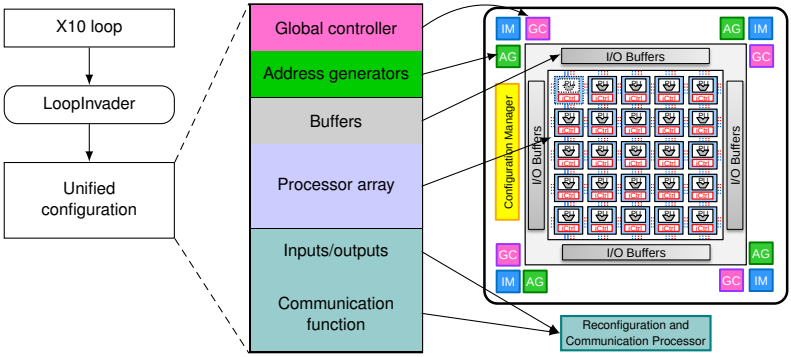


Figure 4.36: LoopInvader generates a unified configuration binary that contains the configuration data for all TCPA components including assembly programs of PEs, address generators (AG), global execution controller (GC), and characterises inputs and outputs (the correspondence is illustrated by arrows and matching colours).

input or output array, tile size, scanning order, and buffer size. Using a function instead of fixed parameters allows us to use arbitrary access patterns without changing the format of the configuration data.

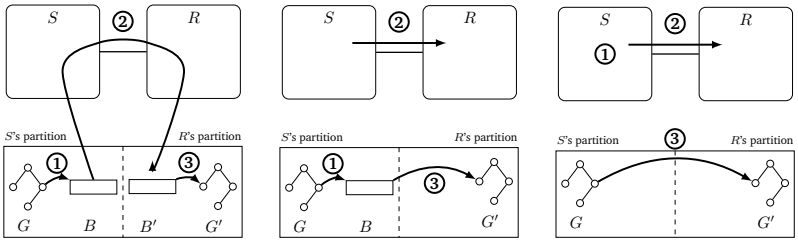
To support this unified configuration and test the compiler as well as novel architectural hardware features, we also implemented an easily extensible TCPA simulator. Benchmarking and testing of the full compiler flow will be a major task of our 2017 work program in Project C3.

C3

Compiler Optimisations for Invasive Architectures

Inter-tile data transfers We continued our work on accelerating inter-tile data transfers on our non-cache-coherent tiled architecture [MT17]. One particular challenge on our platform is optimising programs written in the modern, object-oriented programming language X10. While X10 makes it easy to distribute work over multiple coherence domains by providing built-in support for message passing, X10 programs, due to their high-level nature, often transfer *pointered* data structures.

However, message passing of pointered data structures entails costly (de-)serialisation. Consider the situation that tile S has a linked list in its memory partition and wants to send it to tile R. Tile S must first convert the list to a format suitable for message passing, i. e. serialise it to a byte stream, which R then receives to reconstruct (deserialise) a copy of the original list. The (de-)serialisation causes a large overhead, both memory-wise and computation-wise. As such pointered data structures occur frequently in general-purpose applications, especially if written



(a) Message passing. **(b)** Message passing via shared memory. **(c)** Cloning.

Figure 4.37: Schematic comparison of approaches to transfer an object graph G from sending tile S to receiving tile R. Temporary buffers are denoted by B, B'; and G' is the resulting copy of G.

in high-level object-oriented languages like X10, it is important to accelerate their transfer.

Data transfers can be accelerated on non-cache-coherent systems by exploiting shared memory and managing coherence in software, i. e. explicitly triggering the needed cache write-backs and invalidations. However, existing techniques for transferring data in this fashion were limited to contiguous "flat" data structures not containing any pointers. Hence, in our setting, most data transfers would still require (de-)serialisation.

C3

Therefore, we proposed, implemented, and evaluated a novel approach for transferring pointered data structures between shared memory partitions without requiring coherent caches. In our approach, the receiver directly accesses the data structure in the sender's memory partition and makes a deep copy of it, i. e. clones it, in the receiver's partition, thereby avoiding the need for serialisation and temporary buffers. To guarantee correctness, the software forces the necessary cache write-backs and invalidations with object granularity. Figure 4.37 shows a schematic comparison of our new data transfer approach compared to existing techniques. We showed that in a programming language following the partitioned global address space (PGAS) model, such as X10, the compiler and runtime system can issue the cache operations safely and fully automatically with zero overhead. Existing X10 programs do not have to be modified to benefit from our technique. We could show using the standard X10 benchmark suite IMSuite that our approach reduces the time spent for communication by up to 39.8% compared to the state-of-the-art.

Moreover, we found that both existing approaches and our new technique benefit greatly from range-based cache operations (called range operations in the following), i. e. hardware-supported operations that write back, invalidate, or flush all cache lines relevant to a given (logical) address range. This functionality has been requested before in prior work on software-managed cache coherence, however, so far it was never explored in the context of a non-cache-coherent architecture.

In cooperation with Project B1 we implemented novel non-blocking range operations. We implemented these operations as an instruction set extension to our LEON3 processors based on the SPARC V8 instruction set. In our design, the instructions offload the actual work to an enhanced cache controller. In every cycle, during which the processor does not execute a load or store, the cache controller uses this spare cycle to work on range operations. It therefore takes n spare cycles to apply an operation to a range spanning n cache lines. From the processor's view, these instructions only take one processor cycle. We could show that for our test inputs, this condition is fulfilled for the average data transfer.

Verified SSA construction During the first funding phase, we developed a simple and efficient SSA-construction algorithm. Since then, we formally verified the correctness of our algorithm [BLU16] using the theorem prover Isabelle/HOL. We also proved that the algorithm always constructs pruned SSA form, and minimal SSA form in case of reducible control flow graphs.

C3

In a follow-up work[12], we extend the minimality proof to irreducible control flow graphs. Thus, our algorithm always constructs minimal and pruned SSA form. We also show that minimal and pruned SSA form leads to a minimal number of phi functions. Furthermore, we provide a new criterion for minimal SSA form that coincides with the existing one but is more general and easier to check.

Synthesised instruction selection In order to rapidly support new special instructions provided by Project B1, we developed an instruction selection synthesiser for the libFIRM back ends[13]. The synthesiser uses a new program synthesis algorithm to discover FIRM patterns for the new instructions. This algorithm greatly improves performance over existing

[12] M. Wagner. "Minimal Static Single Assignment Form". Master thesis. Karlsruhe Institute of Technology (KIT), IPD Snelting, Nov. 2016.

[13] A. Fried. "Synthesizing Instruction Selection". Master thesis. Karlsruhe Institute of Technology (KIT), IPD Snelting, Aug. 2016.

approaches. Therefore, it does not need hints to restrict its search space, and can run unsupervised.

Our implementation is capable of covering the integer subset of the x86 instruction set, as well as some instructions from the BMI extension. Thus, it easily covers the SPARC integer instructions, as they are less complex compared to the x86 ones. Furthermore, it demonstrates that the approach can handle small and mid-size special instructions.

Information Flow Control For Invasive Applications

G. Snelting's group develops JOANA, a tool for software security analysis (information flow control, IFC). JOANA is one of the few IFC tools worldwide that handles full Java with unlimited threads, and provides high precision (few false alarms) [GHMS16]. Recently, a new *(i)RLSOD* algorithm was developed for JOANA, which discovers all probabilistic leaks, while still allowing secure parallel execution of public ("low") statements [Bre+16][14].

It is planned to use JOANA also for invasive X10 programs. An X10 front end is under development. RLSOD requires a probabilistic scheduler, as well as sequential consistency; the latter will be provided by the invasive memory model (see Project A1). On December 1st, a meeting with the group of Felix Freiling explored how JOANA can be integrated with the hardware-based invasive security features investigated in Project C5.

Demonstrator Activities and Integration Work

After extensive debugging work by Project C3 and communication with Gaisler by Project Z2, we could identify a problem in a stock component as the cause of a long-standing bug in our system. The bug, whose symptom were incoherent caches, was confirmed by Gaisler who recommended a different configuration that solved our issue. Furthermore, we continued our investigation concerning various data corruption problems related to hardware-accelerated inter-tile data transfers. While some problems remain, the automated testing framework established by Project C1 shows noticeable improvements in system stability.

To assist in testing our system, we ported the existing benchmark

[14] D. Giffhorn and G. Snelting. "A new algorithm for low-deterministic security". In: *International Journal of Information Security* 14.3 (2015), pp. 263–287. DOI: 10.1007/s10207-014-0257-6.

suite IMSuite[15] to our tiled architecture. IMSuite contains 12 X10 benchmark kernels that mainly implement popular graph-based algorithms, such as computation of spanning trees or vertex colourings. These programs serve as valuable test cases whose complexity is situated between simpler test cases provided by Project C1 and considerably larger invasive applications provided by Project D3. Additionally, they also allow evaluating the performance of our system.

Furthermore, in cooperation with Project B1, we added support for the *i*-Core special instructions to the X10 compiler. Thus, programmers can now write X10 programs that exploit custom hardware accelerators. The first user of this functionality is Project D3 where special instructions accelerate the tsunami simulation SWE-X10. To fully exploit the hardware accelerators, it is crucial to move data to the *i*-Core's local memory, which it can access much faster than other memory types. Therefore, in cooperation with Project C1 and Project B1, we developed a DMA-based mechanism and software interface that enables an asynchronous prefetching of input data. We make this prefetching available in X10 by standard language means, i. e. the prefetching is presented to the X10 programmer like a normal activity that can be synchronised with using a regular `finish` block. We plan to continue development and evaluate SWE-X10 on such a heterogeneous manycore platform.

X10 Code Generation for Operating Points

C3

Project A4 investigates design-time characterisation techniques of invasive applications to determine optimised operating points that contain a set of claim constraints. The characterisation ensures various non-functional requirements, such as power. In Project C3, we implemented a transformation in Erlangen that generates X10 source code corresponding to the claim constraints described by the operating points (see project report of Project A4, 2015). Such a source code is crucial to ensure that the application satisfies the non-functional program properties analysed at design time.

Publications

[Bha+16] V. Bhadouria, A. Tanase, M. Schmid, F. Hannig, J. Teich, and D. Ghoshal. "A Novel Image Impulse Noise Removal Algorithm Optimized for Hardware Accelerators". In: *Journal of Signal*

[15] S. Gupta and V. K. Nandivada. "IMSuite: A Benchmark Suite for Simulating Distributed Algorithms". In: *Journal of Parallel and Distributed Computing* 75 (2015), pp. 1–19. DOI: 10.1016/j.jpdc.2014.10.010.

Processing Systems (Nov. 1, 2016). DOI: 10.1007/s11265-016-1187-5.

[Bre+16] J. Breitner, J. Graf, M. Hecker, M. Mohr, and G. Snelting. “On Improvements Of Low-Deterministic Security”. In: *Principles of Security and Trust (POST)*. Ed. by F. Piessens and L. Viganò. Vol. 9635. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2016, pp. 68–88. DOI: 10.1007/978-3-662-49635-0_4.

[BLU16] S. Buchwald, D. Lohner, and S. Ullrich. “Verified Construction of Static Single Assignment Form”. In: *25th International Conference on Compiler Construction*. Ed. by M. Hermenegildo. CC 2016. ACM, 2016, pp. 67–76. DOI: 10.1145/2892208.2892211.

[GHMS16] J. Graf, M. Hecker, M. Mohr, and G. Snelting. “Tool Demonstration: JOANA”. In: *Principles of Security and Trust (POST)*. Ed. by F. Piessens and L. Viganò. Vol. 9635. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2016, pp. 89–93. DOI: 10.1007/978-3-662-49635-0_5.

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. “Providing Fault Tolerance Through Invasive Computing”. In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[MT17] M. Mohr and C. Tradowsky. “Pegasus: Efficient Data Transfers for PGAS Languages on Non-Cache-Coherent Many-Cores”. In: *Design, Automation Test in Europe Conference Exhibition (DATE)*. 2017. forthcoming.

[Tan+16] A. Tanase, M. Witterauf, Éricles R. Sousa, V. Lari, F. Hannig, and J. Teich. “LoopInvader: A Compiler for Tightly Coupled Processor Arrays”. Tool Presentation at the University Booth at Design, Automation and Test in Europe (DATE), Dresden, Germany. Mar. 14–18, 2016. URL: https://www.date-conference.com/system/files/file/date16/ubooth/37913.pdf.

[Tei16] J. Teich. “Invasive Computing – Editorial”. In: *it – Information Technology* 58.6 (Nov. 24, 2016), pp. 263–265. DOI: 10.1515/itit-2016-0043.

[WTHT16] M. Witterauf, A. Tanase, F. Hannig, and J. Teich. “Modulo Scheduling of Symbolically Tiled Loops for Tightly Coupled Processor Arrays”. In: *Proceedings of the 27th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. London, United Kingdom: IEEE, July 6–8, 2016, pp. 58–66. DOI: 10.1109/ASAP.2016.7760773.

C5: Security in Invasive Computing Systems

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

Ruan de Clercq, Gabor Drescher, Johannes Götzfried, Pieter Maene,
Tilo Müller, Furkan Turan, Alexander Würstlein

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.

Project C5 is a new project established within the second funding phase. In 2015, we began to devise the attacker model and study security properties (especially isolation properties) as constraints for invasive computing (work package 1). In 2016, we achieved different levels of isolation at different architectural layers. We focused on hardware (i. e. isolating applications running on the same core, work package 4) and the systems software layer (isolating memory abstractions in collaboration with Project C1, work package 3). Furthermore, in collaboration with Project A1, Project A4, and Project B5 we proposed spatial isolation as a countermeasure to mitigate side-channels with a new design-time/run-time mapping of applications.

Attacker Model

Our attacker model consists of four hierarchic 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

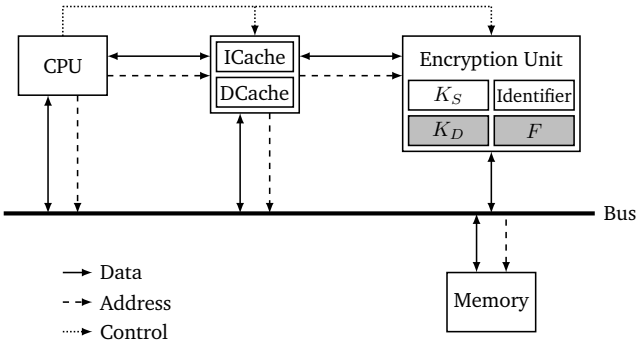


Figure 4.38: Encryption unit inserted between cache and main memory to guarantee confidentiality even in the event of a full (software-based) system compromise.

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 currently not considered in the project.

Results

Recently, we have been investigating different solutions to isolate applications from each other and preserve their integrity and confidentiality at any time. One such approach is memory encryption that can be dynamically configured to guarantee security on demand for invasive applications. For example, we propose the use of a new encryption unit (Figure 4.38) between the cache and main memory to guarantee confidentiality even in the event of a complete software compromise. While strong isolation guarantees can be given on a intra-tile level, at least confidentiality is provided on a inter-tile level, e. g. for global memory.

In the same direction, we developed a solution [Göt+16] that allows unmodified processes to transparently work on encrypted data. It can be deployed and enabled on a per-process basis without recompiling user-mode applications. In every enabled process, data is only stored in cleartext the moment it is processed, and otherwise remains encrypted in main memory. In particular, the required encryption keys do not reside in main memory, but are stored in CPU registers only. Hence, our solution effectively thwarts memory disclosure attacks, which grant unauthorised access to process memory, as well as physical attacks. In the default configuration, only up to 4 memory pages are exposed in cleartext at the same time.

Apart from process memory, kernel memory often contains sensitive data as well. Thus, we presented a hypervisor-based solution [GDPM16] that encrypts the entire kernel and user space to protect against attacks on main memory. This solution is fully transparent to the guest operating system and all applications running on top of it. At any time, only a small working set of memory pages remains in clear while the vast majority of pages are constantly kept encrypted. By utilising CPU-bound encryption, the symmetric encryption key is never exposed to main memory. With the default configuration of 1024 cleartext pages, successful attacks are rendered highly unlikely due to large caches on the invasive platform.

In some scenarios not even the hypervisor can be trusted. Therefore we introduced a hardware-based mitigation [WGGM16] against memory disclosure attacks. Our FPGA-based prototype with accompanying software components demonstrates the viability, security and performance of our novel approach for partial main memory encryption via memory proxies. The memory proxy approach is compared to other existing mitigation techniques, and possible further uses beyond encryption are discussed, as well. We effectively protect against attacks on main memory, while being transparent to applications and the operating system after initialisation.

Since only a part of the software stack usually needs to be protected, and kernel components are especially at risk, we developed a novel approach on isolating operating system components [RGM16]. Although existing trusted computing solutions have not been designed to work in kernel mode, we found a way of wrapping kernel functionality within secure containers by moving parts of it to user space. Kernel components are strictly isolated from each other such that a vulnerability in one kernel module cannot be escalated to a full kernel compromise. Besides integrity our implementation ensures that the confidentiality of the secure container is protected against all software level attacks as well as physical attacks.

C5

Logical isolation between applications is not enough when it comes to side-channel attacks. Thus we provided a spatial isolation scheme with the help of novel design-time/run-time mapping [Wei+16]. Different applications concurrently running on modern MPSoCs can interfere with each other when they use shared resources. This interference can cause side channels, i. e. sources of unintended information flow between applications. To prevent such side channels, we propose a hybrid mapping methodology that attempts to ensure *spatial isolation*, i. e. a mutually-exclusive allocation of resources to applications in the MPSoC. As a first step, we compute compact and connected application

mappings (called *shapes*) at design time. In a second step, run-time management uses this information to map multiple spatially segregated shapes to the architecture. Besides spatial isolation, we also took first steps in securing interconnects [Tur+16].

Summary

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. We argue that all our solutions for isolating applications by logical means as well as by design-time/run-time mapping can provide on-demand security for the invasive platform [Dre+16].

Outlook

We plan to map the existing landscape of trusted computing architectures to further refine our solutions and design the best possible solution for the invasive platform. Attackers target many different types of computer systems in use today, exploiting software vulnerabilities to take over the device and make it act maliciously. Reports of numerous attacks have been published, against the constrained embedded devices of the Internet of Things, mobile devices like smartphones and tablets, high-performance desktop and server environments, as well as complex industrial control systems. Trusted computing architectures give users and remote parties like software vendors guarantees about the behaviour of the software they run, protecting them against software-level attackers. We aim to define the security properties offered by current architectures. Our goal is to present detailed descriptions of selected hardware-based attestation and isolation architectures from academia and industry. We want to compare those designs with respect to the security properties and architectural features they offer.

Based on our results of systematically examining existing trusted computing architectures, we aim at developing a scalable approach for current heterogeneous MPSoCs. We believe that a hardware-based mechanism guaranteeing confidentiality of code and data between applications can be implemented based on the SPARC LEON3 processor, including tool-chain extensions for developers. Its overhead should be independent of the number of applications running on the system in order to support the heavy load of manycore systems. In addition, it has to be compatible with the complex memory hierarchies of these

systems, requiring multi-level protection mechanisms, giving developers the choice to enable them depending on their security requirements. Finally, a zero-software trusted computing base is required in order to protect against system-level attackers. Since these mechanisms will be integrated at the level of a single core, their area overheads should be small. Furthermore, for them to be used by application developers, they should have limited impact on performance.

Finally, in the future we plan to additionally look into exploit mitigation mechanisms such as control flow integrity solutions [Cle+16]. We plan to propose a hardware-based security architecture which protects software running on the invasive platform by guaranteeing software integrity and control flow integrity. This will allow the multicore platform to defend against a large number of attacks, including code injection, code reuse, and fault-based attacks on the program counter. In addition, we also want to defends against software copyright infringement and reverse engineering. All protection mechanisms will be enforced in hardware using cryptographic techniques.

Publications

[Cle+16] R. de Clercq, R. de Keulenaer, B. Coppens, B. Yang, P. Maene, K. de Bosschere, B. Preneel, B. de Sutter, and I. Verbauwhede. "SOFIA: Software and Control Flow Integrity Architecture". In: *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. Dresden, Germany: IEEE, 2016, pp. 1172–1177.

[Dre+16] G. Drescher, C. Erhardt, F. Freiling, J. Götzfried, D. Lohmann, P. Maene, T. Müller, I. Verbauwhede, A. Weichslgartner, and S. Wildermann. "Providing Security on Demand Using Invasive Computing". In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 281–295. DOI: 10.1515/itit-2016-0032.

[GDPM16] J. Götzfried, N. Dörr, R. Palutke, and T. Müller. "HyperCrypt: Hypervisor-based Encryption of Kernel and User Space". In: *11th International Conference on Availability, Reliability and Security (ARES'16)*. Ed. by S. Research. Salzburg, Austria: IEEE, 2016. URL: https://www1.cs.fau.de/hypercrypt.

[Göt+16] J. Götzfried, T. Müller, G. Drescher, S. Nürnberger, and M. Backes. "RamCrypt: Kernel-based Address Space Encryption for User-mode Processes". In: *11th ACM Asia Conference on Computer and Communications Security (ASAICCS)*. (Xi'an, Shaanxi, China). Special Interest Group on Security, Audit and Control (SIGSAC). ACM, 2016. DOI: 10.1145/2897845.2897924. URL: https://www1.cs.fau.de/ramcrypt.

[RGM16] L. Richter, J. Götzfried, and T. Müller. “Isolating Operating System Components with Intel SGX”. In: *1st Workshop on System Software for Trusted Execution (SysTEX'16)*. Trento, Italy: ACM, 2016. DOI: 10.1145/3007788.3007796. URL: https://www1.cs.fau.de/sgx-kernel.

[Tur+16] F. Turan, R. de Clercq, P. Maene, O. Reparaz, and I. Verbauwhede. “Hardware Acceleration of a Software-based VPN”. In: *26th International Conference on Field Programmable Logic and Applications (FPL'16)*. Lausanne, Switzerland: IEEE, 2016, pp. 1–9. DOI: 10.1109/FPL.2016.7577321.

[Wei+16] A. Weichslgartner, S. Wildermann, J. Götzfried, F. Freiling, M. Glaß, and J. Teich. “Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs”. In: *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*. (Sankt Goar, Germany). ACM, May 23, 2016, pp. 153–162. DOI: 10.1145/2906363.2906370.

[WGGM16] A. Würstlein, M. Gernoth, J. Götzfried, and T. Müller. “Exzess: Hardware-based RAM Encryption against Physical Memory Disclosure”. In: *Architecture of Computing Systems (ARCS'16)*. Nuremberg, Germany: Springer, 2016. DOI: 10.1007/978-3-319-30695-7_5. URL: https://www4.cs.fau.de/~arw/exzess.

D1: Invasive Software–Hardware Architectures for Robotics

Tamim Asfour, Walter Stechele

Manfred Kröhnert, Dirk Gabriel

The main research topic of Project D1 is the exploration of benefits and limitations of invasive computing in humanoid robotics. Our goal is to demonstrate invasion mechanisms and negotiation for resources in the context of complex robotic scenarios with concurrent processes and timely varying resource demands. Invasive computing mechanisms should allow for efficient resource usage and shorter execution times while adhering to predictability of non-functional properties, e. g. power, dependability, security. Therefore, research on techniques of self-organisation are key to efficient allocation of available resources in situations where multiple applications bargain for the same resources.

In 2016, we enhanced our resource-aware motion planning algorithm towards a fully malleable application and combined environment knowledge with resource utilisation statistics generated from profiling data into context-sensitive resource models. Furthermore, we investigated how a chain of vision algorithms behaves on different resource patterns.

Context-Sensitive Resource Models

The execution of a robot action is influenced by the internal state of the robot and the environmental context in which the action is performed. Context dependent parameters are execution time, number and type of selected algorithms, or CPU and memory utilisation. For example, grasping an object in a cluttered environment takes longer and requires more CPU and memory than grasping an object from an empty table. In the cluttered scene, all visible objects must be recognised by comparing them against a database of known objects and additionally, a large number of objects in a scene leads to longer motion planning execution times since each additional object increases the possibility of collisions.

The developed resource model stores statistics about execution time, CPU and memory utilisation for each robot action and includes all action

specific algorithms executed by so called robot components [Krö16]. Furthermore, these statistics are associated with the environmental context the robot was operating in while data was collected. This allows distinguishing which context can lead to changes in execution time or resource utilisation. Figure 4.39 shows CPU and dynamic memory utilisation of the resource model for place object on a table action. This action has a mean duration of 45 seconds and CPU utilisation during this action is high for the HeadIK, ViewSelection and RobotStateComponent components of the robot program. These components are required for calculating the kinematics of the head and where the robot should look at. Furthermore, CPU time is also required for sensor processing components (ForceTorqueUnit, ForceTorqueObserver) as well as components required for moving the robot arm to its target position (SystemObserver, KinematicUnitObserver, XMLStateComponentVisualServoGroup).

The right part of Figure 4.39 shows how much dynamic memory is used by the running robot components. Except for the RobotControlUnit, the TCPControlUnit, and the XMLSTateComponentPlaceObjectGroup, most components do not allocate large quantities of additional memory. The XMLStateComponentPlaceObjectGroup is responsible for coordinating all subtasks of the robot action, while the RobotControlUnit continuously receives sensor value updates and updates the internal representation of the robot accordingly. Allocation in the TCPControlUnit is performed by the intermediate calculations required to steer the robot arm towards the object placement position.

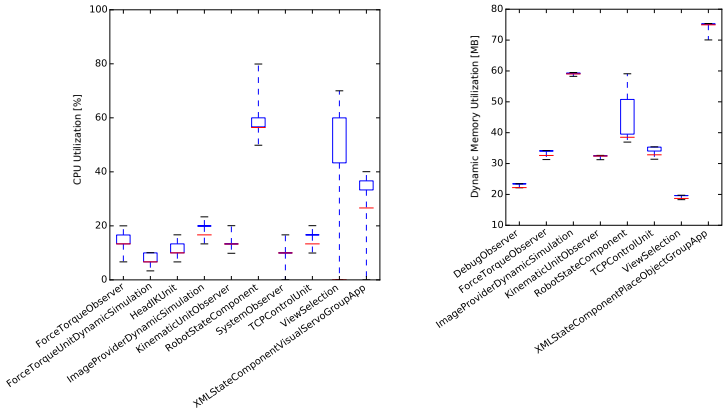


Figure 4.39: The left graph shows CPU utilisation while the right graphs shows dynamic memory utilisation of components required by the robot for placing an object on the table.

Malleable Resource-Aware Motion Planning

Motion planning is used in robotics to calculate collision free motions for a robot in a given environment. Figure 4.40 shows a typical motion planning in which the robot ARMAR-III must pull a bottle out of a crate while avoiding collision with any other bottle and the crate itself. The depicted task is difficult to compute, due to the narrow free space in the crate, where minimal movements of the robot arm result in collisions.

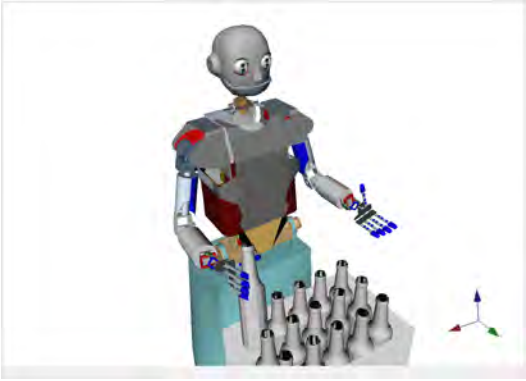


Figure 4.40: The goal for the humanoid robot ARMAR-III is to pull a bottle out of the crate without colliding with any other bottles or the crate.

In [KGVA16] we presented a parallelised resource-aware motion planning algorithm which is capable of adaptively requesting additional processing resources based on planning problem difficulty. One key aspect of the algorithm is a more efficient resource utilisation as compared to static allocation. To make this algorithm fully resource-aware, we enhanced it to support releasing resources when requested from an external resource manager. The resulting adaptive utilisation of processing resources is shown in Figure 4.41 which is based on an exemplary execution of the pull bottle out of the crate planning task. Starting at -220 seconds, the algorithm gradually acquires a maximum of 8 CPUs, each being fully utilised and resulting in a CPU utilisation of 800 %. The external resource manager requests the algorithm to release 6 CPUs after 70 seconds, allows using 5 CPUs after 100 seconds, allows using only one CPU after 120 seconds, and so on.

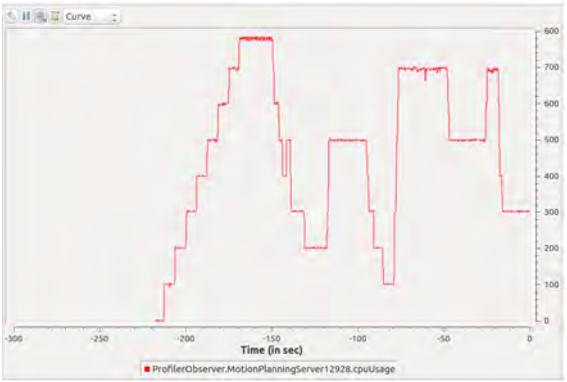


Figure 4.41: Adaptive CPU utilisation in [%] of the resource-aware motion planning algorithm. The task is to plan a motion for pulling a bottle out of a crate. After startup the algorithm gradually acquires new CPUs, each accounting for a rise of 100 % CPU utilisation. An external resource manager continuously updates the number of maximum allowed CPU resources and the algorithm responds accordingly by reducing its internal degree of parallelism and releasing CPUs.

Application Chains

As previously shown the time required to execute a robot action depends on the environmental context. Although the variation is low for the most basic algorithms the set of available resources highly influence the execution time whereby the system must guarantee a limited processing delay to ensure correct behaviour. We investigated one visual object detection algorithm based on SIFT (Scalable Invariant Feature Transform) features in detail. This algorithm consist of three concurrent stages (Harris Corner, SIFT extraction and SIFT matching) which can be parameterised to reduce the computational complexity while decreasing the result's quality. During runtime the system has to automatically select the overall best available configuration for the three stages including the mapping of workload onto the different resources and the stage parameters. The goal is to achieve the highest possible quality within the given timing and resource constraints. Further non-functional constraints like power and security may be included later.

As the requested configuration can be found by a computational intensive optimisation we will use results from the Design Space Exploration (DSE) developed by Project A1. During runtime the application control will select a suitable operating point from the Pareto front, which contains the full configuration.

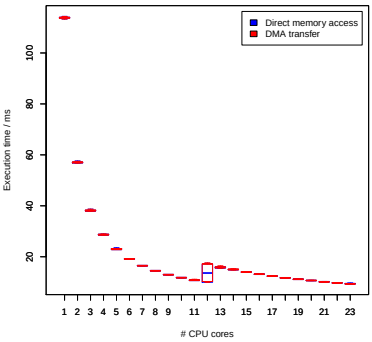


Figure 4.42: Execution Time of the Harris Corner stage on a single socket

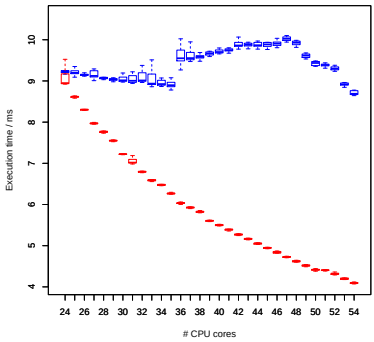


Figure 4.43: Execution Time of the Harris Corner stage on multiple sockets

In order to provide the required behavioural description for the DSE, an implementation of the algorithm supporting different communication patterns and resource distributions based on OctoPOS x86_64 has been finished in 2016. Figure 4.42 shows the relationship between the total execution time of the Harris Corner stage and the number of available CPU cores on a single socket. The workload is distributed equally on all cores. From one to eleven cores the execution time scales nicely with the increasing core count. Afterwards not only physical cores are used, but virtual cores using the Intel Hyper-Threading technology. They share the ALU with a physical core and therefore interfere with another *i*-let which results in a higher execution time. This issue can be solved by adapting the workload of the different *i*-lets.

When the cores belong to different sockets, the data must be transferred from one NUMA domain to another. As in Figure 4.43 shown an explicit DMA data transfer has an higher overhead than accessing the data beyond domain borders.

Outlook

In the next year, we will start integrating our work into a more complex demonstration scenario for the review meeting in 2018. Resource models generated from observation of robot actions will build the basis for predicting future robot actions and their associated resource utilisation. This prediction information can then be used by the agent system within OctoPOS for enhanced resource allocation decisions and

to start algorithms such as the resource-aware motion planning ahead of time. Furthermore, we plan a combination of invasive computer vision algorithms and the resource-aware motion planning where the amount of free resources allocated to motion planning is determined by the workload of the vision algorithms.

Publications

[Krö16] M. Kröhnert. “A Contribution to Resource-Aware Architectures for Humanoid Robots”. Dissertation. High Performance Humanoid Technologies (H2T), KIT-Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany, July 22, 2016.

[KGVA16] M. Kröhnert, R. Grimm, N. Vahrenkamp, and T. Asfour. “Resource-Aware Motion Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. (Stockholm, Sweden). May 2016, pp. 32–39. DOI: 10.1109/ICRA.2016.7487114.

[Wäc+16] M. Wächter, S. Ottenhaus, M. Kröhnert, N. Vahrenkamp, and T. Asfour. “The ArmarX Statechart Concept: Graphical Programming of Robot Behaviour”. In: *Frontiers in Robotics and AI* 3.33 (2016). DOI: 10.3389/frobt.2016.00033.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

D3: Invasion for High-Performance Computing

Hans-Joachim Bungartz, Michael Gerndt, Michael Bader

Isaías Comprés, Ao Mo-Hellenbrand, Josef Weidendorfer

The overall goal of Project D3 lies in two areas: The first one is to investigate and exploit the potentials of invasive computing for state-of-the-art high-performance computing applications[16,17] on standard HPC architectures [LGG16]. The second area is to provide application-level support for the development of invasive computing hardware platforms.

Invasion for HPC

Significant results have been achieved in the infrastructure required to support invasive MPI applications in HPC systems. Our infrastructure consists of a batch scheduler, a runtime scheduler, a collection of daemons running on each managed node, a launcher for MPI applications, the PMI library that interfaces MPI processes with the resource manager, and finally, the MPI library itself. We described each of these components and their organisation in our previous report.

From the components enumerated before, all are now functional with the exception of the batch scheduler and the scheduling plugin for the runtime scheduler. The MPI library in particular has reached feature completion and stability. The MPI library together with the current implemented components of the infrastructure provide an already stable enough platform for invasive MPI applications to be developed and run. In addition to this, the resource manager is already able to handle multiple applications and shuffle resources among them with a random algorithm. We can now perform application runs and measurements in the SuperMUC petascale system, by dynamically configuring our

[16] M. Bader et al. "Invasive Programming as a Concept for HPC". In: *Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks 2011 (PDCN)*. Feb. 2011. DOI: 10.2316/P.2011.719-070.

[17] M. Gerndt et al. "An Integrated Simulation Framework for Invasive Computing". In: *Proceedings of the Forum on Specification and Design Languages (FDL)*. Vienna, Austria: IEEE, Sept. 18–20, 2012, pp. 209–216.

infrastructure inside of a Load Leveler allocation. With this technique we gathered the required data for our publication at the EuroMPI 2016 [CMHGB16] conference in Edinburgh, Great Britain.

Invasive Resource Management (iRM) The main achievement of our infrastructure development efforts is the latency hiding algorithm for the adaptation of individual invasive MPI applications. Our infrastructure is now capable of hiding the scheduling and process creation latencies from preexisting application processes. Additionally, adaptations can be performed in groups where resources are reallocated in a collection of applications asynchronously and concurrently. This functionality will allow future schedulers to achieve important objectives such as power level stabilisation by minimising idle time in nodes.

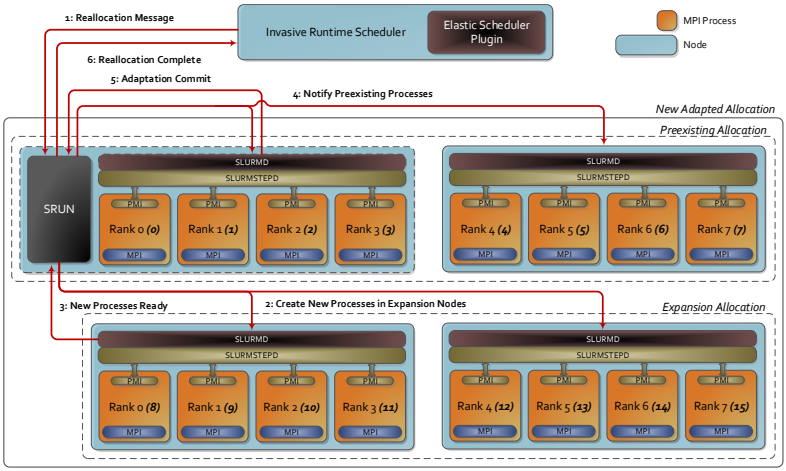


Figure 4.44: Step by step resource adaptations between the resource manager and MPI application.

We have divided the adaptation of a single application into six steps: 1) *reallocation message*, 2) *create new processes in expansion nodes*, 3) *new processes ready*, 4) *notify preexisting processes*, 5) *adaptation commit*, and 6) *reallocation complete*. Figure 4.44 illustrates these steps. The arrows indicating which components participate during each step, point from the component that initiates the action towards the components that performs it.

Invasive MPI (iMPI) The Invasive MPI library has reached feature completion and stability. This means that future development only remains in the infrastructure, mainly related to our schedulers. The MPI library and key parts of the infrastructure provide a reliable software stack for our invasive MPI application developers today.

Our finalised API consists of the same original 4 operations introduced in earlier reports. Their final API is now in place, and is described in listings 4.1.

```
int MPI_Init_adapt( int *argc, char ***argv, int *local_status);
int MPI_Probe_adapt(int *current_operation, int *local_status,
                    MPI_Info *info);
int MPI_Comm_adapt_begin( MPI_Comm *intercomm, MPI_Comm *new_comm,
                    int *staying, int *leaving, int *joining);
int MPI_Comm_adapt_commit();
```

Listing 4.1: C MPI extensions.

We have observed interest by some members of the MPI forum. We are in conversations on a possible presentation about our extensions at the MPI forum next year. There is potential that our work will influence the Dynamic Processes support of the next MPI standard.

Invasive HPC Applications Last year, we have finished the development of a *Statistical Inverse Problem*, in which the locations of multiple obstacles in a fluid channel are recovered. This application represents a class of Bayesian inference problems, which belong to the *embarrassingly parallel* application category and is inherently suitable for the invasive HPC framework. This year, we have devoted efforts to the development of following applications:

- *Invasive Tsunami Simulation*

  The tsunami simulation befit the invasive HPC framework very well, due to the fact that it is based on adaptive triangular grids, which produces dynamic compute loads and requires changing computational resources during runtime. It represents a class of grid-based simulation problems, in which data dependencies exist among parallel processes. This type of problems poses a major challenge for invasive computing: whenever there is a change in resources, the computation domain must be re-partitioned, which naturally adds overhead to the total resource adaptation time. Therefore, an efficient communication scheme for data redistribution and the frequency of resource adaptation must be carefully chosen.

Taken such consideration into account, we are able to achieve an efficient invasive implementation based on an existing tsunami simulation application [MRB16]. This invasive version probes the resource manager for adaptation instruction at an appropriate and adjustable frequency. In case of adaptation, MPI communication due to data re-distribution is reduced to minimum such that the overall performance is not comprised. We are currently carrying out more performance analysis on SuperMUC.

- *Invasive Porous Media Flow Simulation*

  The porous media flow simulation simulates the behaviour of fluids flowing through a porous medium, such as oil infiltrating into soil. Similar to the tsunami simulation, this application is based on adaptive triangular grids and poses the same challenge described above. The difference, however, is that this application has one dimension more than the tsunami simulation. It is a 3-D simulation with two fully adaptive and one static dimensions, due to which the computational intensity is brought to another level. At the moment, we are modifying an existing implementation[18] to incorporate the resource adapting scheme with iMPI.

- *Elastic Machine Learning: Multivariate Regression*

  This application solves a typical machine learning problem—high dimensional regression. It is designed to befit the invasive HPC framework by utilising the sparse grid combination technique to decompose the original problem into a set of simpler, independent subproblems. The result is an embarrassingly parallel application that is tailored to the invasive HPC framework with a master-worker implementation. At the moment, the normal MPI implementation is completed. We are in the process of incorporating the resource adapting scheme with iMPI

D3

Demonstrator Platform: X10 Applications

In the area of MPSoCs, we continue our efforts on the development of different applications in X10 that could be run on the invasive multi-tile hardware demonstrator platform. These applications should be of practical value and be able to showcase the benefits of invasive computing. Our goal for the current funding phase is to demonstrate

[18] O. Meister and M. Bader. "2D adaptivity for 3D problems: Parallel SPE10 reservoir simulation on dynamically adaptive prism grids". In: *Journal of Computational Science* 9 (May 2015). Special Issue ICCS 2015, pp. 101–106.

on the InvasIC hardware platform design features such as reliability, predictability, flexibility, security, among others.

- A computational fluid simulation with arbitrary 2-D geometries is currently under development. This is a classical grid-based simulation application with an increased complexity and greater resource requirements compared to the previously developed heat simulation. A parallel version has been implemented in X10. An invasive extension is under development.

- A prototype of a malleable simulation framework utilising the sparse grid combination technique is under investigation. To fully demonstrate the flexibility of invasive computing, malleable applications are highly desirable. Experiments are being carried out on simulations with finite difference methods (FDM), since FDM is proven to be suitable for computations on sparse grids.

Outlook

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

- **HPC Development:** (1) Improve and extend the resource manager with *intelligence*, i. e. the resource manager is pre-trained with certain features such as application type, application priority, energy level, among others, such that it can make better decisions in shorter time. (2) Carry out rigorous performance analysis with multiple invasive applications for a combat scenario, as well as study its impact on the overall system throughput and energy consumption.

- **X10 Development:** contribute to the topic of *dark silicon* (Project B3) and make features accessible to scientific algorithms. Provide core-level fault-tolerant applications, such as an iterative solver for a linear system with full local storage of the right-hand-side data, or a simple solver using the sparse grid combination technique.

Publications

[Bak+16] A. Bakhtiari, D. Malhotra, A. Raoofy, M. Mehl, H.-J. Bungartz, and G. Biros. “A Parallel Arbitrary-Order Accurate AMR Algorithm for the Scalar Advection-Diffusion Equation”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. Salt Lake City, UT, USA: IEEE, 2016.

[CMHGB16] I. Comprés, A. Mo-Hellenbrand, M. Gerndt, and H.-J. Bungartz. “Infrastructure and API Extensions for Elastic Execution of MPI Applications”. In: *Proceedings of the 23rd European MPI Users' Group Meeting*. EuroMPI 2016. ACM, 2016, pp. 82–97. DOI: 10.1145/2966884.2966917.

[LGG16] W. Liu, M. Gerndt, and B. Gong. “Model-based MPI-IO tuning with Periscope tuning framework”. In: *Concurrency and Computation: Practice and Experience* 28.1 (2016), pp. 3–20. DOI: 10.1002/cpe.3603.

[MRB16] O. Meister, K. Rahnema, and M. Bader. “Parallel Memory-Efficient Adaptive Mesh Refinement on Structured Triangular Meshes with Billions of Grid Cells”. In: *ACM Transactions on Mathematical Software* 43.3 (2016), 19:1–19:27. DOI: 10.1145/2947668.

Z: Central Services

Jürgen Teich

Jürgen Kleinöder, Katja Lohmann, Sandra Mattauch, Ina Derr, Frank Hannig

The central activities and services in InvasIC are coordinated and organised by Project Z. These activities and services are subdivided into two parts:

The first part is administrative support, organisation of meetings (internal project meetings, doctoral researcher retreats) and assistance for visits of guest researchers and for researchers travelling abroad. Technical support and tools for communication and collaboration are provided as well as support and organisation of central publications. Last but not least, financial administration and bookkeeping is one of the central services.

The second part concerns public relations. Contacts with important research sites are established as well as an international Industrial and Scientific Board. Scientific ideas and results are discussed at various workshops and conferences.

For detailed information on the general idea and organisation of InvasIC as well as on the progress made in the different projects, the InvasIC website `http://www.invasic.de` is maintained.

A detailed listing of the scientific meetings and events organised and conducted by Project Z is provided in Part III of this report.

Z2: Validation and Demonstrator

Jürgen Becker, Frank Hannig, Thomas Wild

Marcel Brand, Stephanie Friederich, Leonard Masing, Sven Rheindt

The major goal of Project Z2 is to provide a common demonstrator environment for validating and demonstrating the principles of invasive computing. Basis for this demonstrator environment are multi-FPGA prototyping platforms at each site that can accommodate invasive MP-SoC architectures with dozens of processor cores of different type. In the current funding phase, Project Z2 acquired a new FPGA-based prototyping platform from ProDesign, which contains four Xilinx Virtex-7 2000T FPGAs. The ProDesign proFPGA platform has about four times more capacity than the Synopsys CHIPit Platinum system as used in the first funding phase, and thus, allows to implement larger designs of up to 64 or 80 standard RISC processors. The Synopsys CHIPit Platinum is still used for debugging and integration work on smaller scale invasive architectures allowing parallel work by several researchers. Both platforms are kept compatible with the same architectural description, exchanging only some required parts (i. e. DDR/SRAM controller, transactor, clocking, etc.).

For demonstrating the unique advantages of invasive computing, e. g. to enforce non-functional properties of programs, the contributions from all project areas, the developed concepts at the hardware and software (compiler, OS and application) level are integrated on this common demonstrator platform. In cooperation with all projects and the working groups of the CRC, our tiled invasive multicore architectures are continuously expanded and prototyped for demonstration of invasive applications from project area D.

An example configuration of an invasive multi-tile architecture as shown in Figure 4.45 consists of the following components: Several computing tiles, each with a set of of RISC processors (LEON3 processor cores or the *i*-Core as developed in Project B1), compute tiles based on TCPAs (Tightly-Coupled Processor Arrays) as developed in Project B2, as well as I/O and memory tiles. The tiles which contain contributions

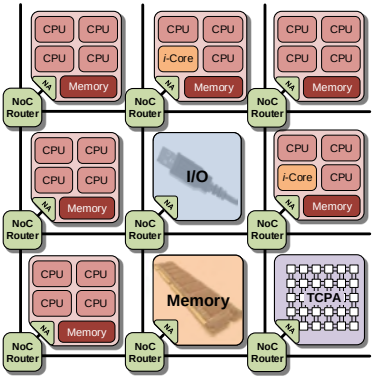


Figure 4.45: A 3×3 invasive multi-tile architecture consisting of seven compute tiles, as well as one memory and one I/O tile.

Figure 4.46: The four FPGAs inside the proFPGA system including the extension board setup.

from all projects of project area B (architecture) are connected via the *i*NoC as provided by Project B5. The operating system OctoPOS and the run-time support layer *i*RTSS (Project C1) are running on top of this hardware platform. Within Project Z2, different variants of such a heterogeneous hardware architecture, the associated system software, as well as a lot of invasive applications on top are made available for both prototyping systems.

Progress on the new Demonstrator Platform

A proFPGA system is available for integration, development and test at each site. For a consistent work environment and reusability of developed FPGA configurations, each proFPGA system has to be configured with extension boards. Figure 4.46 shows the extension board setup. The configuration of the upper left FPGA consists of two SRAM extension boards, one DDR memory card and one DVI extension board. All other FPGAs are only equipped with two SRAM extension cards. The shown setup enables on the one hand a configuration of a 2×2 tiled invasive architecture that can be used for demonstrations on a single FPGA, and on the other hand prepares the proFPGA platform for extended 4×4 tiled architectures.
The important hardware components for those architectures are explained in detail in the following:

- **SRAM:** The tile-local memory (TLM) for each tile consists of 8MB

SRAM memory that is available via extension boards. Each FPGA has access to two SRAM extension boards, each providing 24MB of SRAM memory. The extension boards were specifically designed by ProDesign for the requirements of our CRC. Each TLM is connected inside a tile over an AHB interface and is available in two versions: (a) a synchronous design that operates at the same frequency as the other parts within a tile, utilising a standard Gaisler memory controller, and (b) an asynchronous design in form of a custom memory controller that operates the SRAM at a much higher frequency than the other parts and prefetches data for faster burst transfers than version (a).

- **DDR:** Besides the TLM, an invasive multi-tile architecture employs large DDR global memory units, which can be accessed only over the NoC. In the current funding phase, Project Z2 has acquired DDR extension boards which are placed at specific positions in the design. To enable access to the DDR, Project Z2 incorporated a Xilinx DDR3 controller IP core into the invasive multi-tile architecture. The generation and pin assignment is fully automated and embedded in the synthesis scripting. Since the DDR requires high clock speeds to work reliably, a bridge to the DDR memory was introduced, allowing the rest of the design to run at a different speed from the DDR controller.

- **DVI:** The proFPGA system is able to receive and transfer DVI streams via an HDMI interconnection on an extension board. The extension board is connected by a custom controller as part of the I/O tile utilising an 8 MB SRAM bank of one of the SRAM extension boards as a double frame buffer. This buffer can be used to read in the data of a video stream, an RGB video frame, and making the whole frame available for a customised time to the system via the aforementioned asynchronous SRAM memory controller. It can also be used to write the data inside the frame buffer to the output stream of the DVI controller.

- **Tool flow:** The automation of a full tool flow for the proFPGA platform is now completed. The new tool flow is entirely based on Xilinx Vivado and can be started from a console in a scripted mode. Additionally, the scripts create a full Vivado project which can be opened in the graphical user interface to view results and support the debugging efforts with the available tools and views on the design.

- **Test suite:** The continuous integration of the architecture components is supported by the establishment of a test suite, i. e. a collection of binaries and test cases that can be automatically executed in batch mode on a prototype and, in the end, create statistics about the results. The automated scripting for the test suite was adapted to work on the proFPGA system and is now fully compatible with both systems.

Integration of components specific to invasive computing

In addition to the above mentioned work related to the migration to the new prototyping system, Project Z2 continued its contribution to the realisation of the overall CRC demonstrator platform. In the last year, there has been a lot of progress regarding the integration of components developed by the individual projects. The goal was to bring all contributions of the hardware projects into one large design and make each special feature selectable through a configuration file before synthesis. Project Z2 provided support for the integration by giving introductory and intensive discussions at the Doctoral Researcher Retreat with the colleagues of the involved projects, provided technical documentation in the Wiki and directly helped with integration, test, and verification of the designs. The integration of the following hardware components into the architecture and the tool flow on the proFPGA system has made significant progress:

***i*-Core** Project B1 fully integrated the *i*-Core into the provided tool flow and the latest set of invasive multi-tile designs. The integration was enabled and eased by the new tool flow for the proFPGA system and supportive work as well as successful testing was provided by Project Z2.

TCPA Project B2 and Project Z2 together integrated the TCPA into the provided tool flow and provide multi-tile designs. Tests to ensure the correct functioning and interfacing with the operating system OctoPOS were finished successfully in cooperation with Project C1. Moreover, a 2×2 architecture including one TCPA tile has been developed for testing hard real-time capabilities of invasive computing architectures.

Atlas Atlas is a hardware-based security architecture protecting code and data confidentiality even when the operating system has been compromised. A transparent encryption unit was added between the processor's cache and memory. This unit is controlled through custom

instructions which were added to the ISA. For this purpose, Project C5 developed a new library with all LEON3 components that required changes. As already possible for the *i*-Core, tiles can now be configured with a number of Atlas cores. Currently, the modified design can be synthesised using the new Vivado-based tool flow, and Project C5 are in the process of debugging the integrated design on the proFPGA system with the support of Project Z2. In addition, it is planned to make necessary changes to the software tool chain.

Testing and Debugging

The integration of an extended test suite exposed some communication bugs, especially a lot of corner cases, that have not been triggered by earlier tests. Therefore, a comprehensive test environment was designed and implemented, containing, amongst others, a Modelsim simulation of the full tiled architecture, which helped to trigger, locate and fix bugs. For solving these problems, Project Z2 worked together in close cooperation with Project B2, Project B5, and Project C1.

Outlook

In 2017, Project Z2 will focus on two major directions: (a) scaling of the invasive heterogeneous tiled architecture in terms of size and performance, and (b) setting up of demonstrators.

Scaling of the performance involves optimisation of critical paths inside the hardware architecture, decoupling parts of the architecture by connecting components asynchronously (e. g. memory controllers, NoC interfaces), as well as further improving the overall system stability. In addition, we want to expand the current 2×2 tile designs to 4×4 tiled architectures, comprising mixtures of 64–80 RISC cores as well as *i*-Core and TCPA tiles to utilise the full proFPGA system.

Together with all other projects of the CRC, Project Z2 will take care and support setting up demonstrators for showing the benefits of invasive computing in the second funding phase, namely predictability enforcement for hard real-time, fault tolerance, and security properties on demand of applications. Common demonstration scenarios were discussed in several WG3 meetings (see also corresponding section in this report), not only for public dissemination but also with regard to the review meeting in 2018. In more detail, the ability of invasive computing to isolate applications from each other shall demonstrate enforceable *timing predictability*, *fault tolerance*, and *security*. The heart of this demonstrator will be a computationally intensive cyber-physical

system with hard real-time requirements that will be controlled by an InvasIC architecture. To interface the proFPGA prototyping system to the physical world, we have started to build several I/O interfaces. In order to showcase fault-tolerant loop processing in TCPAs, as already theoretically and simulatively investigated by Projects B2, C2, and C3, we will design and integrate fault injectors into the prototyped invasive MPSoC architecture.

5 Working Groups

WG1: Predictability

Coordinators: Michael Gerndt, Michael Glaß

Focal points of interdisciplinary investigation in this working group are all questions around the current lead topic of the CRC: The *predictability* of non-functional aspects of parallel computation. The term **-predictability* was defined and used to express a current deficiency of today's multicore systems and parallel applications to provide a bounded (guaranteed) quality of the their execution—not only w. r. t. execution time, but also to security, reliability and/or power consumption. The topics of this working group stem from the fact that predictability is an all-encompassing concept that requires consideration across architecture, system software and services, up to the level of applications. Here, WG1 serves as a discussion group for these topics, organises information exchange, and triggers and coordinates collaborations between projects and project areas with respect to predictability.

In 2014 and 2015, WG1 could already identify the most *relevant topics* concerning *-predictability within this CRC, create a *glossary* of all relevant predictability terms to enhance the common understanding of predictability concepts, and provide a *predictability landscape* that outlines predictability challenges which (a) can be solved by employing existing analysis techniques, (b) are tackled within this CRC, and (c) may be subject to future research directions. The focus of WG1 in 2016 was set on two main topics: (I) Coordinate the aspect of predictability throughout the invasive computing design flow and (II) enhance the external perception of predictability topics covered by this CRC as well as *-predictability in general.

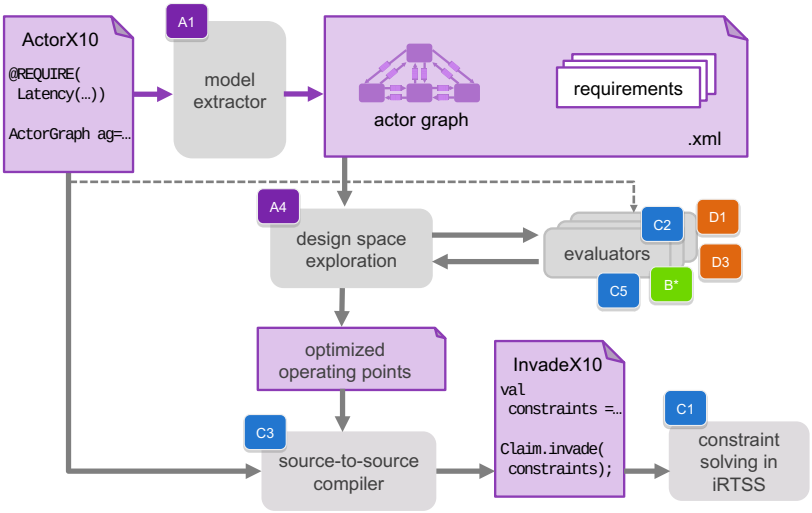


Figure 5.1: Design flow to support *-predictability in invasive computing: Starting with an application written in actorX10, an automatic model extractor gathers a graph-based specification that is suitable as an input for the design space exploration which itself employs various evaluators to determine the quality numbers such as latency or energy consumption of each operating point candidate. Given the set of optimised operating points—where each contains a constraint graph which encodes constraints how to feasibly embed the point at run time—a source-to-source compiler then transforms the constraint graph and other determined parameters into the InvadeX10 constraint language. Based on the latter, the invasive run-time support system then solves the respective constraints which delivers—if possible—a claim at run time where the operating point can be feasibly embedded.

Achievements

Figure 5.1 shows the invasive-computing design flow in the context of *-predictability as coordinated by this working group. In 2016, the following important steps to realise this design flow have been completed:

WG1

- In a collaborative work of Project C2, Project A4, and Project A1, an actor-oriented extension of the X10 programming language called actorX10 [Rol+16]—which was initially discussed at the doctoral-researcher course “Benchmarking for Multi-Criteria-Predictable Multi-Core Computing” at the Sarntal Academy in South Tyrol, 2015—has been implemented, published, and will be pub-

licly available soon. From such an actor-oriented specification, graph-based models and performance models as required by the design space exploration step can now be automatically extracted via respective transformations and tool support.

- Two transformations delivered by Project A1 and Project C3 now allow to (I) derive a graph-based exploration and performance model of an actorX10 application as required for the design space exploration step and (II) translate the constraint graphs which are the main result of the DSE to the InvadeX10 constraint system such that the requirements on *what* to claim at run time to embed a certain operating point is fully contained in the program itself.

The required interfaces for the design flow have been discussed at two physical meetings, at the 1st of June 2016 in Munich as part of a bigger WG3 meeting as well as at the 15st of September 2016 as part of the CRC annual meeting in Blaubeuren. At the latter meeting, one *-predictability demonstrator which has been initially projected at the Sarntal Academy in 2015 could already be presented to the CRC's industrial board in October 2016.

Activities

To enhance the public perception of this CRC's work on predictability and the concept of *-predictability in general, several activities have been initiated:

1. Two keynotes have been presented by Prof. Teich at (I) the 8th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO 2016), Prague, Czech Republic on "The Role of Restriction and Isolation for Increasing the Predictability of MPSoC Stream Processing" and (II) the 6th International Workshop on Polyhedral Compilation Techniques (IMPACT 2016), Prague, Czech Republic on "Symbolic Loop Parallelization for Adaptive Multi-Core Systems – Recent Advances and Benefits" with the latter focusing on the ability of tightly coupled processor arrays (TCPAs, Project B2) to create strictly predictable execution times for loop nests. Moreover, Prof. Teich gave a talk on "Adaptive Restriction and Isolation for Predictable MPSoC Stream Processing" at the Friday Workshop on "Resource Awareness and Application Autotuning in Adaptive and Heterogeneous Computing" of the Design, Automation and Test in Europe (DATE 2016) conference, Dresden, Germany.

2. In a special issue on Invasive Computing in it – Information Technology, guest edited by Prof. Teich, an overview article [Wil+16] that details how *-predictability is integrated in and realised by means of the invasive computing paradigm and design flow is presented. A significant part of this overview article is based on the coordinative work of WG1 to establish the outlined design flow.

3. Two Dagstuhl-Seminars have been organised with this CRC and in particular this working group as nuclei:
   - *Dagstuhl-Seminar 16052 "Dark Silicon: From Embedded to HPC Systems", January 31–February 3, 2016 [GGPR16]*. This seminar was organised and coordinated by PIs Prof. Gerndt, Prof. Henkel, and Prof. Glaß together with Prof. Sri Parameswaran, UNSW Sidney, and Dr. Barry L. Rountree, LLNL Livermore and discussed implications and challenges arising from dark silicon for both embedded and HPC systems. Amongst others, the topic of predictability w. r. t. performance, power consumption, and temperature gained special attention.
   - *Dagstuhl-Seminar 16441 "Adaptive Isolation for Predictability and Security", October 30–November 4, 2016*. This seminar was organised by PIs Prof. Teich and Prof. Verbauwhede together with Prof. Tulika Mitra, NUS, and Prof. Lothar Thiele, ETH Zurich and discussed methods for temporal and spatial isolation to enforce non-functional properties on timing predictability as well as security without sacrificing any efficiency or resource utilisation.

WG2: Memory Hierarchy

Coordinators: Lars Bauer, Gregor Snelting

In the first funding phase of InvasIC, cache coherence was *only* provided within a tile. In the current second funding phase, Project B5 proposed the work package "Dynamic and run-time-adaptive cache coherence", which essentially proposes to dynamically extend the size of a cache-coherence region beyond the boundaries of a single tile. For instance, it could be extended over two or three (possibly neighboured) tiles. As such an extension affects several aspects of the memory hierarchy (e. g. memory consistency) and other projects as well, the memory hierarchy working group WG2 provides a platform to discuss all aspects related to this topic, as well as other topics that are related to the memory hierarchy (e. g. about memory management; see below).

As in 2015, WG2 also organised a mini workshop in 2016 that took place in May 2016. All related projects participated in the meeting in Munich. This WG2 annual report summaries the results and current discussions that came up during that WG2 Meeting.

Inter-Tile Cache Coherence

The basic idea of this topic was already presented and discussed in the last annual report and can also be found in the description of Project B5. At this year's meeting, different implementation variants and protocols were presented by Project B5 to address the cache coherence of different tiles. During these discussions, the relevance of not *only* providing cache coherence, but of also providing mechanisms to ensure atomicity of dedicated atomic operations was emphasised. For clarity, an atomic operation is an operation (assembly instruction) that does multiple things as one inseparable operation. For instance consider a function `swap` as follows:

```
int swap(int* addr, int new) {
    int old = *addr;
    *addr = new;
    return old;
 }
```

If `swap` is an atomic operation, then that ensures that no other *i*-let can modify `*addr` *after* it was read by `swap` but *before* it is overwritten by it. Due to a rather flexible communication approach, envisioned by Project B5 to realise cache coherence between tiles, it was possible to add extensions for atomic operations as well.

For integration purpose, a language construct is needed that allows the application to express its desire to have multiple cache-coherent tiles. That is basically a constraint in the invade call. The agent system of Project C1 needs to support such a new constraint and plans to do so in the next year. Additionally, an interface between Project B5 and OctoPOS is needed that allows to configure two (or more) tiles to be cache-coherent, i. e. to activate the inter-tile cache coherency for those tiles that the agent system decided to make cache-coherent.

As potential application that could benefit from inter-tile cache coherence, we discussed a parallel multi-dimensional tree search algorithm, where it is not clear upfront which part of the tree will actually be accessed by which core during the search. So traditionally (i. e. without inter-tile cache coherence), either the entire tree needs to be copied to each core upfront (results in high overhead) or it needs to be copied as demanded (results in several individual small transfers which result in high latency). Instead, the inter-tile cache coherency can be activated and used to exchange exactly those message between the cores/caches that are needed.

MMU

We discussed the steps required to integrate an MMU into the current memory hierarchy. Hardware-wise, Gaisler provides a MMU for the LEON3, but it is known to have issues along with certain cache configurations. To avoid having on-going debugging challenges due to Gaisler cache configurations, we agreed that we would use a known-good MMU/Cache configuration, for instance one of those that is known to be functional by booting and running Linux on it.

WG2

When using an MMU, we can no longer use the scratchpad. To clarify: Gaisler uses the term *scratchpad* to denote a core-local memory, i. e. a small addressable SRAM per core that can only be accessed by the core

to which it belongs. No other bus device can access that scratchpad. That is not the typical definition of scratchpad used by the community. There, it would denote a relatively bigger addressable SRAM that can be accessed by all bus participants. To avoid any naming confusions, we call that bigger memory as tile-local memory (TLM). We will keep using TLM, but to use an MMU we have to disable the core-local scratchpad.

As a special case, the *i*-Core needs some additional treatment to support an MMU. This is due to its dedicated 2×128 bit ports from its reconfigurable fabric to the TLM. As these ports bypass the bus, they also bypass the regular MMU and thus addresses would go untranslated. Therefore, an extra MMU is needed to translate accesses via these ports. This special MMU is basically a simple address translation unit. On every address-space switch on the *i*-Core, the content of this special MMU has to be updated as well. OctoPOS will add support for this along with their MMU support. Project B1 implemented and integrated a corresponding code into Linux already and will provide that code as a basis.

X10 Memory Model

As a foundation for further formalisation, Project A1 developed a memory model for the X10 programming language. The memory model is an important part of the specification for the compiler, which must correctly map the language's memory model to the target architecture's memory model. This is essential for reasoning about the correctness of programs, especially when we are discussing inter-tile cache coherence. For example, we considered scenarios, where the compiler was responsible for some aspects of inter-tile cache coherence. However, the current state of discussion is that the hardware should maintain the same memory model for intra-tile as for inter-tile cache coherence (SPARC Total Store Order), such that no changes to program or compiler are necessary.

WG3: Benchmarking and Evaluation

Coordinators: Michael Bader, Walter Stechele

In 2016, WG3 had three main activities, starting in March with the DATE Friday Workshop, in June a joint WG1 and WG3 demonstrator workshop in Munich to plan common demonstrators for the TRR, and in September a demonstrator session for the Industrial Board at the annual meeting in Blaubeuren.

DATE Workshop The DATE Friday Workshop on Resource Awareness and Application Autotuning in Adaptive and Heterogeneous Computing has been co-organised by Cristina Silvano (PoliMilano), Walter Stechele (TUM), and Stephan Wong (TU Delft), with contributions from high performance computing arranged by Michael Bader (TUM). The workshop had invited talks from industry and academia and a poster session with 12 peer-reviewed contributions. Invited talks included an introduction to Invasive Computing by Jürgen Teich, contributions from High Performance Computing from the Leibniz Computing Centre (LRZ) and RSC, a leading Russian developer and integrator of HPC solutions, industry contributions from Xilinx and Bosch, university contributions from Axel Jantsch (TU Vienna) and João Cardoso (University Porto), as well as a panel featuring Sharon Hu (University of Notre Dame).

Demonstrator Workshop The goal of the joint WG1 and WG3 workshop in June was to identify a smaller set of common demonstrators, suitable to stimulate and illustrate cooperation between the projects, to identify further needs on the hardware and software side and to define show-cases for the benefit by the invasive programming paradigm, such as predictability. Four main common demonstrators have been discussed:

1. robotic visually guided grasping on Intel & proFPGA prototyping platforms,

2. visual control of an inverted pendulum on proFPGA prototyping platforms,

3. competition of InvadeX10 applications *fighting* for resources on proFPGA,

4. a design flow demo to illustrate the interaction of the invasive programming stack.

Additionally, there are project-specific demos, such as *i*-Core reconfiguration, TCPA error correction, dark-silicon management, monitors, cache coherence, memory protection, robotic motion planning, X10 multigrid, and invasive programming models (invasive MPI) and applications (e. g. tsunami simulation) for commodity HPC platforms.

Many project contributions, such as compilers and programming models, design space exploration, the actorX10 model, invasive hardware extensions, *i*RTSS, etc. (to name just a few) will be relevant for several of these common demonstrators. Hence, a crucial task for 2017 will be to determine in which of the demonstrators the impact of these invasive concepts can be evaluated in the best way. Tailoring and fine-tuning the demonstration scenarios will therefore stay the main focus of WG3, and define the work for 2017, already in preparation for the final review of second funding phase.

Demonstrator session Over the summer, substantial progress could be observed towards the demonstrators. For the meeting with the Industrial Board in September, ten demos could be shown, illustrating various results on timing predictability, security, power efficiency, and fault tolerance through invasive computing. The demonstrations covered a wide range of topics, including Design Space Exploration, inverted pendulum control, visual object tracking, robotic pick and place, multigrid combat, dark-silicon management, and security attacks. Related methods covered actorX10 models, invasive simulation and compilation, operating point selection, all based on OctoPOS running on X86 and on the FPGA platform.

WG4: Power Efficiency and Dark Silicon

Coordinators: Santiago Pagani, Frank Hannig

The focus of this working group is to align research problems in the direction of the *dark silicon* challenge.

The term *dark silicon* has been coined with respect to the increasing power density problem: since the classical Dennard scaling will be no longer applicable in upcoming silicon technology nodes, the power density, i. e. the amount of electrical power that is dissipated per chip area, increases drastically. In the past, the power density could be kept at tolerable levels since the increased amount of transistors per chip was (power-density-wise) compensated by lowering V_{dd}. This, however, is not possible any longer. Dark silicon denotes the problem of future multicore/manycore systems where a considerate amount of computing and/or communication resources needs to stay 'dark', i. e. unused in order not to exceed what is called the *Thermal Design Power* (TDP): this, in short, is the maximum amount of power that a chip can be operated at without suffering short or long term damage. As a matter of fact, dark silicon is becoming a severe problem for all future manycore systems where performance, predictability, and efficiency matter. Especially, in invasive computing, the dark-silicon problem will significantly matter because it is targeted towards high efficiency, e. g. how to make best use of on-chip computing and communication resources at the lowest cost possible (e. g. amount of consumed energy, cost of chip packaging for cooling, etc.). In other words: if the dark-silicon problem would not be addressed, invasive computing would lose its advantages compared to competing manycore systems. Hence, addressing the dark-silicon problem will enable the invasive computing paradigm to come ahead for the upcoming generations of technology nodes.

Towards this end, this working group aims at facilitating information exchange by building power and dark-silicon models for an InvasIC-wide power and dark-silicon estimation. Furthermore, it targets collecting results for power/dark-silicon estimation and agent knowledge as well as

investigate common possibilities for cross-level energy efficiency optimisations. Through integrated dark-silicon efforts across different projects, it is not only ensured that the (worst-case) thermal constraints of the on-chip computation and communication resources are not violated, but this also aids in the predictability of an invasive computing system as a shutdown of cores can be avoided, thereby satisfying the constraints of also applications with predictable execution requirements.

Because of its key importance, this working group enables information exchange across different projects to not only build a common understanding and nomenclature for power efficiency and the dark-silicon problem, but also facilitates integration of research efforts of various projects including Projects B2, B3, B4, C1, etc.

A summary of the key goals and targets of the WG, aligned to the road map of the current InvasIC-wide dark-silicon research activities, is listed below:

1. Aligning research problems to the dark silicon, power/energy efficiency, and temperature challenges.
2. InvasIC-wide dark-silicon modelling and estimation that will also require full system power modelling and estimation.
3. Interfacing and facilitate information exchange, for instance,
   a) How to pass the dark-silicon information/constraints to the agent-layer?
   b) How to interface between the monitoring (Project B4) and iDoC (Project B3)?
   c) Knowledge exchange on power models (e. g. abstraction level, technologies, estimation tools) as independent activities of individual projects.
4. **The overarching goal is**: Infrastructure development through
   a) Building and integrating power models, dark-silicon models.
   b) Full system simulation with dark silicon of multi-tile invasive architectures.
   c) Integration of different dark-silicon works, e. g. Projects B2, B3, B4, B5, C1.
   d) Prototyping dark-silicon effects through emulation and demonstration and how application projects can incorporate dark-silicon and energy efficiency effects.

A couple of examples of joint collaborative research activities from the last year are listed below.

1. The research work of Project B3 explored the impact of dark-silicon management for performance optimisation, dark-silicon management for ageing optimisation, and dark-silicon management for energy optimisation (for more details and a list of publications, we refer to Project B3).
2. **Open-Source Software**: Two tools implementing some work of B3, specifically, the Thermal Safe Power (TSP) power budgeting technique and the MatEx transient/peak temperature computation framework, are available at http://ces.itec.kit.edu/download.
3. Some collaborative results of the InvasIC-wide dark-silicon patterning are summarised below.
   a) **Collaborative activities between Project B3 and Project B2**: Integrating other fabrics like TCPA and *i*-Core and exploring the dark-silicon patterning impact. Exploring the temperature and power density variations for TCPA, RISC-like cores, and *i*-Core. Requirements: ASIC synthesis output, layout / floorplan, power estimates or power model, performance traces, same technology, etc. Integrated system simulations. Derive Thermal Safe Power (TSP) for heterogeneous fabrics [Khd+16].
   b) **Collaborative activities between Project B3 and Project C1**: Coordinated resource and dark-silicon management: Interfacing with Project C1 to forward the dark-silicon constraints to the agent-layer? Studying the impact of dark-silicon management on the efficiency *i*RTSS. Analysing potential conflicts between the dark-silicon management and *i*RTSS.
   c) **Collaborative activities between Project B3 and Project B5**: Integrated *i*NoC and tiles patterning for dark silicon. Interesting problems to answer are: Darkening the routers or dark tiles or not? Multi-layer vs. multiple V-f levels for *i*NoC? *i*NoC and multiple voltage islands? Single layer: Re-routing in case of dark routers within the active layer.
   d) **Collaborative activities between Projects B1, B2, B3, B4, and C1**: Integrated and coordinated a cross-layer monitoring and optimisation approach for distributed dark-silicon management for heterogeneous multi-tile architectures [Pag+16].

Further WG Activities: The WG team has also performed several activities for dissemination at a wider and international level through seminars and embedded tutorial organisations. A summary of these activities is given below.

1. *Embedded Tutorial on "The Dark Silicon Problem: Technology to the Rescue?" at the IEEE/ACM 19th Design, Automation and Test in Europe Conference (DATE), 2016*: In this embedded tutorial, we consider how researchers are leveraging new technologies (especially, 3D integration and new transistor technologies) in order to address the dark-silicon problem. More details can be found at https://www.date-conference.com/date16/conference/session/2.2.
2. *Dagstuhl Seminar on "Dark Silicon: From Embedded to HPC Systems", 2016*: This seminar involved presentations on the state-of-the-art in power and energy management in HPC and on techniques mitigating the dark-silicon problem in embedded systems. In a joint session, commonalities and differences as well as collaboration potential in the area of dark silicon were explored. More details can be found at http://www.dagstuhl.de/16052.

Publications

[Khd+16] H. Khdr, S. Pagani, Éricles R. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores". In: *IEEE Transactions on Computers (TC)* (July 2016). DOI: 10.1109/TC.2016.2595560.

[Pag+16] S. Pagani, L. Bauer, Q. Chen, E. Glocker, F. Hannig, A. Herkersdorf, H. Khdr, A. Pathania, U. Schlichtmann, D. Schmitt-Landsiedel, M. Sagi, Éricles Sousa, P. Wagner, V. Wenzel, T. Wild, and J. Henkel. "Dark Silicon Management: An Integrated and Coordinated Cross-Layer Approach". In: *it – Information Technology* 58.6 (Sept. 16, 2016), pp. 297–307. DOI: 10.1515/itit-2016-0028.

III

Events and Activities

Summary

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

Figure 5.2: From left to right: Ina Derr (Financial Services), Dr.-Ing. Jürgen Kleinöder (Managing Director), Dr. Katja Lohmann (Deputy Managing Director), Prof. Jürgen Teich (Coordinator and PI) and Dr. Sandra Mattauch (Public Relations)

In the following sections, we are again proud to summarise major events and a whole bunch of activities in 2016. These include Internal Meetings (Section 6), Training Courses and Tutorials (Section 7) as well as further InvasIC Activities (Section 8). Last but not least, we present the current composition of the Industrial and Scientific Board in Section 9.

Figure 5.3: At the annual meeting in Blaubeuren, September 2016

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 – InvasIC. In 2016, researchers met at the following opportunities (list not being exhaustive):

Event	Date	
Semi-annual Meeting 2016	Feb. 15/16, 2016, Adelsried	At the semi-annual meeting, all projects and working groups presented their progress in short talks. Additionally, each project introduced the ideas for the demonstration platform on a plenary session.
Doctoral Researcher Retreat	Mar. 7–9, 2016, Aalen	This year, the doctoral researchers met in Aalen to discuss progress and further challenges of the second funding phase.
WG Memory Hierarchy	May 3, 2016, Munich	Members of the working group met in Munich to (i) continue discussion on inter-tile cache coherency and (ii) start discussions on MMU for the InvasIC architecture.
WG Demonstrator	June 1, 2016, Munich	Members from the working group met in Munich to define a small set of demo scenarios for the demonstrator.
Doctoral Researcher Retreat	Sept. 12–14, 2016, Blaubeuren	The 9th InvasIC DRR took place at the Tagungszentrum Blaubeuren in conjunction with the annual meeting.
Annual Meeting 2016	Sept. 15/16, 2016, Blaubeuren	The main focus of the first day was on reviewing and discussing the progress achieved in the fields of predictability and demonstrations. On the second day of the annual meeting, the InvasIC Industrial and Scientific Board attended to evaluate the ideas and progress of the presented projects and demonstrations.
Actor Meeting I	Nov. 3, 2016, Munich	At the meeting, the integration of the actor concept and actor graphs in the *i*RTSS agent system were discussed.
Actor Meeting II	Dec. 8, 2016, Munich	Main topic of the meeting was a continuation of the discussion on integrating the concepts of application characterisation in the *i*RTSS agent system.
A1/C3 Meeting	Dec. 12, 2016, Karlsruhe	The researchers from Project A1 and Project C3 met to discuss ongoing work as well as ideas for the third funding phase.

7 Training Courses and Tutorials

The following internal workshops and training courses were organised under the coordination of Project Z, to give InvasIC members the opportunity to strengthen their soft skills, train their key qualifications, and improve their knowledge on topics related to invasive computing.

Event	Date	
Workshop Rhetoric	July 28/29, 2016, Karlsruhe	Members of InvasIC were invited to join a two-day workshop on “Rhetoric” held by the professional didactics trainer Barbara Berndt.
Workshop Small Talk and Networking	Oct. 6/7, 2016, Munich	The seminar gave young researchers the opportunity to improve their language skills in the areas of research and academia and to learn successful networking.
Workshop Time- and Self-Management	Oct. 27/28, 2016, Erlangen	The workshop was organised with the objective of becoming familiar with tools and techniques to master and control work load instead of being controlled by it.

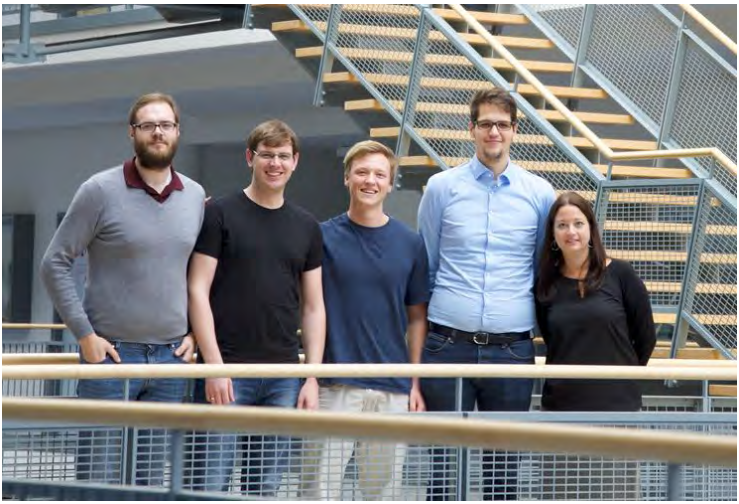


Figure 7.1: Small Talk and Networking, October 2016 in Munich

Workshops

Based on multiple requests and wishes of our doctoral researchers, InvasIC organised workshops at all three sites of the Transregio on topics identified at the annual meeting in Adelsried.
The first Workshop on “Rhetoric” was held by Barbara Berndt from i-communicate. The participants had a look at different ways of articulation, body language signs and the common rules of presentation. Movements on the stage, voice variations and gesticulations were practised and improved. Theoretical inputs were combined with exercises, and a video analysis of short talks given by the doctoral researchers completed the soft skill seminar.

Figure 7.2: Workshop on Rhetoric, Karlsruhe, July 2016

For many scientists, conferences are not only the place to gain knowledge but mainly to get in touch with the people behind. At certain occasions, like coffee breaks, they start talking to each other and even manage to stay in contact over a longer period of time. It needs networking-management to make “what goes around, comes around” happen. But how to start a conversation? Which are good topics, which ones should be avoided? And when is the best moment to bid farewell? All these questions were answered during the workshop “Small Talk and Networking” which took place in October in Munich.
In addition, Project Z organised a workshop on “Time- and Self-Management” in Erlangen, where the doctoral researchers analysed time structure and working habits. With different models and methods the atten-

dees get ideas of how to prioritise tasks, structure the day and work with goals to stay motivated. In addition, they discussed different ways of successful work-life-balance and found individual solutions for common time-management problems. In all three workshops, the participants were sensitised to apply gender mainstreaming principles in daily work.

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". Additionally, PIs of InvasIC gave talks and seminars at important research sites and conferences ("Invited Talks and Seminars") 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. Videos of the respective talks are provided at our website `http://www.invasic.de`.

Figure 8.1: Chancellor's Prof. Jason Cong together with Prof. Jürgen Teich and Dr.-Ing. Frank Hannig

Award for Bachelor Thesis of Theresa Pollinger

Theresa Pollinger, bachelor graduate of Prof. Jürgen Teich, received an award of Brose Company on the occasion of the graduation ceremony of Faculty of Engineering, FAU, for her bachelor thesis titled "Video-based Object Tracking with a Pan-Tilt-Zoom Camera using the Parallel Programming Language X10".

Prof. Dr. Zoran Salcic visited the TCRC

In recognition of his academic achievements and his contribution to academic cooperation with German specialist colleagues, Prof. Zoran Salcic (University Auckland, New Zealand) received a Research Award by the Alexander von Humboldt Foundation to continue his research with the Friedrich-Alexander-Universität Erlangen-Nürnberg for three months. This invitation, which has been initiated by Prof. Teich, has offered an opportunity to continue the collaboration with specialist colleagues in Germany that was brought about by the Research Award of the Alexander von Humboldt Foundation in 2010. The successful cooperation was re-intensified in 2016, when Prof. Salcic returned to the Chair of Hardware/Software Co-Design for another three months for researching new ideas.

Figure 8.2: Prof. Zoran Salcic

InvasIC Seminar

Place and Date	Title	Speaker
Erlangen, Jan. 29, 2016	Elastic Computing – Towards a New Paradigm for Distributed Systems	Prof. Schahram Dustdar (TU Wien)
Munich, Feb. 24, 2016	Hyperion	Martin Vorbach (PACT XPP Technologies)
Erlangen, Mar. 4, 2016	Bytespresso, toward embedded domain-specific languages for super-computing	Prof. Shigeru Chiba (University of Tokyo)
Munich, Apr. 15, 2016	Revisiting the Perfect Chip Paradigm: Cross-Layer Approaches to Design-ing and Monitoring Reliable Systems using Unreliable Components	Prof. Fadi Kurdahi (University of California)
Erlangen, May 6, 2016	Control-theoretic approaches to Energy Minimization under Soft Real-Time Constraints	Prof. Martina Maggio (Lund University)

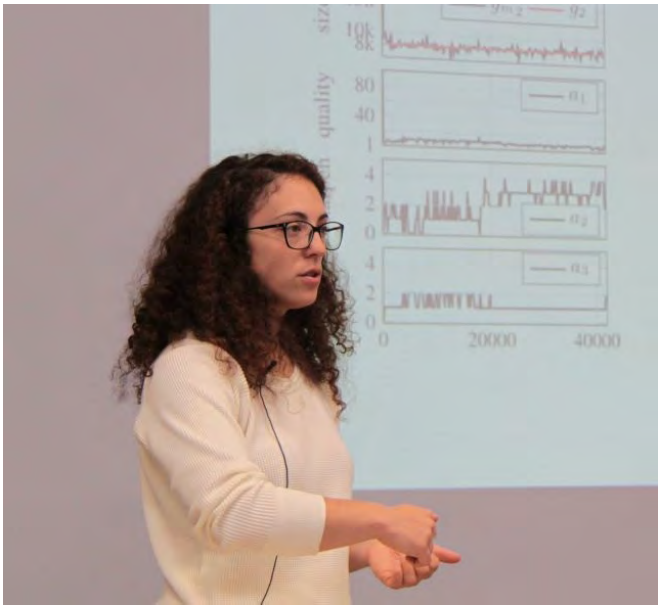


Figure 8.3: Professor Martina Maggio giving a talk at the InvasIC Seminar

Place and Date	Title	Speaker
Erlangen, Aug. 5, 2016	Bridging the gap between embedded systems and automation systems	Prof. Partha S. Roop (University of Auckland)
Erlangen, Aug. 8, 2016	Adaptive Parallel and Distributed Software Systems	Dr. Pramod Bhatotia (Technische Universität Dresden)
Erlangen, Sept. 19, 2016	High-Level Synthesis and Beyond	Prof. Jason Cong (University of California)
Erlangen, Sept. 30, 2016	From tamed heterogeneous cores to system wide intrusion tolerance	Dr. Marcus Völp (University of Luxembourg)
Erlangen, Oct. 19, 2016	Security Enhanced Multi-Processor System Architecture for Mixed-Critical Embedded Applications	Dr. Morteza Biglari-Abhari (University of Auckland)
Erlangen, Nov. 25, 2016	Resource Allocation under Uncertainty – Online Scheduling with Hard Deadlines	Prof. Dr. Nicole Megow (University of Bremen)

Figure 8.4: Prof. Partha S. Roop at the InvasIC Seminar

Invited Talks

Place and Date	Title	Speaker
Prague, Czech Republic, Jan. 18, 2016 Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO 2016)	Keynote: The Role of Restriction and Isolation for Increasing the Predictability of MPSoC Stream Processing	Prof. J. Teich (FAU)
Prague, Czech Republic, Jan. 19, 2016 International Workshop on Polyhedral Compilation Techniques (IMPACT 2016)	Keynote: Symbolic Loop Parallelization for Adaptive Multi-Core Systems – Recent Advances and Benefits	Prof. J. Teich (FAU)
Saarbrücken, Germany, Feb. 26, 2016 University Saarbrücken	Talk: RaPping and Compilation for Highly Dynamic Parallelism	Prof. G. Snelting (KIT)
Dresden, Germany, Mar. 18, 2016 Friday Workshop on Resource Awareness and Application Autotuning in Adaptive and Heterogeneous Computing (DATE 2016)	Talk: Restriction and Isolation for Predictable MPSoC Stream Processing	Prof. J. Teich (FAU)
Austin, USA, June 6, 2016 University of Texas at Austin	Talk: Predictable MPSoC Stream Processing Using Invasive Computing	Prof. J. Teich (FAU)
Lübeck, Germany, July 29, 2016 University of Lübeck	Talk: Predictability, Fault Tolerance, and Security on Demand using Invasive Computing	Prof. J. Teich (FAU)

Figure 8.5: Prof. Jürgen Teich giving a keynote talk at IMPACT 2016, Prague

Organised Conferences and Workshops

Place and Date	Title	Organiser
Dagstuhl, Germany Jan. 31–Feb. 3, 2016	Dagstuhl Seminar 16052: Dark Silicon: From Embedded to HPC Systems	Prof. M. Gerndt (TUM) Prof. M. Glaß (FAU) Prof. S. Parameswaran (UNSW) Dr. B. L. Rountree (LLNL)
Dresden, Germany, Mar. 14–18, 2016	Design, Automation and Test in Europe (DATE 2016)	Prof. J. Teich (FAU) Programme Chair
Dresden, Germany, Mar. 18, 2016 Design, Automation and Test in Europe (DATE 2016)	First DATE Friday Workshop on Resource Awareness and Application Autotuning in Adaptive and Heterogeneous Computing	Prof. W. Stechele (TUM) Prof. C. Silvano (Politecnico di Milano) Prof. S. Wong (TU Delft)
Nuremberg, Germany, Apr. 4–7, 2016	29th GI/ITG International Conference on Architecture of Computing Systems (ARCS)	Prof. D. Fey (FAU) General Chair Prof. J. Teich and Prof. W. Schröder-Preikschat (FAU) General Co-Chair Dr.-Ing. F. Hannig (FAU) Programme Chair
Nuremberg, Germany, Apr. 4/5, 2016 International Conference on Architecture of Computing Systems (ARCS 2016)	International Workshop on Multi-Objective Many-Core Design (MOMAC)	Dr.-Ing. S. Wildermann (FAU) Prof. M. Glaß (FAU)
Munich, Germany, June 30/July 1, 2016	The Munich Workshop on Design Technology Coupling	Dr. H. Graeb (TUM) Dr. S. Nassif (Radyalis)
Dagstuhl, Germany Oct. 30–Nov. 4, 2016	Dagstuhl Seminar 16441: Adaptive Isolation for Predictability and Security	Prof. J. Teich (FAU) Prof. I. Verbauwhede (KU Leuven) Prof. L. Thiele (ETH Zürich) Prof. T. Mitra (NUS)
Hangzhou, China, Nov. 7–9, 2016 International Green and Sustainable Computing Conference (IGSC)	Third Workshop on Low-Power Dependable Computing (LPDC)	Dr. M. Shafique (KIT) Prof. Dr. X. Zhu (NUDT) Dakai Zhu (UTSA)

Design, Automation and Test in Europe

Professor Jürgen Teich served as the Program Chair of DATE 2016 which was held at the International Congress Centre Dresden, Germany, from March 14 to 18, 2016. For the 19th successive year, DATE has prepared an exciting technical programme, says Jürgen Teich, Programme Chair of DATE 2016. With the help of 327 members of the Technical Program Committee, who carried out more than 3000 reviews (about four per submission), finally 199 papers (24%) were selected for regular presentation and 81 additional ones (10%) for interactive presentation.

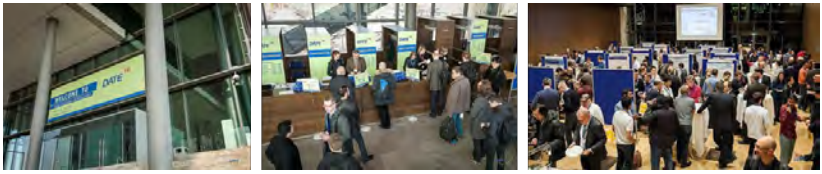


Figure 8.6: Impressions from DATE 2016

DATE 2016 received 829 paper submissions, a large share (42%) coming from authors in Europe, 29% of submissions from Asia, 25% from North America, and 4% from the rest of the world. This clearly demonstrates DATE's international character, its global reach and impact.
DATE was attended by 1400 delegates from 50 countries worldwide.

Figure 8.7: Impressions from DATE 2016

The conference started on Monday, with 10 in-depth technical tutorials offered from experts of the industrial and academic worlds on innovative as well as fundamental topics related to design solutions, power efficiency, the internet of things, secure systems and testing and diagnosis. On the same day, the popular PhD Forum, hosted by EDAA, ACM SIGDA, and IEEE CEDA, gave the opportunity to the 33 selected PhDs to present their work to a broad audience in the system design and design automation community from both industry and academia.

29th GI/ITG International Conference on Architecture of Computing Systems

The 29th International Conference on Architecture of Computing Systems (ARCS 2016) was hosted by the Department of Computer Science at Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany, during April 4 to 7, 2016. ARCS 2016 took place in Nuremberg at FAU's Faculty of Business, Economics, and Law in Nuremberg and attracted 100 participants. The conference continued the long-standing ARCS tradition of reporting top-notch results in computer architecture and other related areas. ARCS was founded in 1970 by the German computer pioneer Prof. Wolfgang Händler, who also founded the Computer Science Department at FAU in 1966.

Figure 8.8: Impressions from ARCS 2016

ARCS was organised by multiple PIs of InvasIC, namely Professor Wolfgang Schröder-Preikschat and Professor Jürgen Teich as General Co-Chairs and Frank Hannig as Program Chair. The strong technical program was complemented by three keynote talks on: "Knights Landing Intel Xeon Phi CPU: Path to Parallelism with General Purpose Programming" by Avinash Sodani, Chief Architect 'Knights Landing' Xeon-Phi processor at Intel Corporation; "Massive Parallelism – C++ and OpenMP Parallel Programming Models of Today and Tomorrow" by Michael Wong, CEO of OpenMP Corporation; and "Heterogeneous Systems Era" by John Glossner, President of the Heterogeneous System

Architecture Foundation (HSAF) and CEO Optimum Semiconductor Technologies; as well as five workshops and a tutorial.

Figure 8.9: Frank Hannig, Avinash Sodani, and Michael Wong giving talks at ARCS 2016

Beside the technical program, the combined visit of the special exhibition "From Abacus to Exascale – Vom Abakus zu Exascale", the conference dinner and best paper award ceremony in the Museum of Industrial Culture in Nuremberg on Wednesday evening was another highlight.

Figure 8.10: Impressions from ARCS 2016

Dagstuhl Seminar 16441 "Adaptive Isolation for Predictability and Security"

Prof. Dr.-Ing. Jürgen Teich (Hardware/Software Co-Design, FAU), Prof. Dr. Ir. Ingrid Verbauwhede (KU Leuven, BE), Prof. Dr.-Ing. Lothar Thiele (ETH Zürich, CH) and Prof. Dr. Tulika Mitra (National University of Singapore, SG) organised and coordinated the Dagstuhl Seminar on "Adaptive Isolation for Predictability and Security" that took place from 30.10.-04.11.2016 at Dagstuhl Castle.

Adaptive Isolation, the topic of the proposed Dagstuhl Seminar, may be seen as a novel and important research topic for providing predictability

Figure 8.11: Participants of the Dagstuhl Seminar 16441

of not only timing but also security and maybe even other properties of execution on a multi-core platform on a per application basis while easing and trading off compile-time and run-time complexity. First, a common understanding of which techniques may be used for isolation including hardware design, resource reservation protocols, virtualisation, and including novel hybrid and dynamic resource assignment techniques were identified. The topic and the findings in several breakout sessions and by individual talks given are completely in line with the topic of Invasive Computing that creates isolation of applications on multi-core resources by default.

Dagstuhl Seminar 16052 "Dark Silicon: From Embedded to HPC Systems"

Prof. Dr. Gerndt (TUM), Sri Parameswaran (UNSW, Sydney, Australia), Barry L. Rountree (LLNL, Livermore, United States) and Prof. Dr.-Ing. Glaß (FAU) organised and coordinated the Dagstuhl Seminar 16052 on "Dark Silicon: From Embedded to HPC Systems". The goal of this Dagstuhl Seminar was to bring together experts from the different domains and to discuss the state-of-the-art and identify future collaboration topics based on common research interests. Three major topics were discussed: Dark Silicon, Power and Energy Usage in HPC, and Hybrid Approaches to Resource Management with longer overview pre-

Figure 8.12: Dagstuhl Seminar 16052

sentations by invited speakers interspersed with research presentations by the attendees. Each part closed with a discussion slot. After these three parts, group discussions helped to identify future collaborative research directions.

International Workshop on Multi-Objective Many-Core Design

Michael Glaß and Stefan Wildermann (FAU) organised the third International Workshop on Multi-Objective Many-Core Design (MOMAC) at ARCS 2016 at FAU's Faculty of Business, Economics, and Law in Nuremberg. Dr. Felix Reimann (Audi Electronics Venture GmbH, Gaimersheim, Deutschland) gave a keynote talk on "Towards A Holistic Design Space Exploration for Automotive E/E Architectures".

The Munich Workshop on Design Technology Coupling

Dr. Helmut Graeb (TUM) and Dr. Sani Nassif (Radyalis) organised in cooperation with the CRC/Transregio Invasive Computing and the SPP 1500 "Dependable Embedded Systems" the "Munich Workshop on Design Technology Coupling (DTC)". The workshop involved different contributions from industry, e. g. Infineon AG, Bosch GmbH, and Volkswagen AG, as well as from the two DFG-funded research programs. In one of the sessions, Prof. Teich gave a short introduction and overview of the topics and benefits of invasive multi-core computing for achieving timing predictability, fault tolerance and security for individual applica-

Figure 8.13: Dr.-Ing. Felix Reimann giving the keynote at MOMAC in Nuremberg

tion programs. This talk was followed by the overview talks “Providing Fault Tolerance Through Invasive Computing” by Dr. Vahid Lari (FAU) and “On-Chip Diagnosis of Multicore Platforms for Power Management” by Mark Sagi (TUM). On the second day of the meeting, the following demonstrations from the CRC/Transregio 89 were presented: An invasive object tracking application that was simulated and visualised in real-time using the simulator InvadeSIM from Project C2, and the code generation for Safe(r) loop computations using the compilation flows developed by Project C3.

Figure 8.14: Panel discussion at the Munich Workshop on Design Technology Coupling

9 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, Intel Director of Visual Computing Institute Saarbrücken)

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)

Georg-Simon-Ohm Hochschule Nürnberg

Prof. Dr. Christoph von Praun
(Faculty Member and Associate Department Chair)

IAV – Automotive Engineering

Elmar Maas (IAV, Gifhorn)

Xilinx

Michaela Blott (Xilinx, Dublin)

Meeting with the Industrial and Scientific Board

The members of the InvasIC Industrial and Scientific Board are periodically informed about progress and news of the CRC/Transregio InvasIC. In September 2016, the members of the Board met during the annual meeting in Blaubeuren. As an introduction, Prof. Jürgen Teich (Coordinator) gave an overview about all projects involved into the CRC/Transregio.

Figure 9.1: Poster session at the annual meeting in Blaubeuren

The following poster and demonstration session brought a good opportunity for the Board's members to get an idea about the current state of research in InvasIC. In the concluding plenary session, the opinions and suggestions of the Board's members were collected and discussed.

Figure 9.2: Members of the Industrial and Scientific Board during the plenary session at the annual meeting in Blaubeuren, September 2016. From left to right: Michaela Blott (Xilinx), Peter Hans Roth (IBM), Klaus-Dieter Schubert (IBM), Urs Gleim (Siemens), Dr. Patricia Sagmeister (IBM Rüschlikon), Elmar Maas (IAV), Hans-Christian Hoppe (Intel)

10 Publications

[Bak+16] A. Bakhtiari, D. Malhotra, A. Raoofy, M. Mehl, H.-J. Bungartz, and G. Biros. “A Parallel Arbitrary-Order Accurate AMR Algorithm for the Scalar Advection-Diffusion Equation”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. Salt Lake City, UT, USA: IEEE, 2016.

[Bha+16] V. Bhadouria, A. Tanase, M. Schmid, F. Hannig, J. Teich, and D. Ghoshal. “A Novel Image Impulse Noise Removal Algorithm Optimized for Hardware Accelerators”. In: *Journal of Signal Processing Systems* (Nov. 1, 2016). DOI: 10.1007/s11265-016-1187-5.

[Bre+16] J. Breitner, J. Graf, M. Hecker, M. Mohr, and G. Snelting. “On Improvements Of Low-Deterministic Security”. In: *Principles of Security and Trust (POST)*. Ed. by F. Piessens and L. Viganò. Vol. 9635. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2016, pp. 68–88. DOI: 10.1007/978-3-662-49635-0_4.

[BLU16] S. Buchwald, D. Lohner, and S. Ullrich. “Verified Construction of Static Single Assignment Form”. In: *25th International Conference on Compiler Construction*. Ed. by M. Hermenegildo. CC 2016. ACM, 2016, pp. 67–76. DOI: 10.1145/2892208.2892211.

[Cle+16] R. de Clercq, R. de Keulenaer, B. Coppens, B. Yang, P. Maene, K. de Bosschere, B. Preneel, B. de Sutter, and I. Verbauwhede. “SOFIA: Software and Control Flow Integrity Architecture”. In: *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. Dresden, Germany: IEEE, 2016, pp. 1172–1177.

[CMHGB16] I. Comprés, A. Mo-Hellenbrand, M. Gerndt, and H.-J. Bungartz. “Infrastructure and API Extensions for Elastic Execution of MPI Applications”. In: *Proceedings of the 23rd European MPI Users’ Group Meeting*. EuroMPI 2016. ACM, 2016, pp. 82–97. DOI: 10.1145/2966884.2966917.

[DBH17] M. Damschen, L. Bauer, and J. Henkel. “Timing Analysis of Tasks on Runtime Reconfigurable Processors”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.1 (Jan. 2017), pp. 294–307. DOI: 10.1109/TVLSI.2016.2572304.

[DSP16] G. Drescher and W. Schröder-Preikschat. “Adaptive Memory Protection for Many-Core Systems”. In: *Adaptive Isolation for Predictability and Security (Dagstuhl Seminar 16441)*. Vol. 6. 2016.

[Dre+16] G. Drescher, C. Erhardt, F. Freiling, J. Götzfried, D. Lohmann, P. Maene, T. Müller, I. Verbauwhede, A. Weichslgartner, and S. Wildermann. “Providing Security on Demand Using Invasive Computing”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 281–295. DOI: 10.1515/itit-2016-0032.

[EHSP16] C. Eibel, T. Hönig, and W. Schröder-Preikschat. “Energy Claims at Scale: Decreasing the Energy Demand of HPC Workloads at OS Level”. In: *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. May 2016, pp. 1114–1117. DOI: 10.1109/IPDPSW.2016.69.

[FLB16] S. Friederich, N. Lehmann, and J. Becker. “Adaptive Bandwidth Router for 3D Network-on-Chips”. In: *Applied Reconfigurable Computing*. 2016, pp. 352–360. DOI: 10.1007/978-3-319-30481-6_30.

[FNB16] S. Friederich, M. Neber, and J. Becker. “Power Management Controller for Online Power Saving in Network-on-Chips”. In: *International Symposium on Embedded Multicore/Manycore SoCs (MCSoC)*. Vol. 10. Sept. 2016.

[GGPR16] H. M. Gerndt, M. Glaß, S. Parameswaran, and B. L. Rountree. “Dark Silicon: From Embedded to HPC Systems (Dagstuhl Seminar 16052)”. In: *Dagstuhl Reports* 6.1 (2016). Ed. by H. M. Gerndt, M. Glaß, S. Parameswaran, and B. L. Rountree, pp. 224–244. DOI: 10.4230/DagRep.6.1.224.

[GDPM16] J. Götzfried, N. Dörr, R. Palutke, and T. Müller. “HyperCrypt: Hypervisor-based Encryption of Kernel and User Space”. In: *11th International Conference on Availability, Reliability and Security (ARES’16)*. Ed. by S. Research. Salzburg, Austria: IEEE, 2016. URL: https://www1.cs.fau.de/hypercrypt.

[Göt+16] J. Götzfried, T. Müller, G. Drescher, S. Nürnberger, and M. Backes. “RamCrypt: Kernel-based Address Space Encryption for User-mode Processes”. In: *11th ACM Asia Conference on Computer and Communications Security (ASAICCS)*. (Xi’an, Shaanxi, China). Special Interest Group on Security, Audit and Control (SIGSAC). ACM, 2016. DOI: 10.1145/2897845.2897924. URL: https://www1.cs.fau.de/ramcrypt.

[GHMS16] J. Graf, M. Hecker, M. Mohr, and G. Snelting. “Tool Demonstration: JOANA”. In: *Principles of Security and Trust (POST)*. Ed. by F. Piessens and L. Viganò. Vol. 9635. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2016, pp. 89–93. DOI: 10.1007/978-3-662-49635-0_5.

[GBH17] A. Grudnitsky, L. Bauer, and J. Henkel. “Efficient Partial Online Synthesis of Special Instructions for Reconfigurable Processors”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.2 (Feb. 2017), pp. 594–607. DOI: 10.1109/TVLSI.2016.2585603.

[HHSP16] T. Hönig, B. Herzog, and W. Schröder-Preikschat. “The Narrow Way: Constructive Measures at Operating-System Level for Low Energy Use”. In: *Proceedings of the 30th Environmental Informatics Conference (EnviroInfo 2016)*. 2016, pp. 329–335.

[Hei+16] J. Heisswolf, S. Friederich, L. Masing, A. Weichslgartner, A. M. Zaib, C. Stein, M. Duden, J. Teich, T. Wild, A. Herkersdorf, and J. Becker. “A Novel NoC-Architecture for Fault Tolerance and Power Saving”. In: *Proceedings of the third International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: IEEE, Apr. 4, 2016, 8 pp.

[Hen+16] J. Henkel, S. Pagani, H. Khdr, F. Kriebel, S. Rehman, and M. Shafique. “Towards Performance and Reliability-Efficient Computing in the Dark Silicon Era”. In: *Proceedings of the 19th Design, Automation and Test in Europe (DATE)*. Dresden, Germany, Mar. 14–18, 2016.

[Khd+16] H. Khdr, S. Pagani, Éricles R. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. “Power Density-Aware Resource Management for Heterogeneous Tiled Multicores”. In: *IEEE Transactions on Computers (TC)* (July 2016). DOI: 10.1109/TC.2016.2595560.

[Krö16] M. Kröhnert. “A Contribution to Resource-Aware Architectures for Humanoid Robots”. Dissertation. High Performance Humanoid Technologies (H2T), KIT-Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany, July 22, 2016.

[KGVA16] M. Kröhnert, R. Grimm, N. Vahrenkamp, and T. Asfour. “Resource-Aware Motion Planning”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. (Stockholm, Sweden). May 2016, pp. 32–39. DOI: 10.1109/ICRA.2016.7487114.

[Lar15] V. Lari. “Invasive Tightly Coupled Processor Arrays”. Dissertation. Hardware/Software Co-Design, Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Nov. 18, 2015.

[Lar16a] V. Lari. *Invasive Tightly Coupled Processor Arrays*. Springer Singapore, 2016. DOI: 10.1007/978-981-10-1058-3.

[Lar16b] V. Lari. “Providing Fault Tolerance Through Invasive Computing”. Talk at DTC 2016, The Munich Workshop on Design Technology Coupling, Munich, Germany. June 30, 2016.

[Lar+15] V. Lari, J. Teich, A. Tanase, M. Witterauf, F. Khosravi, and B. Meyer. “Techniques for On-Demand Structural Redundancy for Massively Parallel Processor Arrays”. In: *Journal of Systems Architecture (JSA)* 61.10 (Nov. 2015), pp. 615–627. DOI: 10.1016/j.sysarc.2015.10.004.

[Lar+16] V. Lari, A. Weichslgartner, A. Tanase, M. Witterauf, F. Khosravi, J. Teich, J. Becker, J. Heißwolf, and S. Friederich. “Providing Fault Tolerance Through Invasive Computing”. In: *it – Information Technology* 58.6 (Oct. 19, 2016), pp. 309–328. DOI: 10.1515/itit-2016-0022.

[LGG16] W. Liu, M. Gerndt, and B. Gong. “Model-based MPI-IO tuning with Periscope tuning framework”. In: *Concurrency and Computation: Practice and Experience* 28.1 (2016), pp. 3–20. DOI: 10.1002/cpe.3603.

[MRB16] O. Meister, K. Rahnema, and M. Bader. “Parallel Memory-Efficient Adaptive Mesh Refinement on Structured Triangular Meshes with Billions of Grid Cells”. In: *ACM Transactions on Mathematical Software* 43.3 (2016), 19:1–19:27. DOI: 10.1145/2947668.

[MT17] M. Mohr and C. Tradowsky. “Pegasus: Efficient Data Transfers for PGAS Languages on Non-Cache-Coherent Many-Cores”. In: *Design, Automation Test in Europe Conference Exhibition (DATE)*. 2017. forthcoming.

[Pag16] S. Pagani. “Power, Energy, and Thermal Management for Clustered Manycores”. Dissertation. Chair for Embedded Systems (CES), Department of Computer Science, Karlsruhe Institute of Technology (KIT), Germany, Nov. 24, 2016.

[Pag+16] S. Pagani, L. Bauer, Q. Chen, E. Glocker, F. Hannig, A. Herkersdorf, H. Khdr, A. Pathania, U. Schlichtmann, D. Schmitt-Landsiedel, M. Sagi, Éricles Sousa, P. Wagner, V. Wenzel, T. Wild, and J. Henkel. “Dark Silicon Management: An Integrated and Coordinated Cross-Layer Approach”. In: *it – Information Technology* 58.6 (Sept. 16, 2016), pp. 297–307. DOI: 10.1515/itit-2016-0028.

[Pag+16a] S. Pagani, A. Pathania, M. Shafique, J.-J. Chen, and J. Henkel. “Energy Efficiency for Clustered Heterogeneous Multicores”. In: *IEEE Transactions on Parallel and Distributed Systems (TPDS)* (2016).

[Pag+16b] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel. "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon". In: *IEEE Transactions on Computers (TC)* (2016). Feature Paper of the Month. DOI: 10.1109/TC.2016.2564969.

[Pag+17] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel. "Thermal Safe Power: Efficient Thermal-Aware Power Budgeting for Manycore Systems in Dark Silicon". In: *The Dark Side of Silicon*. Ed. by A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen. Springer, 2017.

[Pat+16a] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. "Optimal Greedy Algorithm for Many-Core Scheduling". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* PP.99 (2016), pp. 1–1. DOI: 10.1109/TCAD.2016.2618880.

[Pat+17] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. "Defragmentation of Tasks in Many-Core Architectures". In: *ACM Transactions on Architecture and Code Optimization* (2017).

[Pat+16b] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. "Distributed Fair Scheduling for Many-Cores". In: *Design Automation and Test in Europe (DATE)*. Dresden, Germany, 2016.

[Pat+16c] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel. "Distributed scheduling for many-cores using cooperative game theory". In: *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. ACM. Austin, TX, USA, 2016, pp. 133–139.

[PB16] A. Pöppl and M. Bader. "SWE-X10: An Actor-based and Locally Coordinated Solver for the Shallow Water Equations". In: *Proceedings of the 6th ACM SIGPLAN Workshop on X10*. (Santa Barbara, CA, USA). X10 2016. ACM, 2016, pp. 30–31. DOI: 10.1145/2931028.2931034.

[PBSG16] A. Pöppl, M. Bader, T. Schwarzer, and M. Glaß. "SWE-X10: Simulating shallow water waves with lazy activation of patches using ActorX10". In: *Proceedings of the 2nd International Workshop on Extreme Scale Programming Models and Middleware (ESPM2)*. IEEE, Nov. 2016, pp. 32–39. URL: http://conferences.computer.org/espm2/2016/papers/3858a032.pdf.

[PGT17] B. Pourmohseni, M. Glaß, and J. Teich. “Automatic Operating Point Distillation for Hybrid Mapping Methodologies”. In: *Proceedings of Design, Automation and Test in Europe Conference Exhibition (DATE)*. Lausanne, Switzerland: IEEE, 2017. forthcoming.

[RGM16] L. Richter, J. Götzfried, and T. Müller. “Isolating Operating System Components with Intel SGX”. In: *1st Workshop on System Software for Trusted Execution (SysTEX'16)*. Trento, Italy: ACM, 2016. DOI: 10.1145/3007788.3007796. URL: https://www1.cs.fau.de/sgx-kernel.

[RHT16] S. Roloff, F. Hannig, and J. Teich. “InvadeSIM: A Simulator for Heterogeneous Multi-Processor Systems-on-Chip”. Tool Presentation at the University Booth at Design, Automation and Test in Europe (DATE), Dresden, Germany. Mar. 14–18, 2016. URL: https://www.date-conference.com/system/files/file/date16/ubooth/37912.pdf.

[Rol+16] S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. “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.

[RDGL16] V. Rothberg, C. Dietrich, A. Graf, and D. Lohmann. “Function Multiverses for Dynamic Variability”. In: *Foundations and Applications of Self* Systems*. Ed. by J. Andersson, R. Capilla, and H. Eichelberger. Augsburg, 2016, pp. 1–5. URL: https://www4.cs.fau.de/Publications/2016/rothberg_16_dspl.pdf.

[SH16] M. Sagi and A. Herkersdorf. “On-Chip Diagnosis of Multicore Platforms for Power Management”. Workshop Presentation, DTC 2016, The Munich Workshop on Design Technology Coupling, Munich, Germany. June 30, 2016.

[SHJL16] U. Schlichtmann, M. Hashimoto, I. H.-R. Jiang, and B. Li. “Reliability, Adaptability and Flexibility in Timing: Buy a Life Insurance for Your Circuits”. In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE/ACM Press, Jan. 2016, pp. 705–711. DOI: 10.1109/ASPDAC.2016.7428094.

[Taj+16] H. Tajik, B. Donyanavard, N. Dutt, J. Jahn, and J. Henkel. “SPMPool: Runtime SPM Management for Memory-Intensive Applications in Embedded Many-Cores”. In: *ACM Trans. Embed. Comput. Syst.* 16.1 (Oct. 2016), 25:1–25:27. DOI: 10.1145/2968447.

[Tan+16] A. Tanase, M. Witterauf, Éricles R. Sousa, V. Lari, F. Hannig, and J. Teich. “LoopInvader: A Compiler for Tightly Coupled Processor Arrays”. Tool Presentation at the University Booth at Design, Automation and Test in Europe (DATE), Dresden, Germany. Mar. 14–18, 2016. URL: https://www.date-conference.com/system/files/file/date16/ubooth/37913.pdf.

[Tei16] J. Teich. “Invasive Computing – Editorial”. In: *it – Information Technology* 58.6 (Nov. 24, 2016), pp. 263–265. DOI: 10.1515/itit-2016-0043.

[Tei+16] J. Teich, M. Glaß, S. Roloff, W. Schröder-Preikschat, G. Snelting, A. Weichslgartner, and S. Wildermann. “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.

[Tom+16] A. Toma, S. Pagani, J.-J. Chen, W. Karl, and J. Henkel. “An Energy-Efficient Middleware for Computation Offloading in Real-Time Embedded Systems”. In: *Proceedings of the 22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Daegu, South Korea, Aug. 17–19, 2016.

[THMB16] C. Tradowsky, T. Harbaum, L. Masing, and J. Becker. “A Novel ADL-based Approach to Design Adaptive Application-Specific Processors”. In: *Best of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2016. forthcoming.

[Tra+16a] C. Tradowsky, E. Cordero, C. Orsinger, M. Vesper, and J. Becker. *A Dynamic Cache Architecture for Efficient Memory Resource Allocation in Many-Core Systems*. Cham: Springer International Publishing, 2016, pp. 343–351. DOI: 10.1007/978-3-319-30481-6_29.

[Tra+16b] C. Tradowsky, E. Cordero, C. Orsinger, M. Vesper, and J. Becker. *Adaptive Cache Structures*. Cham: Springer International Publishing, 2016, pp. 87–99. DOI: 10.1007/978-3-319-30695-7_7.

[Tur+16] F. Turan, R. de Clercq, P. Maene, O. Reparaz, and I. Verbauwhede. “Hardware Acceleration of a Software-based VPN”. In: *26th International Conference on Field Programmable Logic and Applications (FPL’16)*. Lausanne, Switzerland: IEEE, 2016, pp. 1–9. DOI: 10.1109/FPL.2016.7577321.

[Wäc+16] M. Wächter, S. Ottenhaus, M. Kröhnert, N. Vahrenkamp, and T. Asfour. “The ArmarX Statechart Concept: Graphical Programming of Robot Behaviour”. In: *Frontiers in Robotics and AI* 3.33 (2016). DOI: 10.3389/frobt.2016.00033.

[WWH16] P. Wagner, T. Wild, and A. Herkersdorf. “DiaSys: On-Chip Trace Analysis for Multi-processor System-on-Chip”. In: *Architecture of Computing Systems (ARCS'16)*. Nuremberg, Germany: Springer, 2016.

[WT16] A. Weichslgartner and J. Teich. “Position Paper: Towards Redundant Communication through Hybrid Application Mapping”. In: *Proceedings of the third International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*. Nuremberg, Germany: IEEE, Apr. 4, 2016, 4 pp.

[Wei+16] A. Weichslgartner, S. Wildermann, J. Götzfried, F. Freiling, M. Glaß, and J. Teich. “Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs”. In: *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES)*. (Sankt Goar, Germany). ACM, May 23, 2016, pp. 153–162. DOI: 10.1145/2906363.2906370.

[Wil+16] S. Wildermann, M. Bader, L. Bauer, M. Damschen, D. Gabriel, M. Gerndt, M. Glaß, J. Henkel, J. Paul, A. Pöppl, S. Roloff, T. Schwarzer, G. Snelting, W. Stechele, J. Teich, A. Weichslgartner, and A. Zwinkau. “Invasive Computing for Timing-Predictable Stream Processing on MPSoCs”. In: *it – Information Technology* 58.6 (Sept. 30, 2016), pp. 267–280. DOI: 10.1515/itit-2016-0021.

[WTHT16] M. Witterauf, A. Tanase, F. Hannig, and J. Teich. “Modulo Scheduling of Symbolically Tiled Loops for Tightly Coupled Processor Arrays”. In: *Proceedings of the 27th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. London, United Kingdom: IEEE, July 6–8, 2016, pp. 58–66. DOI: 10.1109/ASAP.2016.7760773.

[WGGM16] A. Würstlein, M. Gernoth, J. Götzfried, and T. Müller. “Exzess: Hardware-based RAM Encryption against Physical Memory Disclosure”. In: *Architecture of Computing Systems (ARCS'16)*. Nuremberg, Germany: Springer, 2016. DOI: 10.1007/978-3-319-30695-7_5. URL: https://www4.cs.fau.de/~arw/exzess.

[ZLS16] G. L. Zhang, B. Li, and U. Schlichtmann. “EffiTest: Efficient Delay Test and Statistical Prediction for Configuring Post-silicon Tunable Buffers”. In: *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. (Austin, TX, USA). ACM, 2016, 60:1–60:6. DOI: 10.1145/2897937.2898017.

[Zwi16] A. Zwinkau. “An X10 Memory Model”. In: *Proceedings of the sixth ACM SIGPLAN X10 Workshop*. X10 ’16. June 2016.