Design Methodologies for Enabling Self-awareness in Autonomous Systems

Armin Sadighi*, Bryan Donyanavard†, Thawra Kadeed‡, Kasra Moazzemi†, Tiago Mück†, Ahmed Nassar†, Amir M. Rahmani†§, Thomas Wild*, Nikil Dutt†, Rolf Ernst‡, Andreas Herkersdorf*, and Fadi Kurdahi†

*Chair for Integrated Systems, Technische Universität München, München, Germany
§Institute of Computer Technology, TU Wein, Vienna, Austria
†Center for Embedded and Cyber-physical Systems (CECS), University of California, Irvine, USA
‡Institute of Computer and Network Engineering, TU Braunschweig, Germany

***Abstract*—This paper deals with challenges and possible solutions for incorporating self-awareness principles in EDA design flows for autonomous systems. We present a holistic approach that enables self-awareness across the software/hardware stack, from systems-on-chip to systems-of-systems (autonomous car) contexts. We use the Information Processing Factory (IPF) metaphor as an exemplar to show how self-awareness can be achieved across multiple abstraction levels, and discuss new research challenges. The IPF approach represents a paradigm shift in platform design by envisioning the move towards a consequent platform-centric design in which the combination of self-organizing learning and formal reactive methods guarantee the applicability of such cyber-physical systems in safety-critical and high-availability applications.**

I. Introduction

In 2001, IBM declared the hardware/software complexity of networked IT (information technology) systems as the grand challenge for continued progress in this industry for the coming decades [1]. IBMs Autonomic Computing initiative was inspired by the way nature deals with complexity which is dominantly founded on the principle of emergent self-organization and self-awareness: a hierarchy of system constituents following few, simple rules which manifest a sophisticated system behavior through hidden causalities. The goal of Autonomic Computing was to integrate such self-x functions (where x stands for -control, -management, -organization, -healing, etc.) into IT equipment in order to attain higher degrees of autonomy in complex system operation.

Like IT systems, emerging Cyber-Physical Systems (CPS) and the Internet of Things (IoT) application domains exhibit a several-orders-of-magnitude increase in complexity, both in the number of devices, as well as in their dynamic and unpredictable interactions. Applying autonomy in this context demands a radically new strategy to conquer and control this runaway complexity. Towards this end, this paper focuses on novel EDA design methodologies enabling the adoption of self-awareness and self-organization paradigms into autonomous Multi-Processor System-on-Chip (MPSoC) platforms deployed in CPS and IoT applications. This approach exploits self-awareness principles in order to conquer complexity, achieve predictability and strengthen the robustness of system design. It represents a conceptual shift in platform design as exemplified by our Information Processing Factory (IPF) paradigm [2], with robust and independent platform operation capabilities focusing on futuristic platform-centric design, rather than the traditional focus on semiconductor devices and software technology.

To set the context, consider the trend towards highly automated driving, where performance requirements demand the use of highly complex manycore and massively parallel processors for critical applications. This includes signal processing, machine learning and automatic control with a rapidly growing set of data representing the vehicle state and its surroundings, other traffic participants and a global traffic context that must be processed in the vehicle or in the cloud. Traditional safety-critical systems in vehicles were much simpler and could be backed by simple monitoring functions to achieve fail-safe function behavior. Error scenarios in automated driving are far more complex when fail-safe requirements are upgraded to fail-operational behavior, because operation has to continue even under failure. This is especially true in SAE J3016 level-3 (and above) of automated driving, where the human driver is temporarily or permanently unavailable as a fail-over option. For highly automated driving, monitoring is not only required to enable fault detection and avoidance, but also to enforce operational boundaries on the system. An upcoming standard called Road Vehicles - Safety of the Intended Functionality (SOTIF) ISO/WD PAS 21448, provides guidance to address safety violations by a fault-free system due to unintended behaviors or technological limitations. As a consequence, automated driving increases diagnostic and monitoring requirements enabling early fault detection and failure avoidance, just as envisioned by the IPF approach described in this paper.

The key methodological innovation is a new approach to control platform dynamics at runtime by combining self-organizing machine learning techniques with formal reactive control methods providing platform worst-case real-time and safety guarantees, as embodied in our IPF paradigm [2], instead of using a single, static, operating point determined at design time. The authors are involved in parallel projects targeting self-aware vehicles for autonomous driving which will provide use cases for IPF research.

978-3-9819263-0-9/DATE18/©2018 EDAA

II. Related Work

Biologically-inspired mechanisms are at the roots of several research initiatives of which we summarize a short list in the sequel. Even though this list is by no means complete, it captures the most relevant related work in the context of this paper.

The DFG SPP1183 "Organic Computing" initiative [3], [4] has been most instrumental for developing and advancing self-organizing systems engineering in Germany. One key aspect of self-organizing systems is trust, raising the question of how to trust a system that changes all the time and reacts autonomously and potentially in unforeseeable ways due to emergent behaviors [5]. This question was addressed in the DFG Research Unit OC-Trust for the related case of Organic Computing. OC-Trust addressed trust, trustworthiness, and trust management as main challenges.

A similar observer/controller-based approach controlling emergent behavior is adopted in the EU-FP7 EPiCS (Engineering Proprioception in Computing Systems) project. It exploits self-awareness and self-expression in the compute platform, middleware software, distributed network infrastructure as well as application programs [6].

Self-expression describes the ability to autonomously adapt the system behavior under changing conditions. Both Organic and Proprioceptive Computing define general principles which need be applied in the proposed IPF, however certain methods are needed to reach the necessary robustness. More importantly, our IPF approach requires to provision hard real-time guarantees for critical tasks in the presence of changing platform properties which cannot be directly observed, calling for controlled system adaptation.

Within the EPSRC project entitled "Bio-Inspired Adaptive Architectures and Systems", a group from the University of York investigates the benefits of self-optimization to gracefully improve power-efficiency and performance of many-core systems [7]. These systems are thus able to cope with permanent and transient faults and continually seek to attain a more optimal system configuration by using a set of online optimization mechanisms on spare processing cores. Another work in self-aware architectures was contributed by Ipek and Martinez [8] where artificial neural network machine learning techniques were used to predict system performance as a function of resource allocation decisions at runtime. Similar to the other discussed projects, these two groups do not consider critical or mixed-critical applications.

The SElf-AwarE Computing (SEEC) framework at MIT [9], [10] attempts to reduce the programming burden of MPSoCs with a self-aware programming model. Applications explicitly specify system goals (for power and performance) and a unified decision engine attempts to adapt algorithms, software and hardware to optimally meet the set goals. A central concept of SEEC is the Observe-Decide-Act control cycle which is executed in cyclic time intervals in which a certain amount of application-specific work has to be accomplished. SEEC uses on-chip self-monitoring to guarantee predictable behavior on power-aware systems [11].

III. Operation Point Management For Mixed-Criticality Applications

A primary objective of the IPF paradigm is to demonstrate the feasibility of a hybrid (real-time predictive control combined with self-aware machine learning) approach to operate complex MPSoC-based hardware and software layers of CPS (and systems-of-systems) on platforms with dynamically changing platform properties and externally imposed mixed criticality workloads. Both platform uncertainties and system dynamics represent huge challenges to reach guarantees. Platform uncertainty arises from variability in semiconductor production and long term aging effects and from imprecise models used at design time. Application uncertainty can have different sources such as lack of application knowledge due to missing data or system complexity, or lack of trust. System dynamics can arise from platform dynamics, such as temperature variation, from short term computational workload dynamics, or from longer term application software change and evolution.

Platform control needs to consider input parameters such as, e.g., required computational workload, maximum temperature, or circuit delay to control output parameters, such as admitted load, load distribution, on-chip message routing, task scheduling parameters, supply voltage, clock frequency etc.

The platform has a state characterized by parameter values. The set of these state parameters forms the Operating Point (OP). The notion of an MPSoC operating point, its observation, control, and its successive adaptation are central aspects of this approach. A CPU core OP is, e.g., characterized by the task (set) mapped to a particular core, the scheduling policy as well as the core supply voltage and frequency (or DVFS policy with multiple V_{dd} and clock frequencies). The MPSoC or platform OP is the aggregate of all individual CPU, memory, accelerator and I/O unit OPs including their interaction.

Fig. 1 provides an abstract notion of the platform OP and permitted system variations depicting a two (out of the in

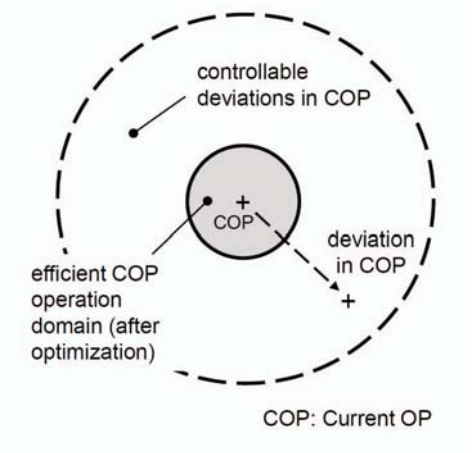

Fig. 1: Current operation point (COP) and its controllable deviation space

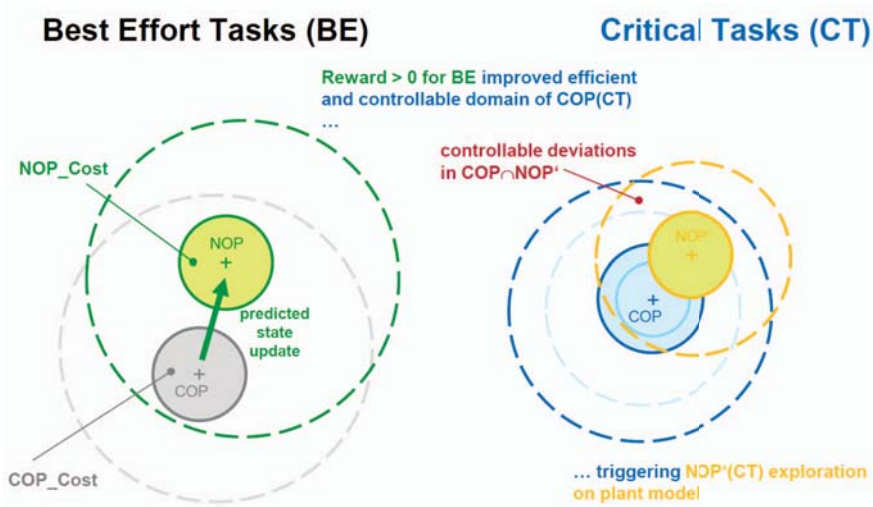

Fig. 2: Transition between control points for sake of system optimization

general n-) dimensional parameter space. The Current OP (COP) is the result of system optimizations to ensure the system will work correctly and efficiently. Minor changes around this COP will not defeat system efficiency if the system is robust enough. Larger OP changes due to changed state parameters (workload, frequency) will reduce system efficiency but are still acceptable up to a (maximum) dashed boundary beyond which the system will no more be controllable and/or crucial guarantees will be violated. Thus, Fig. 1 outlines the classical design where all possible behaviors may only lead to OPs in the dashed bounds. It should be clear that considering system dynamics and uncertainties will require large dashed bounds, resulting in an over-dimensioned design. No real-world system would be developed with such a strategy. Instead, the dashed boundary would be optimized (tightened) to the actual demands as shown in the shaded region of Fig. 1.

While in current design practice the abstract concept of an OP is not explicitly formulated, it is an underlying principle of critical systems design: the design must have enough "safety margin" to provide guarantees even under deviations from normal operating conditions. This requirement not only applies at design time, but prevails throughout the system lifetime. Ensuring this margin is an important task of maintenance which would replace or repair system parts or reconfigure the software if the margins are compromised. With increasing system variability and lifetime, relying on maintenance and updates alone is no option anymore, because this would be too inflexible and costly.

The IPF project investigates reducing maintenance needs by self-adaptation to save cost and extend a system's lifetime. Maintenance will still be necessary if self-adaptation is not possible anymore, e.g. in case of component failure. In critical system design, self-adaptation must consider the required margins. Unfortunately, available margins are not easily observable in a system under operation but must indirectly be derived from system operation and measurement. Therefore traditional self-diagnosis is not sufficient. For that purpose, an online system model with COP is needed, together with a strategy to incrementally adapt the COP to conserve the margins. This model must be continuously updated avoiding modeling errors, e.g., by methods from system identification [12]. Developing such an incremental process will be one of the core challenges of this US/German IPF collaboration. Once introduced, such a process can also be used to improve the efficiency of mixed criticality systems.

In order to quantify the quality of a given OP and to compare performance characteristics between different OPs, we investigate weighted, multi-objective cost functions (OPcost). The general idea being: the lower the average frequency and supply voltage of MPSoC modules, the closer the actual CPU core utilization, task execution time and timing slack are to currently set targets for utilization, execution time and slack (i.e., margin). Hence a particular COP is more effective when the cost metric OPcost of a platform is lowered. The OPcost metric shall also be used to initiate and assess OP changes, i.e., transitions from a COP to a Next Operation Point (NOP).

Fig. 2 suggests that system control shall assess and explore the OPcost-based platform state update for the NOP separated for critical tasks (CT) and non-critical, best effort tasks (BE). NOP adoptions for BE tasks may be attained by means of periodic or threshold-triggered rule-table actions of either hardware or software-based machine learning (ML) entities. Applying an ML action anyway results in a change of MPSoC operation parameters and, thus, in adopting a new NOP. Such a change in the operating point for BE tasks (which, per definition, cannot violate a critical system-level requirement as BE tasks are"best effort") may however improve/increase the stability regions for critical tasks (see bigger dark blue circles on the right of Fig. 2). This would open new system optimization opportunities for critical tasks. Controlled transitions between COP(CT) and NOP(CT) must be safe by all means. Of course we must avoid by all means threats to the real-time and safety critical system functions, if the transition is too fast or leads to a NOP outside the controllable state.

Fig. 2 again illustrates the anticipated effect. Using machine learning-based analysis and applied actions make the platform

move to a new NOP for best effort tasks. System control and reactive resource management will assess NOP' (Next Operation Point Candidates) for critical tasks and, if the predictions proved to be safe, the NOP'(CT) candidate will be adopted as NOP(CT). In general, we need to investigate mechanisms to ensure constructive emergent behaviors within deterministically bounded operation corridors at MPSoC hardware/software levels. Thus, a solution must be found that still guarantees timing and safety under possible prediction errors. If the update and optimization steps are small enough, the system stays in an efficient state and keeps the required guarantees at all times.

IV. Information Processing Factory

We use the metaphor of an Information Processing Factory (IPF) to draw similarities between microelectronics systems and factories [2] as follows: in a factory, all the components must adapt to the current workload. This includes logistics of supplies such as material, energy, water and waste, the manufacturing equipment, the transport, the control and infrastructure (such as heating, air-conditioning, illumination). This adaptation cannot be done offline and must instead be done in real time without interrupting the baseline operations. Future microelectronic systems (e.g., MPSoCs) should operate in a similar manner. Parallel to the baseline operation of the system, a platform operation layer (POL) is continuously monitoring and controlling the performance and health status of the system. This layer monitors the system using a network of on-chip infrastructure that senses cross-layer metrics such as temperature, aging, energy, performance, reliability, and security and accordingly orchestrates the operations of different system components such as application, storage, I/O and even non-electronic functions (e.g. micromechanics, microfluidics, etc.). In performing this orchestration, the POL will consider both the current status (COP) as well as prediction of future states including the expected evolution of the platform (i.e. NOP as well as NOP'(CT) candidates in Fig. 2).

This IPF analogy implies that clusters of component-specific, uncorrelated control occurrences are unable to cope with the complexity coordinating the operations of large scale systems with multi-criteria objective functions. Similarly, a centralized controller model is also inadequate in this case because it cannot scale. Our goal is to demonstrate that a hybrid hierarchical approach, sporting as much modularity as possible and as much centralized as necessary, is a much more effective means of achieving the desired goal while maintaining cost efficiency, low overhead, and scalability. To be more specific, the IPF design flow envisions a multi-layer control system conceptually described in Figure 3. Information provided by Sensor (S) is gathered and merged into self-organization, self-awareness (SO/SA) control processing instances across different hardware/software abstraction layers comprising an MPSOC-based CPS system. These SO/SA instances generate actuation directives affecting the MPSoC system components at same or lower levels of abstraction. SO/SA instances will be endowed with well-defined degrees of autonomy to optimize

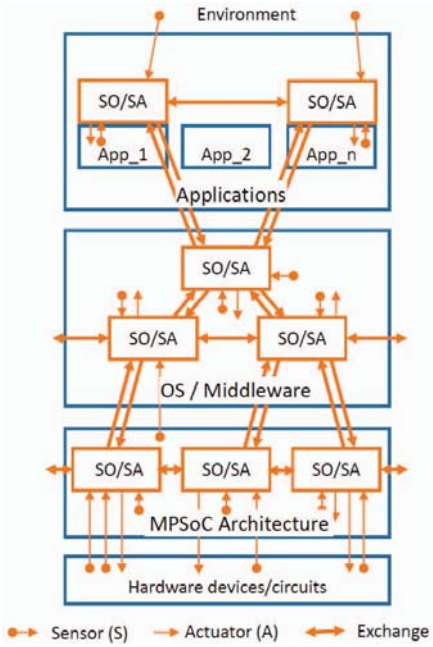

Fig. 3: Multi-Layer SO/SA stack

metrics such as performance, power, and reliability within its own scope of operation. The coordination among these SO/SA instances can be established through awareness of the actuation-to-implication causalities triggered by an action (for example, the impact of frequency scaling or task scheduling on power consumption and performance).

The SO/SA paradigm is not limited in scope to optimization of CPS operational parameters/metrics. In fact, self- and group-awareness can also enable higher level tasks such as self-protection of both the MPSoC and the overall CPS system. It must be noted, however, that empowering multiple SO/SA entities with some or complete autonomy may lead to differing or even contradictory control decisions. Such cases must be strictly avoided especially in safety-critical systems. One possible way to achieve that is to reject NOP'(CT) candidates that would put critical tasks at risk. Our goal is to demonstrate that self-awareness of implications among concurrent SO/SA instances can indeed prevent such situations from occurring.

The following subsections *A-D* will reveal in conjunction with Fig. 3 the major constituents of such a SO/SA hierarchy at different hardware/software abstraction levels and how we anticipate their inter-working.

A. Application

The Cognitive Runtime Management project from UC Irvine focuses on two layers: (1) the application/ecosystem layer where a wireless system is presented as an example of an application-aware cognitive power management employing a Q-learning based paradigm to optimize power management policy for a target application performance level (e.g. Bit error rate), and (2) the application monitoring and checking layer, comprised of a hardware-accelerated *non-uniform verification architecture* (NUVA), optimized to monitor a rich set of user-defined properties (or properties automatically extracted using

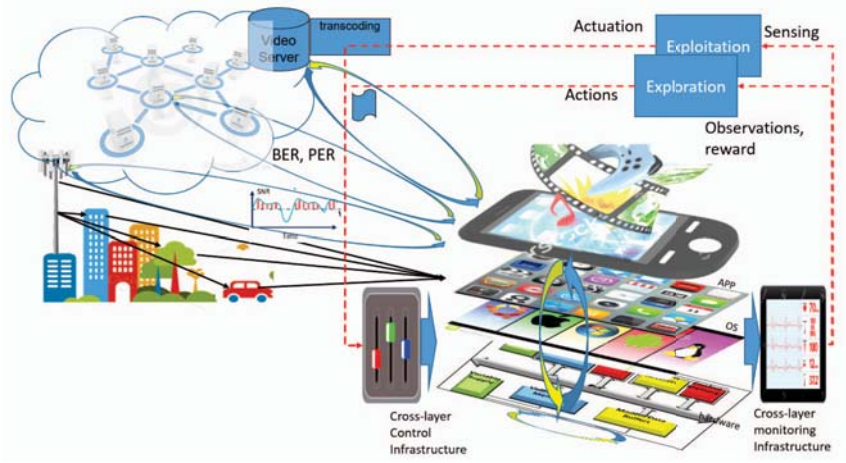

Fig. 4: Intra- and inter-system dependencies of mobile applications

specification mining) defined in an expressive specification language (such as parameterized automata) [13].

To manage the power, performance and reliability in a complex system such as the mobile device (or base station), there are several point techniques, each dealing with one or more knobs and having a limited scope as shown in Fig. 4.

Some techniques may focus on the modem while others on the source codec (e.g., video, gaming). Other techniques may apply inside the processor (e.g., cache power management) while others may exploit the mobile-basestation protocols, the network protocols, or the client-server interaction at the application level. The SO/SA paradigm described in Fig. 3 allows each of these techniques to run with some degree of autonomy while the interdependencies between them is managed through high level monitoring and actuation-to-implication awareness.

B. Deterministic On-Chip Interconnect Resource Management

TU Braunschweig has a long history of contributions to systems self-organization. Most recently, Networks-on-Chip (NoCs) for mixed criticality systems [14] using active resource management including error detection and correction mechanisms [15] have been developed. The results shall be combined with an in-field monitoring, safety analysis and self-configuration framework that was developed as a contribution to a larger DFG Research Group in self-aware vehicles [16]. Both results will be combined in IPF. TU Braunschweig is developing algorithms to analyze the margin of critical tasks and proposes incremental NOP changes as shown in Fig. 1 and 2. These functions will be located in the OS / middleware layer of Fig. 3. The NoC and its resource management requires specific hardware functions on the MPSoC architecture layer where monitoring functions will be installed, as well. On the Applications layer, use cases for autonomous driving will be implemented which are currently developed in the DFG research group.

C. Self-awareness at the Middleware Software and Application Layer

The CPSoC (Cyber Physical System on Chip) project from UC Irvine [17] is an exemplar sensor-rich many-core computing platform that intrinsically couples on-chip and cross-layer physical and virtual sensing and actuation applied across different layers of the hardware/software system stack to adaptively achieve desired objectives and Quality-of-Service (QoS). The CPSoC project came out of the NSF Variability Expedition project [18]. The CPSoC platform takes advantage of an Adaptive-Reflective Middleware which sits between applications and the operating system kernel. The middleware implements closed-loop resource management policies and embeds hierarchical system models that use sensory data to predict how the system state may change given new actuation actions. The middleware offers scalable and autonomous resource management by leveraging formal supervisory control theory combining the strengths of classical control theory with state-of-the-art heuristic approaches to efficiently achieve changing run-time goals [19]. It also provides robustness and stability guarantees to the power management unit by using an adaptive control theoretic technique called Gain Scheduling which decomposes the entire nonlinear operating region of the DVFS controller into linear sub-regions and adaptively switches between static linear feedback controllers designed for each operating sub-region [20].

D. Self-Awareness at the MPSoC Hardware Layer

The ASoC (Autonomic System on Chip) project (TU München and University of Tübingen) [21] [22] within the DFG SPP1183“Organic Computing“ investigated the application of hardware and software reinforcement machine learning techniques (learning classifier systems) [23] on homogeneous multicore processors for optimizing workload balancing, power consumption and resilience against intermittent core failures. Self-organized RISC core behavior was accomplished through hardware-centric reinforcement-based learning classifier tables (LCTs) [24].

In this context, a key challenge is to extend the reinforcement-based machine learning control towards mixed-criticality applications, including hard real-time constraints. Towards this goal, we anticipate the interworking of LCT-based control with predictable reactive resource control as

described in Section IV-B. Furthermore, the combination of LCT with Cyber Physical SoC (CPSoC) leads towards a hierarchical learning environment with goal-oriented supervisory control properties and allows the online generation of new LCT evaluation rules on the device at runtime. With respect to Fig. 3 this will be the establishment of connections between MPSoC Architecture and OS/Middleware layers. An interesting problem here is to investigate the relation between the learning layers and the influences they can have on each other. A possible scenario for this case would be that each layer learns and acts on its own without any collaboration with the other layer. The more challenging, yet more promising approach, is to find suitable collaboration models between the layers. In context of the IPF project this open problem will also be addressed by investigating a top down approach, in which the Middleware layer in Fig. 1 sends information to the hardware layer, as well as a collaborative approach, in which both layers are actively in connection with each other and share information, hence acting toward the best operating point possible for the whole system stack.

V. Conclusion

The existing dilemma in developing and adopting design methodologies for autonomous systems is that including all possible uncertainties and dynamics of a processing platform and its operational environment already at design time is either impossible or leads to unacceptable overengineered designs. On the other hand, strictly limiting the allowed design space to design-time decision making, will constrain the possible applications and their dynamics, error handling options, flexibility and life time of a system. Thus, in order to conquer the complexity of future MPSoCs and Cyber-Physical Systems, several challenges must be embraced, such as change management of the running application portfolio as well as dynamic runtime reconfiguration of the computing, interconnect and memory resources.

A promising approach to tackle system complexity in general is the use of self-awareness which is defined as the ability (of a computing system) to recognize its own state, possible actions and the result of these actions on itself and its environment. The IPF paradigm described in this paper presented an approach that exploits self-awareness in order to solve the dilemma by introducing a hybrid hierarchical approach across multiple hardware/software abstraction layers, which is highly scalable and modular and can be incorporated in different applications. Moreover the combination of self-organizing learning and formal reactive methods will guarantee the use of such systems in safety-critical and high-availability applications like autonomous driving.

Acknowledgment

We acknowledge financial support from the following: NSF Grant CCF-1704859; the Marie Curie Actions of the European Union's H2020 Programme; DFG Grants ER168/32-1 and HE4584/7-1.

References

[1] J. O. Kephart, "Research challenges of autonomic computing," in *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, May 2005, pp. 15–22.

[2] N. D. Dutt, F. J. Kurdahi, R. Ernst, and A. Herkersdorf, "Conquering mpsoc complexity with principles of a self-aware information processing factory," in *CODES*, 2016, pp. 37:1–37:4.

[3] C. Müller-Schloer, "Organic computing - on the feasibility of controlled emergence," in *CODES + ISSS 2004.*, Sept 2004, pp. 2–5.

[4] C. Müller-Schloer *et al.*, *Organic Computing A Paradigm Shift for Complex Systems*. Springer, 2011.

[5] J.-P. Steghöfer *et al.*, *Autonomic and Trusted Computing: 7th International Conference, ATC 2010, Xi'an, China, October 26-29, 2010. Proceedings*. Springer Berlin Heidelberg, 2010, pp. 62–76.

[6] T. Becker *et al.*, "Epics: Engineering proprioception in computing systems," in *2012 IEEE 15th International Conference on Computational Science and Engineering*, Dec 2012, pp. 353–360.

[7] N. Dahir, P. Campos, G. Tempesti, M. Trefzer, and A. Tyrrell, "Characterisation of feasibility regions in fpgas under adaptive dvfs," in *FPL*, Sept 2015, pp. 1–4.

[8] R. Bitirgen, E. Ipek, and J. F. Martinez, "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach," in *MICRO*, ser. MICRO 41. Washington, DC, USA: IEEE Computer Society, 2008, pp. 318–329.

[9] H. Hoffmann *et al.*, "Eec: A framework for self-aware computing," Massachusetts Institute of Technology, Cambridge,, Tech. Rep., oct 2010.

[10] H. Hoffmann *et al.*, "Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments," in *ICAC*, ser. ICAC '10. ACM, 2010, pp. 79–88.

[11] H. Hoffmann, "Coadapt: Predictable behavior for accuracy-aware applications running on power-aware systems," in *2014 26th Euromicro Conference on Real-Time Systems*, July 2014, pp. 223–232.

[12] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[13] A. Nassar, F. J. Kurdahi, and W. Elsharkasy, "Nuva: Architectural support for runtime verification of parametric specifications over multi-cores," in *CASES*, 2015, pp. 137–146.

[14] A. Kostrzewa, S. Tobuschat, and R. Ernst, "Self-aware network-on-chip control in real-time systems," *IEEE Design Test*, vol. PP, no. 99, pp. 1–1, 2017.

[15] E. A. Rambo, C. Seitz, S. Saidi, and R. Ernst, "Bridging the gap between resilient networks-on-chip and real-time systems," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[16] J. Schlatow *et al.*, "Self-awareness in autonomous automotive systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1050–1055.

[17] S. Sarma, N. Dutt, P. Gupta, A. Nicolau, and N. Venkatasubramanian, "On-chip self-awareness using cyberphysical-systems-on-chip (cpsoc)," in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2014, pp. 1–3.

[18] P. Gupta *et al.*, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 8–23, Jan 2013.

[19] A. Rahmani, B. Donyanavard, T. Mück, K. Moazemmi, A. Jantsch, O. Mutlu, and N. Dutt, "SPECTR: Formal Supervisory Control and Coordination for Many-core Systems Resource Management," in *ASPLOS*, 2018.

[20] B. Donyanavard, A. Rahmani, T. Mück, K. Moazemmi, and N. Dutt, "Gain Scheduled Control for Nonlinear Power Management in CMPs," in *DATE*, 2018.

[21] A. Bernauer, O. Bringmann, and W. Rosenstiel, "Generic self-adaptation to reduce design effort for system-on-chip," in *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, Sept 2009, pp. 126–135.

[22] J. Zeppenfeld and A. Herkersdorf, "Applying autonomic principles for workload management in multi-core systems on chip," in *ICAC*. ACM, 2011, pp. 3–10.

[23] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems*. Springer, ISBN 978-3-540-25379-2, 2006.

[24] J. Zeppenfeld, A. Bouajila, W. Stechele, and A. Herkersdorf, "Learning classifier tables for autonomic systems on chip." vol. 2, 01 2008, pp. 771–778.