Toward Smart Embedded Systems: A Self-aware System-on-Chip (SoC) Perspective

NIKIL DUTT, University of California Irvine
AXEL JANTSCH, TU Wien, Austria
SANTANU SARMA, University of California Irvine

Embedded systems must address a multitude of potentially conflicting design constraints such as resiliency, energy, heat, cost, performance, security, etc., all in the face of highly dynamic operational behaviors and environmental conditions. By incorporating elements of intelligence, the hope is that the resulting "smart" embedded systems will function correctly and within desired constraints in spite of highly dynamic changes in the applications and the environment, as well as in the underlying software/hardware platforms. Since terms related to "smartness" (e.g., self-awareness, self-adaptivity, and autonomy) have been used loosely in many software and hardware computing contexts, we first present a taxonomy of "self-x" terms and use this taxonomy to relate major "smart" software and hardware computing efforts. A major attribute for smart embedded systems is the notion of self-awareness that enables an embedded system to monitor its own state and behavior, as well as the external environment, so as to adapt intelligently. Toward this end, we use a System-on-Chip perspective to show how the CyberPhysical System-on-Chip (CPSoC) exemplar platform achieves self-awareness through a combination of cross-layer sensing, actuation, self-aware adaptations, and online learning. We conclude with some thoughts on open challenges and research directions.

Categories and Subject Descriptors: C.2.2 [**Smart Embedded Systems**]: Computing Architecture

General Terms: Smart Embedded System, System-on-Chip, Run-time System, Self-Awareness, Self-Adaptation, Autonomous System

Additional Key Words and Phrases: Index terms—dynamic power management, dynamic reliability management, reliability modeling, variability, resilience sensor, resilience estimation, prediction

ACM Reference Format:
Nikil Dutt, Axel Jantsch, and Santanu Sarma. 2016. Toward smart embedded systems: A self-aware system-on-chip (SoC) perspective. ACM Trans. Embed. Comput. Syst. 15, 2, Article 22 (February 2016), 27 pages.
DOI: http://dx.doi.org/10.1145/2872936

1. INTRODUCTION

The ubiquitous deployment of computing in virtually every facet of today's society has led to the colloquial usage of terms as such "embedded computing," "CyberPhysical Systems," and, more recently, the "Internet of Things (IoT)." At the heart of such systems are software/hardware computing platforms that interact with the physical

This work was partially supported by the National Science Foundation, under grant CCF-1029783 (Variability Expedition).
The authors are in alphabetical order and contributed equally to this manuscript.
Authors' addresses: N. Dutt, Department of Computer Science, Zot Code 3435, Donald Bren School of Information and Computer Sciences, University of California, Irvine, Irvine, CA 92697-3435, USA; email: dutt@ics.uci.edu; A. Jantsch, Gußhausstraße 27-29, 1040 Vienna, Austria; email: jantsch@ict.tuwien.ac.at; S. Sarma, Computer Science Department, University of California Irvine, 8730, Costa Verde Blvd, Apt # 2459, San Diego, CA -92122, USA; email: santanus@uci.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 1539-9087/2016/02-ART22 $15.00
DOI: http://dx.doi.org/10.1145/2872936

world through sensors, actuators, communication/networking, and decision-making engines. These systems range from the tiniest of embedded devices (e.g., small sensor motes [Hengstler 2007; Polastre 2005; Fang 2005; Stoianov 2007; Shnayder et al. 2005; Sudevalayam and Kulkarni 2011]) to complex system-of-systems, such as autonomous swarms of robots [Rubenstein et al. 2012, 2014] and complex human-in-the-loop systems (e.g., an Airbus 330 [INAGAKI 2005; Hoffman et al. 2007]). For the purpose of this article, we refer broadly to all of them as "embedded systems." A common characteristics across this diverse set of embedded systems is the need to operate correctly in the face of highly dynamic environmental conditions and changing application characteristics, as well as changes in the computing platforms itself (e.g., degradation or failures). Moreover, depending on the embedded system context, their architectures are highly customized to achieve the often conflicting constraints of performance, energy efficiency, cost savings, reliability, and the like. Furthermore, the complexity of the embedded software and hardware can vary over several orders of magnitude depending on the application domain, the usage context, and their physical deployment. Consequently, designers of embedded systems aim to increase the level of "smartness" in these systems to adapt seamlessly to changes, increase the level of autonomous operation, and incorporate learning strategies. Depending on the type of embedded system, these needs typically translate into guarantees for functional and nonfunctional constraints, adaption to dynamism, and the ability to tolerate failures.

With the increasing complexity of tasks faced by embedded systems, the designers of software and hardware systems have naturally borrowed concepts from biological systems in an attempt to mimic their ability to be self-aware, evolve, and achieve a high level of resilience in the face of highly dynamic and unpredictable environments. A large body of research in intelligent autonomous systems, agent-based distributed systems, and advanced control theory have all used variants of the phrase "self-x," where "x" variously refers to capabilities such as awareness, healing, optimization, adaptation, and the like. Unfortunately, with this alphabet soup of terminology, there is little consensus of what these terms mean in the context of software and hardware systems and for embedded systems in general. To disambiguate loosely used terminology in the embedded systems literature, in Section 2, we begin by reviewing notions of self-awareness, self-adaptivity, and autonomic systems in the large body of the literature in cognitive sciences (Section 2.1) and on large software systems (Section 2.2), embedded systems (Section 2.3), and Systems-on-Chips (SoCs; Section 2.4). We then propose a taxonomy in Section 3 to structure the terminology and related work.

In spite of the bewildering diversity of embedded systems in general, at the core of all these embedded systems are integrated circuits made of silicon. As the number and variety of those devices grow exponentially, it becomes increasingly harder to guarantee perfect functionality and performance over the entire lifetime of these devices. This article will therefore focus on Smart Embedded Systems (SESs) primarily from the perspective of a SoC, and the associated challenges for developing software and hardware platforms upon which reliable, autonomous, and SESs can be built.

Integrated circuit technology has reached the nanoscale era, introducing a multitude of challenges stemming from the end of perfect Dennard scaling [Esmaeilzadeh et al. 2011; Raghavan et al. 2012; Ferdman et al. 2012] and worsening process variations [Borkar et al. 2003]. "Dark silicon" will be a defining feature of future SoCs, where only small portions of a chip may be powered on at a time in order to manage power density and heat [Esmaeilzadeh et al. 2011]. Further complicating matters, systems are typically not very energy proportional due to high static power and a dearth of active low-power modes aside from CPU Dynamic Voltage/Frequency Scaling (DVFS) [Barroso and Hölzle 2007; Barroso et al. 2013]. The dark silicon phenomenon and the need for greater energy proportionality and efficiency are major driving forces behind research and development in many-core SoCs and heterogeneous devices, architectures, and systems. In particular, many-core computational platforms already face significant resiliency challenges, with errors resulting from manufacturing process variability, exponentially increasing power dissipation and heating, environmental effects (e.g., radiation-induced soft errors [Baumann 2005]), and aging/wearout [Bernstein et al. 2006]. These problems are exacerbated in the nanometer era with exploding core counts and on-chip resources. The combined systemic and random effects in nanoscale technologies result in high variability (and thus higher error rates) manifested from the circuit level all the way to the architecture and system levels [Borkar et al. 2003], requiring new strategies for ensuring application resilience when executing on these computing platforms. Therefore, in Section 4, we present the CyberPhysical System-on-Chip (CPSoC) platform as an exemplar for a self-aware, sensor-actuator-rich SoC platform that incorporates some of the key features required to deliver SESs, including: dynamic balancing of multiple objectives, effective management of limited on-chip resources, self-monitoring and self-awareness to enable adaptation, and learning mechanisms to allow the system to evolve over time.

We conclude in Section 5 with a brief discussion of opportunities, challenges, and research directions for the overall topic of SESs.

2. SELF-AWARE, SELF-ADAPTIVE, AND AUTONOMIC SYSTEMS

2.1. Theories of Cognition

Cognitive science has a long and rich history of theories about the mental faculties that link perception to action [Vernon et al. 2007]. Two main categories can be distinguished: *cognitivist* and *emergent systems* approaches. The cognitive paradigm is based on the classic view that cognition is a “kind of computation” that uses symbolic and abstract representations of the real world and that algorithmically calculates actions [Pylyshyn 1984]. In contrast, proponents of the emergent systems paradigm, which includes *connectionist*, *dynamical,* and *enacting systems* approaches [Vernon et al. 2007], argue that cognition is an emerging phenomenon in self-organizing, dynamic systems that interactively identify and use regular patterns in the environment to continuously adapt, react, and anticipate [Thelen and Smith 1994; Clark 2001]. Cognitivist approaches assume the existence of an objective reality that should be abstracted and symbolically represented, whereas the emergent systems community views the system and its environment as mutually dependent and continuously co-evolving.

The CPSoC approach described in Section 4 resembles and is inspired by the emergent systems paradigm in that it provides for simple faculties of monitoring, actuating, and control in a bottom-up manner to allow larger systems to evolve more potent capabilities as individual and as groups of collaborating systems. However, it hopes to short-cut long learning cycles by equipping systems with key capabilities without waiting for their emergence through evolution.

One cognitive theory of consciousness, which falls into the emergent systems camp and which is relevant for our topic, is Baars’s *Global Workspace Model* (GWM) [Baars 1989]. Many parallel processes operate unconsciously and concurrently, but only one, or one coalition of processes, obtains access to the global workspace at any time, allowing it to broadcast its message globally and thus to marshal many global resources for its purpose. Hence, the global workspace serves to allocate and synchronize limited resources. Since its formulation, many phenomena predicted by the GWM have been confirmed in experiments [Baars 2002; Baars and Franklin 2009] making it today the top contender for explaining consciousness.

2.2. Awareness in Software Systems

The insight that a sense of awareness can facilitate robust and dependable behavior even under radical environmental changes and drastically diminished capabilities have

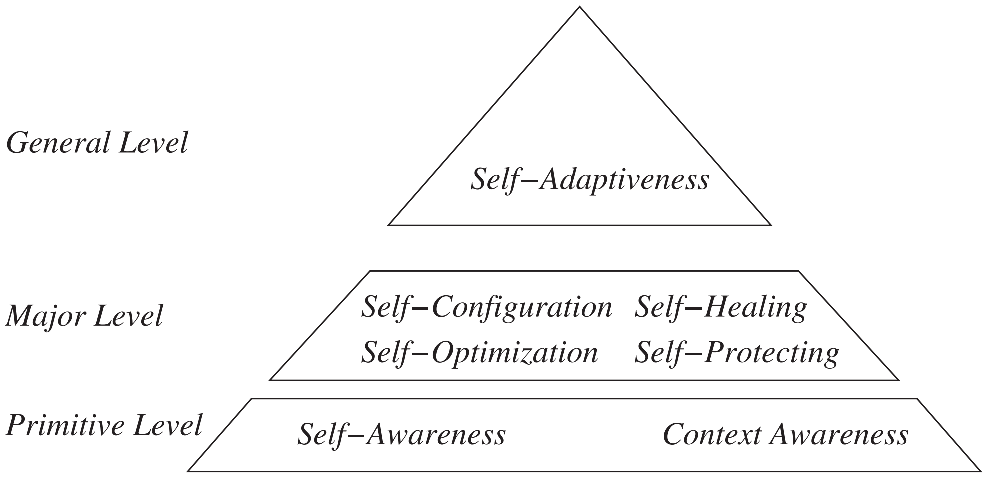

Fig. 1. Hierarchy of self-* properties [Salehie and Tahvildari 2009].

inspired researchers to study the utility of cognitive features like adaptivity, awareness, or consciousness for robots, large scale-software systems, and embedded systems. The benefits are more obvious for some features such as adaptivity but less for awareness or consciousness. Hence, adaptivity, and in particular self-adaptivity, has been the focus of much research. In complex software systems, self-adaptivity is expected to help in managing the complexity [Salehie and Tahvildari 2009]. Manual troubleshooting, reconfiguration, and maintenance are demanding and error prone. Above a certain complexity of the system, these become infeasible. Self-adaptive behavior is triggered either by changes of the system's *self* (internal causes like faults or mode transitions) or by changes of the system's *context* (external events like changes in user request rates or user objectives) [Salehie and Tahvildari 2009].

In 1997, a DARPA Broad Agency Announcement offered a definition of self-adaptive software: "Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible" [Laddaga 2001], which points to several important aspects: (i) The system monitors its own behavior, (ii) it knows what behavior is expected, (iii) it compares its observed behavior to the expected behavior, and (iv) performance matters in addition to functionality. At about the same time, Oreizy et al. highlighted the importance of the environment: "Self-adaptive software modifies its own behavior in response to changes in its operating environment" [Oreizy et al. 1999], which requires that (v) the system monitors its environment, (vi) knows what behavior of the environment is expected, and (vii) knows its own appropriate behavior for a given environment.

Features (i)–(iv) are related to *self-awareness* and (v)–(vii) to *context-awareness,* which form the bases of all self-modifying capabilities such as self-configuration, self-optimization, or self-adaptivity. In the often used hierarchy of self-x properties, these are located in the *primitive level* below the *major level* that is populated by specific self-changing capabilities [Salehie and Tahvildari 2009], as illustrated in Figure 1.

IBM's original vision of autonomic computing [Kephart and Chess 2003], formulated in the early 2000s, puts its emphasis on the upper levels, implicitly assuming that awareness is a simple capability. In contrast, we argue that achieving awareness is hard, but, once achieved, realizing the higher level properties at the major and general levels are difficult but tractable engineering tasks.

In reaction to the DARPA initiative and IBM's vision, research on self-x properties has flourished. Two recent surveys on self-healing by Ghosh et al. [2007] and [Psaier and Dustdar 2011] discuss the approaches taken for detecting and reacting to faulty states of a system. Self-healing is rooted in work on fault-tolerant and self-stabilizing [Dijkstra 1974] systems but emphasizes continuous availability and focuses on the recovery process. Both surveys agree that a kind of self-awareness is critical, but they often view it narrowly as a mechanism to detect faults, which then triggers recovery procedures. Hence, the system perceives itself to be in one of two states: healthy or not healthy. Our understanding is broader and implies a richer perception of a system's own well-being and performance that allows for a nuanced assessment as to which degree expectations and goals are met including a track record and a sense of historical performance.

Partially overlapping are efforts to design self-adaptive systems as elaborated from a variety of aspects in a book edited by Cheng et al. [2009a]. A self-adaptive system is more general than a self-healing system because it also adapts gracefully to changing environmental conditions. Again, publications on self-adaptivity view self-awareness rather narrowly as a means to detect unusual states and focus mainly on the reaction to such observations. However, it has been noticed that a more comprehensive approach to self-awareness aspects would be both desirable and challenging. For instance, Cheng et al. [2009b] note that knowledge of expectations by the environment, for which Finkelstein has coined the term *requirements reflection*, would be useful and conclude that "Future work is needed to develop technologies to provide such infrastructure support" [Cheng et al. 2009b].

In control applications, models of the self have reached significant sophistication. Kaindl et al. propose an explicit, symbolic representation of self for the purpose of monitoring and self-configuring the system based on changing needs and requirements [Kaindl et al. 2013]. An emotion-based approach to assess the inner state and the wellness of a system is described by Sánchez-Escribano and Sanz [2014]. They use a prioritization mechanism to compare and relate the importance of otherwise independent states or events and call it "emotion." Sanz et al. have gone furthest by incorporating an explicit self-model in the control system, one elegantly based on the model of the system used during the design process [Sanz et al. 2007] and resembling requirements reflection mentioned previously. This establishes a secondary control loop in which the primary control algorithm can be adapted.

A. Morin [2006] has formulated nine neurocognitive models of self-awareness distinguishing *unconsciousness*, *consciousness of external stimuli and events*, *self-awareness of public and private self-aspects*, and *meta self-awareness*. Based on Morin's classification, Lewis et al. [2011] consider categories of self-awareness with respect to their relevance to computing systems. They offer a working definition distinguishing between information that a system has about its own state (*private self-awareness*) and knowledge about how it is perceived by its environment (*public self-awareness*). Also, organization of self-aware systems in groups of peers leading to group-awareness is considered. The categorization outlined in Section 3 [Jantsch and Tammemäe 2014] is consistent and to a large degree aligned with Lewis et al.'s definition, but our concepts are more detailed and formulated with the objective of engineering self-aware systems under tight resource constraints.

Chen et al. have proposed a pattern-based approach to the design of self-aware systems [Cheng et al. 2014]. Based on Lewis's classification, they formulate seven patterns for specific functions relevant to self-awareness: *basic information sharing, coordinated information sharing, temporal knowledge sharing, temporal knowledge awareness, goal sharing, temporal goal awareness*, and *meta-self-awareness*. Architectural patterns and a methodology for designing self-aware and self-expressive systems are formulated and applied to case studies with cloud computing and a smart camera networks application.

2.3. Awareness in Embedded and Cyber-Physical Systems

In *embedded* and *Cyber-Physical Systems* (CPS), the main problem is not so much the size-induced complexity of an individual system but rather the tight resource constraints, the large number of those systems and their interaction, and the unpredictable environmental conditions of their deployment. Analysts expect 26 billion devices connected to the Internet of Things by 2020 (www.gartner.com/newsroom/id/2636073). Manual maintenance, diagnostics, and repair of most of these devices will soon be impossible. Thus, there is a growing need for CPSs to have a better understanding of their own state, their behavior, their performance, and the surrounding conditions. We call this "better understanding" *awareness*, which improves the behavior of systems, making them more robust while reducing processing, communication, and energy requirements. A variety of bio-inspired approaches have been proposed for the operation, modeling, design, optimization, and verification of embedded systems and SoCs, as a recent collection illustrates [Cong-Vinh 2011]. For instance, Zakaria et al. [2011] describe techniques to handle uncertainties because of faults due to process variation and limited yield in the management of power consumption and synchronization between different clock domains in SoCs. Evolvable hardware is hardware that can change its architecture and behavior dynamically and autonomously [Yao and Higuchi 1999; Higuchi et al. 2006]. The hardware design is encoded in some kind of "chromosome," and evolutionary techniques such as genetic algorithms are deployed to modify this chromosome and thus the hardware as a reaction to a changing environment or faulty components. Because Field-Programmable Gate Arrays (FPGA) provide a perfect medium for the implementation of evolvable hardware, the research field has flourished since the advent of FPGAs in the 1990s and is continuously exploiting FPGA features as they emerge [Cancare et al. 2011].

However, designing and implementing self-awareness in an ad-hoc manner for every new system is not feasible. Introducing awareness as a separate concept in the CPS infrastructure promises to simplify the development and operation of such systems. Because CPSs are typically Systems-of-Systems (SoS), the awareness must be solved comprehensively, ensuring that the understanding of the situation is coherent and consistent across the SoS.

Bakhouya and coworkers draw more explicit parallels to natural phenomena such as the immune system, cell organization, and ant colonies [Bakhouya 2011; Bakhouya and Gaber 2014]. They correctly put emphasis on positive and negative feedback loops that are pervasive in natural systems and a key in the design of adaptive behavior in SESs.

Awareness is not necessarily confined to individual components; it may just as well emerge in cooperating SoSs. Preden and coworkers have studied distributed surveillance systems and assign particular importance to the role of attention and context-aware processing and sensing [Motus et al. 2009; Preden 2012; Preden et al. 2013; Preden 2014]. They argue that these properties facilitate efficient operation of distributed sensing systems. Based on Endley's *situation awareness* [Endsley 1988], Preden et al. have developed the concept of *situation parameters* [Preden et al. 2013]. A situation is defined by the values and interpretation of a set of situation parameters, which are monitored or computed independently and represent a property of the situation of interest. The information for generating situation awareness is collected and processed independently of the application functionality and can be considered as part of the CPS platform.

Table I. Smart Dynamic Reliability/Resilience Management

	Adaptation Type		Sensing and Monitoring					Decision Making Layer					Cross-Layer Actuation Level				
References	Simple*	Self-Aware†	Ckt	HW	NW	OS	App	Ckt	HW	NW	OS	App	Ckt	HW	NW	OS	App
[Shapiro 2004]	✓	Self-heal	✓	✓	–	✓	–	–	–	–	✓	?	–	–	–	✓	?
[Sylvester et al. 2006]	✓	Self-heal	✓	✓	–	–	–	✓	✓	–	?	–	✓	✓	–	✓	–
[Karl et al. 2006]	–	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Austin et al. 2008]	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Sun et al. 2009]	✓	–	✓	?	–	✓	–	✓	?	–	✓	–	✓	?	–	✓	–
[Das et al. 2009]	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Reddi et al. 2009]	✓	–	✓	✓	–	–	–	✓	✓	–	?	–	✓	✓	–	?	–
[Reddi et al. 2010]	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Leem et al. 2010]	✓	–	✓	✓	–	?	✓	✓	✓	–	?	✓	✓	✓	–	?	✓
[Reddi et al. 2012]	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Kleeberger et al. 2013]	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–	–
[Mercati et al. 2013]	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–
[Li et al. 2013]	✓	–	✓	✓	–	–	✓	✓	✓	–	–	✓	✓	✓	–	–	✓
[Rehman et al. 2014]	✓	–	✓	✓	–	?	✓	✓	✓	–	?	✓	✓	✓	–	?	✓
[Mercati et al. 2014a]	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–
[Mercati et al. 2014b]	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–	✓	✓	–	✓	–

***Implicit Model †Explicit Model, ? Not explicitly discussed; Ckt = Circuit, HW = Hardware, NW = Network, OS = Operating System, App = Application.**

Witnessing the high interest in this topic are surveys on related and relevant topics such as on-chip self-monitoring [Kornaros and Pnevmatikatos 2013a], bio-inspired hardware design [Cong-Vinh 2011], and situation identification techniques [Ye et al. 2012].

2.4. Examples of Smart Embedded Systems

There is a large body of literature on SoCs developed for embedded and CPSs that exhibit self-awareness characteristics at various levels. We have listed an incomplete set of examples focusing on reliability and power management in Tables I and II, respectively. A number of German national projects have focused on computing systems that incorporate self-x properties, including the Organic Computing project [Organic Computing], the InvasIC project [Henkel et al. 2011], and the SPP1500 project on dependable embedded systems [Henkel et al. 2012]. There is also a wealth of research on power management, thermal management [Brooks and Martonosi 2001; Coskun et al. 2008; Ebi et al. 2009; Sarma and Dutt 2014b], and, more recently, on an integration of both objectives [Benini et al. 2000; Mittal 2014; Kong et al. 2012]. The trend toward the more elaborate management of aspects that are considered critical is apparent in research but also in industry, and we expect growing sophistication in the handling of individual concerns such as power consumption, over-heating, reliability, performance, and the like and a widening of scope to the simultaneous management of multiple, critical issues.

Most interesting from our perspective are those projects that maintain a more sophisticated, internal model about the system's state, work that often draws on control theory. For instance, Wang et al. [2009] propose a control algorithm based on an online

Table II. Smart Dynamic Power Management

	Adaptation Type		Cross-Layer Sensing and Monitoring					Decision Making Layer					Cross-Layer Actuation Level				
References	Simple*	Self-Aware†	Ckt	HW	NW	OS	App	Ckt	HW	NW	OS	App	Ckt	HW	NW	OS	App
[Kumar et al. 2003]	✓	–	–	✓	–	✓	–	–	–	–	✓	–	–	–	–	✓	–
[Wu et al. 2004]	✓	–	–	✓	–	–	–	–	–	–	✓	–	–	–	–	✓	–
[Wu et al. 2005]	✓	–	–	✓	–	–	✓	–	–	–	✓	✓	–	–	–	✓	✓
[Isci et al. 2006]	✓	–	✓	✓	–	–	–	–	✓	–	✓	–	–	✓	–	✓	–
[Nathuji and Schwan 2007]	✓	–	–	✓	–	✓	✓	–	✓	✓	✓	–	–	✓	✓	✓	–
[Curtis-Maury et al. 2008]	✓	–	✓	✓	–	✓	–	–	✓	–	✓	–	–	✓	–	✓	–
[Verma et al. 2008]	✓	–	–	✓	–	✓	–	–	✓	–	✓	–	–	✓	–	✓	–
[Sridharan et al. 2008]	✓	–	✓	✓	–	–	–	✓	✓	–	✓	–	✓	✓	–	✓	–
[Rangan et al. 2009]	✓	–	–	✓	–	✓	–	–	✓	–	✓	–	–	✓	–	✓	–
[Wang et al. 2009]	✓	✓	–	✓	–	✓	–	–	✓	–	✓	–	–	✓	–	✓	–
[Bartolini et al. 2010]	✓	–	✓	✓	–	?	–	✓	✓	–	–	–	✓	✓	–	–	–
[Hoffmann et al. 2011]	✓	✓	–	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓
[Bartolini et al. 2011]	✓	✓	–	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–
[Rotem et al. 2012]	✓	–	✓	✓	–	–	–	✓	✓	–	✓	–	✓	✓	–	✓	–
[Sun et al. 2013]	✓	–	✓	✓	–	–	–	✓	✓	–	✓	–	✓	✓	–	✓	–
[Shafique et al. 2013]	✓	–	✓	✓	–	–	–	–	✓	–	✓	–	–	✓	–	✓	–
[Shafique and Henkel 2013]	✓	–	✓	✓	✓	–	–	–	✓	✓	–	–	–	✓	✓	–	–

* **Implicit Model †Explicit Model, ? Not explicitly discussed; Ckt = Circuit, HW = Hardware, NW = Network, OS = Operating System, App = Application.**

model estimator to control accuracy and system stability. In a similar vein, Shafique et al. [2013] use implicit models to predict key features such as required resources for an approaching time interval. The models in these and many other examples are implicit and serve a narrow purpose. History-based prediction is a good example and commonly used. Based on a record of an application's past resource usage, the resource requirements for a future time interval are estimated. The past resource usage, perhaps only a single number, is considered a narrow model that represents a property of interest. Since almost all the approaches in Tables I and II focus on single issues with relatively simple objectives, they maintain narrow, implicit models of the systems themselves. The broader the objectives become and the more aspects that are integrated in the decision process, the richer the internal models grow. The power, thermal and reliability models used in the virtual platform described by Bartolini et al. [2010] are more detailed and elaborate, even though the models integrated in the final device, as part of the online feedback based control algorithm, are simplified and optimized. Often design-time information is not sufficient or accurate enough due to unforeseen influences or aging effects. To counter such limitations, online self-calibration and learning techniques are employed to improve the models used in the control algorithms [Bartolini et al. 2011]. Such needs require more detailed and explicit models to represent more of the system's features, thus gradually increasing their sophistication.

Starting from the other end, more systematic approaches toward self-awareness have been taken by the HAMSoC and SEEC projects.

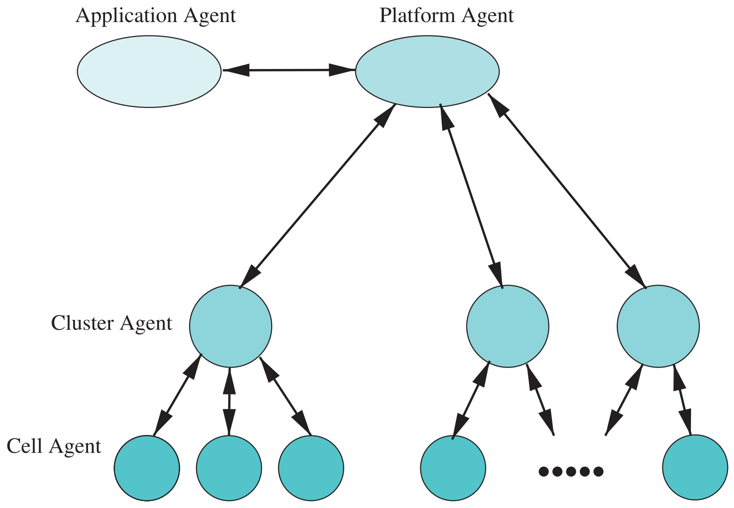

Fig. 2. Agent hierarchy in the HAMSoC system [Guang et al. 2010b].

HamSoC [Guang et al. 2010a, 2010b, 2011] is an SoC platform with a hierarchical agent structure as illustrated in Figure 2. The cell agents are hosted by individual processing cores in a multicore SoC. Clusters are formed along subsystem boundaries, and the platform agent is responsible for the entire chip. Although the agents at the platform layer and below are application independent, the application agent is customized toward the application needs. The agents perform a set of activities including *Communicate*, *Configure*, *Inquire*, *Order*, *Report*, and *Inform* [Guang et al. 2010b] with the objective of monitoring the system's state and performance, communicate with other agents across the hierarchy, and reconfigure the system to adapt it to a changed situation. The agents and their actions are defined in a generic and abstract way to form a framework suitable for a variety of applications and implementations. As an application case, a power management system for an NoC-based multicore SoC has been implemented and evaluated [Jafri et al. 2012; Guang et al. 2011]. The cell and cluster agents are realized in hardware whereas the agents at the platform and application layers are software programs. In the case study, performance and power attributes are monitored and controlled, but the framework is fairly general and would allow the monitoring of any interesting property, whereas the decision process could be assigned to the appropriate agent at the cluster, platform, or application layer.

SEEC [Hoffmann et al. 2010b] is a general framework for self-aware computing using an Observe-Decide-Act (ODA) paradigm. As illustrated in Figure 3, the system cyclically monitors key features, applies a control and decision algorithm, and deploys appropriate actions to adapt to changes in the environment and its own state. It is based on the heartbeats API library [Hoffmann et al. 2010a], which defines a cyclic event called a heartbeat. Through API functions, the application can register rate and latency performance goals in terms of the heartbeat period. Hence, the heartbeats API is a standardized means to monitor the performance of an application. The application itself or a separate agent can then adapt and optimize the system's behavior, for instance by allocating and scheduling resources appropriately. The approach has been further developed and evaluated in several applications for performance optimization [Hoffmann et al. 2010b], power management [Hoffmann et al. 2011, 2013], and managing of multiple objectives [Hoffmann 2014]. Also, the concept of *knobs* has been introduced [Hoffmann et al. 2011] to expose steering facilities such as processor

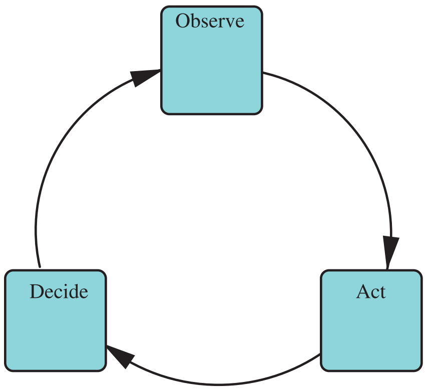

Fig. 3. The SEEC activity cycle [Hoffmann et al. 2010b].

speed or power modes. As a conceptual framework, it allows the user to adopt different decision-making strategies and algorithms, which has been explored and studied extensively [Santambrogio et al. 2010; Maggio et al. 2011].

As we observe the extensive work published in this area, we note that all approach the domain of self-awareness from different directions and angles. The "single-issue" approaches, as listed only incompletely in Tables I and II, introduce specific and concrete aspects of self-monitoring and self-adaptation to solve a particular well, but narrowly defined problem. Although the proposed techniques lead to effective solutions in the given scope, they do not easily generalize to a situation where a range of objectives has to be met simultaneously under a set of constraining conditions. In particular, various kinds of uncertainties and incomplete information constitute additional complications.

To overcome these limitations, a few general frameworks have been developed, such as HAMSoC and SEEC. Both propose a basic concept (hierarchical agent network in the case of HAMSoC and an ODA cycle with the heartbeat paradigm in the case of SEEC) and apply it to increasingly broad application scenarios while further developing and refining the frameworks. We note that these endeavors are complementary in terms of the insight they generate and the techniques they describe and study.

3. CLASSIFICATION AND TAXONOMY/LEVELS OF AWARENESS

In contrast, the classification we describe here starts from the other end and lists the attributes that we expect to see in a self-aware system. The various concepts reviewed are certainly multidimensional and have too many facets and aspects to easily press them into a simple scheme of classification. But if we concentrate for a while on awareness and self-awareness, which are the basis for many higher level cognitive abilities, we can identify different features that, although not arranged linearly, constitute a well-structured space that will allow us to better assess specific realizations in SESs. The framework of awareness that we use [Jantsch and Tammemäe 2014] requires several properties before we can say that the system is aware of something or self-aware:

—**Abstraction** of the primary input data into a semantic domain that is meaningful for the system at hand.
—**Disambiguation** of the possible interpretations to always settle on exactly one interpretation of the reality at any given time. When new data become available, the interpretation may change, but, at any given time, there is only one interpretation used by the system.

—**Semantic interpretation** is the result of abstraction and disambiguation, and it represents a relevant property of the system or its environment.
—**Desirability scale** provides a uniform goodness scale for the assessment of all observed properties.
—**Semantic attribution** maps properties into the desirability scale, suggesting how good or bad an observation is for the system.
—**History of a property** reflects the awareness of a property and implies awareness of its change over time. This history may be more or less detailed and may slowly fade as time passes, but it certainly is required to allow for the assessment of properties, the environment, and the system itself in a historical context.
—**Goals** provide the context in which interpretation and semantic attribution is meaningful.
—**The purpose** of an SES is to achieve all its goals.
—**Expectation on environment** implies that the system expects a specific environment, which is a precondition to realize if the environment is profoundly changing. The system's goals are often dependent on the environment.
—**Expectation on Subject** similarly implies that the system's own state and condition are continuously assessed to detect deviations, degradation, excellent performance, and malfunctions.
—**Inspection engines** that continuously monitor and assess the situation require a specific machinery that integrates all observations into a single, consistent world.

To realize all these properties in an SES is rather ambitious and not always necessary. Depending on which of these properties are present and to what degree, we can group systems into five levels:

—**Level 1, Adaptive:** A classic PID controller adapts to changes in the environment by following reference values. Such a system does some abstraction and has some expectations but in rather limited ways.
—**Level 2, Property Aware:** The system derives a *semantic interpretation* and attribution of monitored data. The system has *expectations* regarding the monitored property. It also has *goals,* and the attribution is done with respect to these goals. The more properties it follows in this way, the larger the share of the environment becomes that it is aware of. If the system monitors its own properties, we call it *self-aware*.
—**Level 3, History Aware:** If, in addition to properly interpreting and classifying properties, the system maintains a *history* of observations, the environmental changes over time are monitored and assessed. Moreover, a *history self-aware* system can monitor and assess its own performance and relate it to expectations and goals.
—**Level 4, Predictive:** The inspection mechanism that allows the system to observe and assess the environment and itself can be used to study future scenarios as support for decision-making. A system with the capability to simulate if-then-else scenarios is called predictive.
—**Level 5, Group Aware:** In addition to the self and the environment, the system recognizes a *peer group* with shared goals and/or similarity in behavior.

As in many other classifications, the details and boundaries can be debated, but this gives us a simple framework to categorize the work done in this field. A classification similar in ambition and scope has been proposed by Lewis et al. [2015] and Faniyi et al. [2014]. Inspired by Neisser's work in psychology [Neisser 1997], they distinguish between the five awareness levels *stimulus-aware*, *interaction-aware*, *time-aware*, *goal-aware,* and *meta-self-aware*. There is no simple mapping to our categorization, but all concepts found in one can also be identified in the other, although with different emphasis. Our property awareness is similar to Lewis et al.'s stimulus-awareness but makes the abstraction mechanism explicit. Similarly, our Level 1 of an adaptive system is related to interaction-awareness but with a focus on adaptivity while keeping the concept of interaction rather implicit. Our history awareness and Lewis et al.'s time-awareness resemble each other quite closely. Lewis et al.'s goal-awareness and meta-self-awareness separate aspects of inspection and reasoning about goals and the self, which are combined in our predictive level. Our Level 5 of group awareness is not covered in Lewis et al.'s set of levels, but they consider it as a distinct aspect under the label of *collectives and emergent self-aware systems* [Lewis et al. 2015].

This direct comparison of these two schemes of categorization highlights that there is no single scheme yet that structures the relevant concepts in an obviously more natural way than others. Depending on preference, emphasis, and objectives, one may choose and adapt a proper categorization. However, it is also reassuring since different schemes tend to cover the same ground and thus important aspects have most likely not been overlooked.

Reviewing the state of the art with our five-level scheme in mind, we observe that most embedded and CPSs proposed do some abstraction and interpretation of individual properties such as power consumption, performance, and occurrence of specific faults. Regarding these properties, there are also, mostly implicitly, defined goals and expectations. With the exception of the work done by Preden and Helander [2006] and the heartbeat framework [Hoffmann et al. 2010a], history records of properties are not kept or used in any systematic way. The systems described by Sanz et al. [2007] and, to a more limited extent, by Kaindl et al. [2013] use fairly sophisticated inspection engines and modeling capabilities. The work by Sánchez-Escribano and Sanz [2014] is an interesting attempt toward what we have called a *unified desirability scale* and a semantic attribution. A kind of desirability scale is found whenever several objectives are targeted simultaneously [Hoffmann 2014; Wang and Wang 2011; Sylvester et al. 2006]. However, it appears implicitly as part of an objective function that is itself often not maintained explicitly, meaning it cannot be generalized. We expect from a self-aware system that perhaps tens of partially independent observed properties are easily related to each other with respect to an equally large set of desired goals. Similarly, purpose and goals are either implicitly hidden in some decision algorithm, hard-coded at design time, or both. In order to meet a larger set of more or less independent goals, which are also likely to change over time, a more systematic and explicit representation of objectives has to be developed. An interesting step is found in the heartbeats framework [Hoffmann et al. 2010a], which allows applications to register performance goals that are then monitored by the framework. It would be interesting to explore if this approach can be generalized to functional objectives and made sufficiently flexible to allow dynamic formulation of new goals. Along the same lines, we note that expectations on the environment are handled in a similarly ad-hoc and implicit way, if at all.

The work by Preden [2014, 2012] and Motus et al. [2009] is to our knowledge the most advanced attempt toward *group awareness,* although it is still limited in this respect. However, the recent strong attention on *fog computing* [Jennings and Stadler 2014; Satoh 2013; Hong et al. 2013; Bonomi et al. 2012] may speedily advance this field and contribute to an understanding of group awareness in cooperating systems, how it emerges, and what it is good for.

In summary, apart from classic control systems that populate Level 1, the overwhelming majority of the work discussed here appears in Level 2, although researchers realize property awareness to grossly different degrees. There are some isolated attempts to address the core features of Levels 3, 4, and 5. No level above Level 1 is covered satisfactorily, hence, significant research challenges remain. We try to address some of these challenges at different levels using a new computing paradigm in Section 4.

4. CYBERPHYSICAL-SYSTEM-ON-CHIP (CPSOC): A PARADIGM AND ARCHITECTURE FOR SMART EMBEDDED SYSTEMS

From an SoCperspective, SESs are an emerging area of computing system with unique architectural attributes. SoCs as SES have many similarities with autonomic computing systems [Kephart et al. 2003] but are severely resource- and capability-constrained. They can be analyzed through the computing-communication-control (C3)-centric notions of CPSs [Lee 2008] but are limited due to the lack of explicit notions of the operating systems and compilation principles in C3. Furthermore, emerging embedded computing platforms that deploy complex SOCs will be characterized by the following key features that provide both challenges and opportunities for simultaneously managing system resilience, energy, and adaptivity:

—They will see much larger fault rates. More integration results in larger platforms facing more dominant failure mechanisms (with technology scaling), causing increased fault rates [Bernstein et al. 2006]. This is especially true for memories in emerging data-centric platforms [Nassif et al. 2010; Singhee and Rutenbar 2010].
—They will be monitor-rich. To assess the state of health of the system, these computing systems will employ a network of interconnected monitors looking for signatures of faults, wearout, and impending failures [Floyd et al. 2007; Kornaros and Pnevmatikatos 2013b; Lefurgy et al. 2013]. These monitors will span circuit, microarchitecture, and software layers.
—They will be aggressively heterogeneous in the computing fabric, covering all dimensions: processing (for accelerating application/domain-specific functions) [Borkar and Chien 2011], interconnect (to handle scalability and high-throughput) [Shafique et al. 2014], and memory (combining volatile and nonvolatile storage; e.g., Dhiman et al. [2009]).
—They will be memory-heavy and will deploy heterogeneous memory technologies. The data-centric nature of several emerging applications creates demand for denser memories. Memories are likely to dominate energy as well as reliability concerns [Nassif et al. 2010; Singhee and Rutenbar 2010] for computing systems. Moreover, technology trends such as 3D integration [Borkar 2011] and heterogeneous memory organizations (e.g., combining traditional SRAMs with emerging faster, denser, nonvolatile memories) [Wu et al. 2009] pose new challenges for energy efficiency and resilience.

These concerns highlight the need for SES architectures that dynamically balance multiple objectives across multiple levels of the design abstraction stack, manage their limited resources, and are always keenly aware of their own accomplishments and shortcomings. These abilities and attributes distinguish them from traditional embedded systems design and motivate the need for a new design paradigm specifically suitable for SES, as proposed in some recent works [Sarma et al. 2013, 2015]. We briefly discuss such a paradigm and highlight its suitability for SES.

4.1. Overview

CPSoC [Sarma et al. 2013, 2015] is an SES paradigm that combines a sensor-actuator-rich C3-centric paradigm with that of an adaptive and reflective middleware (a flexible hardware-software stack and interface between the application and OS layer) to control the manifestations of computations (e.g., aging, overheating, parameter variability, etc.) on the physical characteristics of the chip itself and the outside interacting environment. Inspired by the C3 paradigm of CPSs [Lee 2008] and the adaptive and learning abilities of autonomous computing [Kephart et al. 2003], CPSoC provides a computing framework that assures the dependability of cyber/information processing (i.e., the cyber aspects such as integrity, correctness, accuracy, timing, reliability, and security) while simultaneously addressing the physical manifestations (in performance, power, thermal, aging, wear-out, material degradation, reliability, and dependability) of information processing on the underlying computing platform. Not unlike the reference architecture proposed by Lewis et al. [2015], CPSoC aims to coalesce these two traditionally disjoint aspects/abstractions of the cyber/information world and the underlying physical computing worlds into a unified abstraction of computing by using cross-layer virtual/physical sensing and actuation to enable a C3-centric self-aware computing platform.

The CPSoC architecture consists of a sensor-actuator-rich computation platform supported by adaptive NoCs (cNoC, sNoC), Introspective Sentient Units (ISU), and an adaptive and reflective middleware to manage and control both the cyber/information and physical environment and characteristics of the chip [Sarma et al. 2013, 2015]. The CPSoC architecture is broadly divided into several layers of abstraction, for example, applications, operating system, network and bus communication, hardware, and the circuit/device layers. CPSoC inherits most features of MPSoC in addition to on-chip sensing and actuation to enable the ODA paradigm. Unlike traditional MPSoC, each layer of the CPSoC can be made self-aware and adaptive by a combination of software and physical sensors and actuators, as shown in Figure 4(a). These layer-specific feedback loops are integrated into a flexible stack that can be implemented either as firmware or middleware, as shown by the dotted line in Figure 4(a).

CPSoC distinctly differs from a traditional MPSoC in several ways. Traditional MPSoC paradigms lack the ability to sense the system states and behaviors across layers of system stacks due to lack of architectural support; they are incapable of exploiting and exposing process and workload variations due to a lack of suitable abstractions at multiple layers. Furthermore, they sacrifice usable performance and energy opportunities by adopting worst-case design (guard bands), and they lack support for multilevel actuation mechanisms and adaptations to aggressively meet competing and conflicting demands. Moreover, traditional MPSoCs lack self-learning mechanisms that can anticipate failures and predict vulnerabilities. CPSoC overcomes these limitations, as detailed later.

4.2. CPSoC Features

The CPSoC framework supports four key ideas: (i) physical and virtual sensing and actuation, (ii) simple/self-aware adaptations, (iii) multi- or cross-layer interactions and interventions, and (iv) predictive modeling and learning. We briefly describe these herein. (A detailed description is found in our Technical Report [Sarma et al. 2013].)

4.2.1. Cross-Layer Virtual and Physical Sensing and Actuation. CPSoCs are sensor-actuator-rich MPSoCs that include several on-chip physical sensors (e.g., aging, oxide breakdown, leakage, reliability, temperature, and performance counters, as well as voltage, current, and power sensors [Sarma et al. 2013, 2015]) on the lower three layers, as shown by the On-Chip-Sensing-and-Actuation (OCSN) block in Figure 4(b). On the other hand, virtual sensing is a physical-sensorless sensing of immeasurable parameters using indirect computation [Sarma et al. 2012]. It can be viewed as a software sensor that provides indirect measurement of abstract conditions, contexts, inferences, or estimates by processing (e.g., combining, aggregating, or predicting) sensed data from either a set of homogeneous or heterogeneous sensors. It is also a computational technique that enhances and/or adds sensing capability, introduces sensing options,

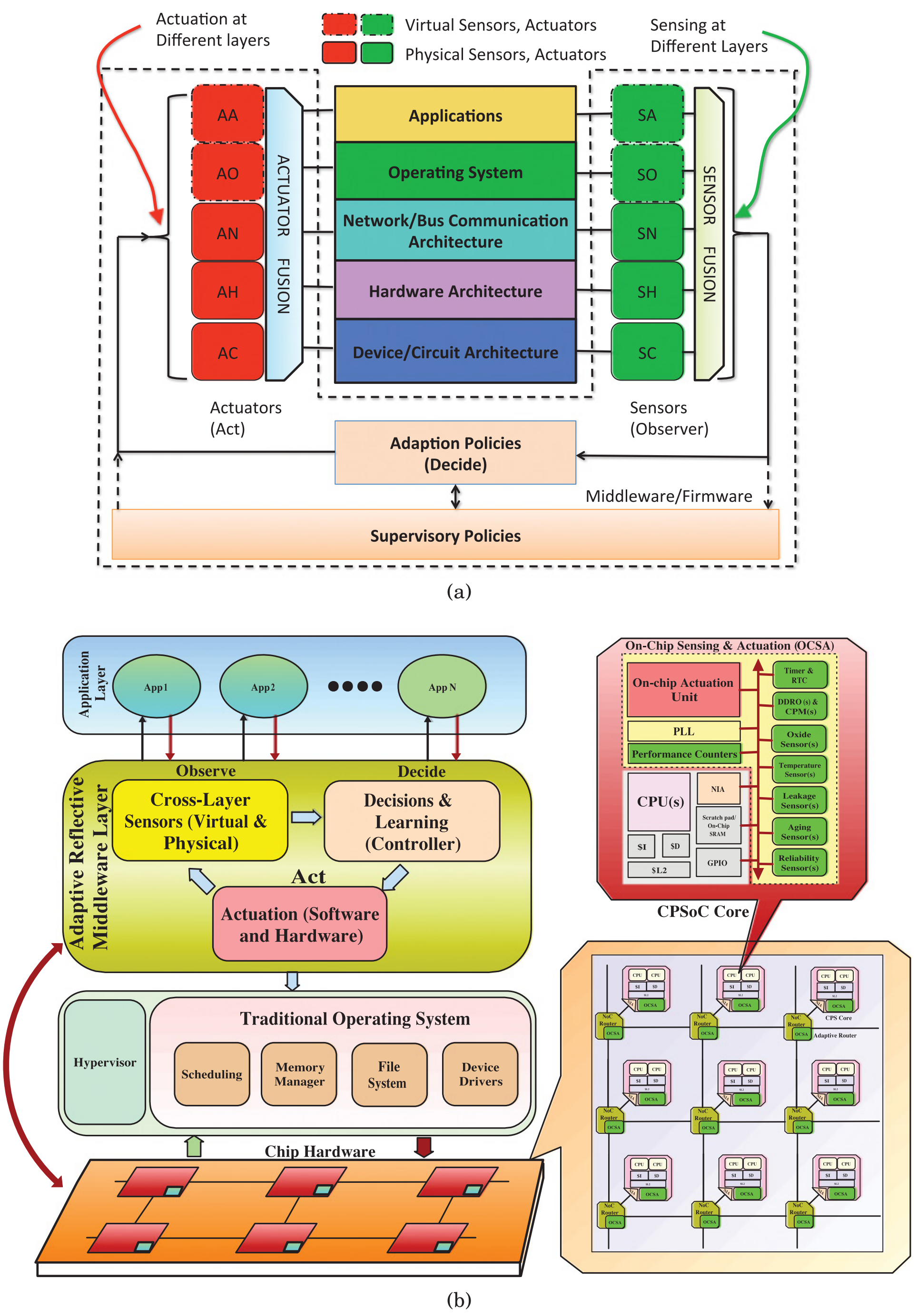

Fig. 4. (a) Cross-layer virtual sensing and actuation at different layers of CPSoC. (b) CPSoC architecture with adaptive Core, NoC, and the Observe-Decide-Act Loop as Adaptive, Reflexive Middleware [Sarma et al. 2013, 2015].

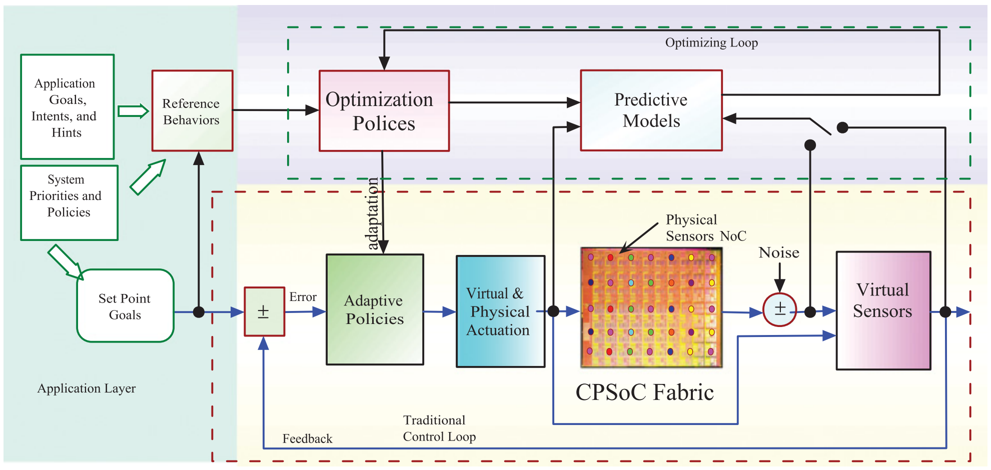

Fig. 5. Adaptation using predictive control model and policies in CPSoC [Sarma et al. 2014, 2015].

increases sensitivity, enables efficient sensor resource uses, and overcomes physical placement and cost restrictions. When combined with different kinds of sensors, virtual sensing enables consensus to resolve faults and errors while providing a test bedfor on-chip sensor fusion [S. Sarma et al. 2014]. The need for such an overprovisioned sensing architecture [Sarma and Dutt 2014a] for MPSoCs has also been identified by Intel [Borkar 2013].

Similarly, we define virtual actuations [Sarma et al. 2013, 2015] (e.g., application duty cycling, algorithmic choice, checkpointing) that are software/hardware interventions that can predictively influence system design objectives such as performance, power, and reliability. Virtual actuation can be combined with the physical actuation mechanisms commonly adopted in modern chips (e.g., DVFS and Adaptive Body Biasing [ABB] to control chip performance, power, and parametric variations); the notion of actuator fusion in CPSoC represents virtual and physical actuations that are combined across different layers of abstraction [Sarma et al. 2013, 2015].

4.2.2. Simple and Self-Aware Adaptations. Self-awareness is used to describe the ability of the CPSoC to observe its own internal behaviors as well as external systems it interacts with such that it is capable of making judicious decisions that optimize performance and other Quality of Service (QoS) metrics [Kephart et al. 2003]. Self-aware systems will be capable of adapting their behavior and resources to automatically find the best way to accomplish a given goal despite changing environmental conditions and demands. A self-aware system must be able to monitor its behavior to update one or more of its components (hardware architecture, operating system, and running applications) to achieve its goals.

Two key attributes of the self-aware CPSoC are adaptation of each layer and multiple cooperative ODA loops. As an example, the unification of an adaptive computing platform (with combined DVFS, ABB, and other actuation means) along with a bandwidth adaptive NoC [Sarma et al. 2013, 2015] offers extra dimensions of control and solutions in comparison to traditional MPSoC architecture. These cooperative and hierarchical control loops (e.g., the combination of traditional control loop; dotted lower box in Figure 5) together with virtual sensing enabled optimized loop (upper loop in Figure 5) effectively translate user goals or QoS into one or more multiple design objectives [Sarma et al. 2013, 2015].

4.2.3. Online Learning. Predictive modeling and online learning abilities of the system behavior, as well as internal and external (environmental) states, provide self-modeling abilities in the CPSoC paradigm. The system behavior and states can be built using on- or offline linear or nonlinear models in time or frequency domains [Ljung 1998]. We specifically use statistical and neural network approaches [Haykin et al. 2009; Fausett 1994] such that the model accuracy can be traded-off for model computational complexity. We use regression-based linear predictors and nonlinear neural predictors to build models of the system's performance and power and energy consumption using the cross-layer events, hardware counters, and on-chip sensor data. In addition, use of coupling parameters (a metric that quantifies the interactions between layers) helps to develop application and cross-layer interaction models for nominal and abnormal operations. We use the predictive and learning abilities of CPSoC to improve autonomy in managing the system resources and assisting proactive resource utilization in the runtime system [Sarma et al. 2013, 2015].

4.2.4. Multi- or Cross-Layer Interactions and Interventions. On-chip self-awareness with cross-layer virtual and physical sensing and actuation is a key enabling technology for efficient use of heterogeneous architectures and applications with guaranteed runtime system goals and QoS (performance, reliability, power, thermal behavior) in a highly dynamic environment. Our previous work demonstrated the use of multi- and cross-layer interactions and interventions for managing multiple design constraints (e.g., power, performance, thermal, resilience, aging), as well as in different design contexts (e.g., mobile platforms, data-intensive applications, long-mission applications, etc.); our Technical Report [Sarma et al. 2013] details several sample applications where self-awareness is used to improve energy efficiency, increase system lifetime by reducing aging effects, and improve system performance under thermal constraints. In the next section, we present sample instances of use cases and applications of CPSoC.

4.3. CPSoC Application in Smart Embedded Systems Design

On-chip self-awareness with cross-layer virtual and physical sensing and actuation is a key enabling technology for efficient use of heterogeneous architectures and applications with guaranteed runtime system goals and QoS (performance, reliability, power, thermal behavior) in a highly dynamic environment. Our Technical Report [Sarma et al. 2013, 2015] contains several sample applications where self-awareness is used to improve energy efficiency, increase system lifetime by reducing aging effects, and improve system performance under thermal constraints. We highlight a few of these application opportunities in the following subsections.

4.3.1. Smart Power-Reliability Online Co-Management. Reliability and power modeling for emerging systems should be cross-layer and consider the full system stack [Quinn et al. 2011; Mitra et al. 2010, 2011; Carter et al. 2010] for holistic system-level optimization and to avoid the pessimistic and pathological assumptions of a single-layer approach. As emerging Heterogeneous MultiProcessor Platforms (HMPs) (Figure 4(b)) are moving toward aggressively heterogeneous architectures distinct in the number and type of compute cores, the cross-layer and predictive modeling capability in CPSoC can be used to achieve adaptive policies in the ODA middleware layer in Figure 4(b). These emerging platforms face diverse multithreaded workloads, requiring a complex OS scheduler and load balancer to fully exploit the platform's heterogeneity for managing power-performance and system resilience. Existing OS kernels' scheduling and load balancing schemes (Figure 6(a)) are openly accepted as inefficient for such systems [Grey 2013]. For instance, the vanilla Linux kernel load balancer evenly distributes

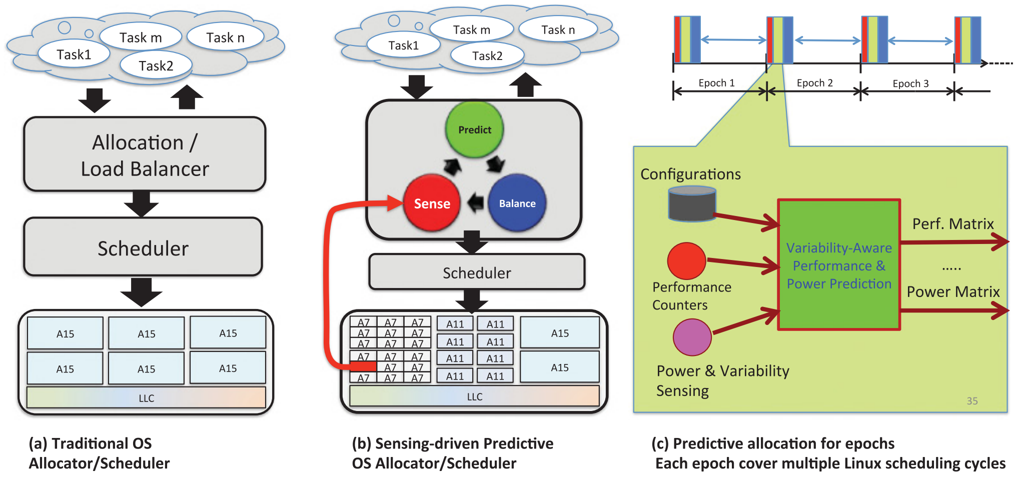
Task1
Task m
Task n
Task2
Allocation / Load Balancer
Scheduler
A15
LLC
(a) Traditional OS Allocator/Scheduler
Predict
Sense
Balance
Scheduler
A7
A11
A15
LLC
(b) Sensing-driven Predictive OS Allocator/Scheduler
Epoch 1
Epoch 2
Epoch 3
Configurations
Performance Counters
Power & Variability Sensing
Variability-Aware Performance & Power Prediction
Perf. Matrix
.....
Power Matrix
(c) Predictive allocation for epochs
Each epoch cover multiple Linux scheduling cycles

Fig. 6. Runtime system for emerging CPSoC/HMP platforms [Sarma et al. 2015].

the workload among cores even if the cores have distinct processing capabilities, resulting in serious performance and energy efficiency loss. Recent efforts to address this important issue (e.g., the IKS [Mathieu 2013] and the GTS [ARM Inc 2013] Linux extensions) are limited to the very specific case of ARM's big.LITTLE with two core types. Smart and self-aware capabilities specifically help in achieving versatile operating system support for emerging, aggressively heterogeneous platforms like CPSoC/HMP in addition to jointly addressing the resiliency and energy efficiency issues of such architectures.

In an initial investigation, we developed SmartBalance [Sarma et al. 2015], a smart Linux load balancer that is a closed-loop sensing-driven opportunistic load balancer that uses on-chip sensing, estimation, and prediction, as well as global optimization for aggressive HMPs. As shown in Figure 6(b) and (c), SmartBalance consists of three phases—sense, predict, and balance—executed at runtime in periodic epochs, where each epoch covers multiple Linux scheduling periods. Unlike the open-loop standard Linux load balancer, which distributes the threads evenly, this closed-loop feedback-driven approach makes judicious decisions to distribute the threads smartly (i.e., matched to the core type) to best achieve the system goal(s) (e.g., energy efficiency); it uses predictive models for the performance and power impact of each thread executing on different heterogeneous cores without the overhead of sampling at each core type.

Experiments executed with a synthetic mix of PARSEC benchmarks (representing a dynamically varying workload) on both the standard vanilla Linux as well as the SmartBalance-enhanced allocator, with the goal of achieving energy efficiency (IPS/watt), demonstrate more than 50% energy-efficiency improvement over the standard vanilla Linux kernel load balancer and 20% improvement over the state-of-the-art ARM GTS scheme on a quad-core heterogeneous MPSoC [Sarma et al. 2015]. Although these results look promising for energy efficiency, opportunities exist to extend this approach to develop policies for a resilience-power-aware load balancer for such emerging architectures.

4.3.2. Resiliency-Aware Smart Allocation for Improving System Reliability. Task allocation for emerging platforms has significant impact in heterogeneous resource and resilience management. Although task allocation for homogeneous architectures has been studied extensively for performance and power management, the implications of cross-layer heterogeneity on reliability during runtime allocation is an open problem, especially for emerging HMP platforms. The notion of a lifetime reliability characterization at the system level for different failure mechanisms using built-in reliability sensors in the CPSoC paradigm can make a runtime allocator reliability-aware. Although life-time reliability characterization has been used at the architectural level, elevating it to the OS and higher abstraction layers requires solving key abstraction and awareness challenges for such SESs. The lifetime reliability characterization matrix must incorporate a number of awareness properties in similar ways, typically as performance and power characterization matrices [Sarma et al. 2015] at the OS layer. This capability can enable lifetime reliability at the OS layer and use these for the intelligent allocation or balancing of threads to improve system reliability for emerging heterogeneous platforms.

4.3.3. Resiliency-Aware Scheduling for Power Management. Resiliency-awareness through the metric of lifetime reliability captures the impact of several failure mechanisms that eventually causes permanent faults. However, the lifetime reliability metric does not capture transient and intermittent reliability issues (e.g., Single Event Upset [SEU] and Single Event Transients [SET]). A metric for transient and intermittent faults at the OS level to complement the lifetime reliability metric can improve the awareness capability of existing systems substantially by impacting the effect of transient errors while sustaining the required lifetime of emerging platforms. A resiliency-aware intelligent scheduling scheme that can directly manage system power at the system level can lead to a synergistic scheduler capable of coordinating between the scheduler and the power management governors.

5. CHALLENGES AND RESEARCH DIRECTIONS

Embedded systems lie at the heart of computing in its many forms, from large CPSoSs, to the ubiquitous Internet-of-Things. The design and implementation of these embedded systems face tremendous challenges for correct operation in the face of highly dynamic environments, as well as in managing disparate and often cross-purpose constraints. The notion of "smartness" borrowed from biological systems is touted as a possible solution to deal with these challenges, but, unfortunately, there is little common understanding of the inherent properties underlying smartness. This article first reviewed commonly used "self-x" terms (e.g., self-awareness and self-adaptivity) and presented a taxonomy to define and structure terminology related to smartness, then used this taxonomy to position a sampling of the large body of related work in smart software and hardware systems. We then focused on a SoC perspective for SESs and presented the CPSoC platform as an exemplar for a hardware/software platform that exhibits the properties of self-monitoring and self-awareness to enable adaptation as well as learning mechanisms to evolve the system over time.

This article has barely scratched the surface of a wide range of challenges and opportunities facing the designers of next-generation SESs. In trying to raise the level of "smartness," there are significant challenges in embedding new models of cognition and intelligence emerging from neuroscience and psychology, all within the typical constraints of an embedded system (e.g., cost, power/energy, thermal, reliability, weight, etc.). Lightweight abstractions of these neurobiological learning concepts need to be developed, not only to fit into the multidimensional constraint envelope, but also with a view to making such smart techniques easily implementable (e.g., design or programming), especially given the shrinking design cycles for modern embedded systems. Furthermore, there is a whole slew of traditional and emerging challenges when smartness is embedded into these systems; some of them are novel design challenges not experienced in traditional algorithm-design methodologies but are encountered in the context of evolvable hardware [Yao and Higuchi 1999]:

—How do we express "correctness" when the smart system compensates for smaller and bigger misbehavior anyway?
—How do we validate a smartly adapting system?
—Shall we replace conventional specify-design-validate methodologies by a provide-smartness-and-set-objectives paradigm?
—How can we perform tradeoff analysis for smartness features?
—How can we quantify uncertainty, dynamicity, and variability in the platform, the environment, and the applications?
—How do we develop efficient learning algorithms to support smart, resource-constrained embedded systems?

These are but a few of the many challenges and opportunities facing designers of next-generation SESs, and they call for a principled approach toward a design science for smart software and hardware systems.

ACKNOWLEDGMENTS

We would like to thank (in alphabetical order) Luca Benini, Hank Hoffmann, Joerg Henkel, Dionisios N. Pnevmatikatos, Xin Yao for their useful comments and feedback in improving the manuscript.

REFERENCES

ARM Inc. 2013. big.LITTLE technology: The future of mobile. Retrieved from http://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf.

Todd Austin, Valeria Bertacco, Scott Mahlke, and Yu Cao. 2008. Reliable systems on unreliable fabrics. *IEEE Design & Test of Computers* 25, 4 (2008), 322–332.

B. J. Baars. 1989. *A Cognitive Theory of Consciousness*. Cambridge University Press.

B. J. Baars. 2002. The conscious access hypothesis: Origins and recent evidence. *Trends in Cognitive Science* 6, 1 (2002), 47–52.

Bernard J. Baars and Stan Franklin. 2009. Consciousness is computational: The LIDA model of global workspace theory. *International Journal of Machine Consciousness, World Scientific Publishing Company* (2009).

M. Bakhouya. 2011. A bio-inspired architecture for autonomic network-on-chip. In *Autonomic Networking-on-Chip Bio-Inspired Specification, Development, and Verification*, Phan Cong-Vinh (Ed.). CRC Press, 1–20.

M. Bakhouya and J. Gaber. 2014. Bio-inspired approaches for engineering adaptive systems. *Procedia Computer Science* 32 (2014), 862–869. DOI:http://dx.doi.org/10.1016/j.procs.2014.05.503 The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. 2013. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture* 8, 3 (2013), 1–154.

Luiz André Barroso and Urs Hölzle. 2007. The case for energy-proportional computing. *IEEE Computer* 40, 12 (2007), 33–37.

Andrea Bartolini, Matteo Cacciari, Andrea Tilli, and Luca Benini. 2011. A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 1–6.

Andrea Bartolini, Matteo Cacciari, Andrea Tilli, Luca Benini, and Matthias Gries. 2010. A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores. In *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI*. ACM, 311–316.

Robert C. Baumann. 2005. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability* 5, 3 (2005), 305–316.

L. Benini, A. Bogliolo, and G. De Micheli. 2000. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8, 3 (June 2000), 299–316. DOI:http://dx.doi.org/10.1109/92.845896

Joseph B. Bernstein, Moshe Gurfinkel, Xiaojun Li, Jörg Walters, Yoram Shapira, and Michael Talmor. 2006. Electronic circuit reliability modeling. *Microelectronics Reliability* 46, 12 (2006), 1957–1979.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the ast Edition of the MCC Workshop on Mobile Cloud Computing (MCC'12)*. ACM, New York, NY, 13–16. DOI:http://dx.doi.org/10.1145/2342509.2342513

Shekhar Borkar. 2011. 3D integration for energy efficient system design. In *Proceedings of the 48th Design Automation Conference*. ACM, 214–219.

Shekhar Borkar. 2013. Achieving energy efficiency by HW/SW co-design. In *Proceedings of the 3rd Berkeley Symposium on Energy Efficient Electronic Systems*. https://www.youtube.com/watch?v=ZKVObiEjANE.

Shekhar Borkar and Andrew A. Chien. 2011. The future of microprocessors. *Communications of the ACM* 54, 5 (2011), 67–77.

Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. 2003. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th Annual Design Automation Conference (DAC'03)*. ACM, New York, NY, 338–342. DOI:http://dx.doi.org/10.1145/775832.775920

David Brooks and Margaret Martonosi. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA 2001)*. IEEE, 171–182.

F. Cancare, S. Bhandari, D. B. Bartolini, M. Carminati, and M. D. Santambrogio. 2011. A bird's eye view of FPGA-based evolvable hardware. In *Proceedings of the 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 169–175. DOI:http://dx.doi.org/10.1109/AHS.2011.5963932

Nicholas P. Carter, Helia Naeimi, and Donald S. Gardner. 2010. Design techniques for cross-layer resilience. In *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 1023–1028.

Tao Chen, Funmilade Faniyi, Rami Bahsoon, Peter R. Lewis, Xin Yao, Leandro L. Minku, and Lukas Esterle. 2014. The handbook of engineering self-aware and self-expressive systems. *Computing Research Repository (CoRR)* abs/1409.1793 (2014). http://arxiv.org/abs/1409.1793.

Betty H. C. Cheng and others. 2009b. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, Betty HC Cheng, Rogério de Lemos, Paola Inverardi, and Jeff Magee (Eds.). Springer.

Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, and others. 2009a. *Software Engineering for Self-adaptive Systems: A Research Roadmap*. Springer.

A. Clark. 2001. *Mindware: An Introduction to the Philosophy of Cognitive Science*. Oxford University Press, New York.

Phan Cong-Vinh (Ed.). 2011. *Autonomic Networking-on-Chip: Bio-Inspired Specification, Development, and Verification*. CRC Press. http://www.crcnetbase.com/isbn/9781439829134.

Ayse Kivilcim Coskun, Tajana Simunic Rosing, and Kenny C. Gross. 2008. Temperature management in multiprocessor SoCs using online learning. In *Proceedings of the 45th ACM/IEEE Design Automation Conference (DAC 2008)*. IEEE, 890–893.

Matthew Curtis-Maury, Filip Blagojevic, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. 2008. Prediction-based power-performance adaptation of multithreaded scientific codes. *IEEE Transactions on Parallel and Distributed Systems* 19, 10 (2008), 1396–1410.

Shidhartha Das, Carlos Tokunaga, Sanjay Pant, Wei-Hsiang Ma, Sudherssen Kalaiselvan, Kevin Lai, David M. Bull, and David T . Blaauw. 2009. RazorII: In situ error detection and correction for PVT and SER tolerance. *IEEE Journal of Solid-State Circuits* 44, 1 (2009), 32–48.

Gaurav Dhiman, Raid Ayoub, and Tajana Rosing. 2009. PDRAM: A hybrid PRAM and DRAM main memory system. In *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC'09)*. IEEE, 664–669.

E. W. Dijkstra. 1974. Self-stabilizing systems in spite of distributed control. *Communications of the ACM* 17, 11 (1974), 643–644.

Thomas Ebi, M. Faruque, and Jörg Henkel. 2009. Tape: Thermal-aware agent-based power econom multi/many-core architectures. In *Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers (ICCAD 2009)*. IEEE, 302–309.

Mica R. Endsley. 1988. Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors and Ergonomics Society 32th Annual Meeting*. 97–101. DOI:http://dx.doi.org/10.1177/154193128803200221

Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA 2011)*. IEEE, 365–376.

P. J. Montestruque, L. A. McMickell, M. B. Lemmon, M. Yashan, Sun Hui, Fang Koutroulis, I. Haenggi, M. Min, Xie Xiaojuan, Xie Fang, and Lei Antsaklis. 2005. Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience. *IEEE Transactions on Instrumentation and Measurement* 54, 6 (Nov. 2005 2005), 2342–2358. DOI:http://dx.doi.org/10.1109/TIM.2005.858557

F. Faniyi, P. R. Lewis, R. Bahsoon, and X. Yao. 2014. Architecting self-aware software systems. In *Proceedings of the 2014 IEEE/IFIP Conference on Software Architecture (WICSA)*. 91–94.

Laurene V. Fausett. 1994. *Fundamentals of Neural Networks*. Prentice-Hall.

Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. *ACM SIGARCH Computer Architecture News* 40, 1 (2012), 37–48.

Michael S. Floyd, Soraya Ghiasi, Tom W. Keller, Karthick Rajamani, F. L. Rawson, Juan C. Rubio, and Malcolm S. Ware. 2007. System power management support in the IBM POWER6 microprocessor. *IBM Journal of Research and Development* 51, 6 (2007), 733–746.

Debanjan Ghosh, Raj Sharman, H. Raghav Rao, and Shambhu Upadhyaya. 2007. Self-healing systems - Survey and synthesis. *Decision Support Systems* 42, 4 (January 2007), 2164–2185. DOI:http://dx.doi.org/10.1016/j.dss.2006.06.011

G. Grey. 2013. big.LITTLE software update. Retrieved from http://www.linaro.org/blog/hardware-update/big-little-software-update/, 2013.

Liang Guang, Ethiopia Nigussie, Pekka Rantala, Jouni Isoaho, and Hannu Tenhunen. 2010a. Hierarchical agent monitoring design approach towards self-aware parallel systems-on-chip. *ACM Transactions on Embedded Computer Systems* 9, 3 (2010), 1–24. DOI:http://dx.doi.org/10.1145/1698772.1698783

L. Guang, G. Plosila, J. Isoaho, and H. Tenhunen. 2011. HAMSoC: A monitoring-centric design approach for adaptive parallel computing. In *Autonomic Networking-on-Chip: Bio-Inspired Specification, Development, and Verification*, Phan Cong-Vinh (Ed.). CRC Press, 135–164.

Liang Guang, Juha Plosila, Jouni Isoaho, and Hannu Tenhunen. 2010b. Hierarchical agent monitored parallel on-chip system: A novel design paradigm and its formal specification. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)* 1, 2 (2010).

Simon S. Haykin, Simon S. Haykin, Simon S. Haykin, and Simon S. Haykin. 2009. *Neural Networks and Learning Machines*. Vol. 3. Pearson Education, Upper Saddle River, NJ.

D. Sufen Fong, Aghajan H. Hengstler, and S. Prashanth. 2007. MeshEye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proceedings of the 6th Internatinoal Symposium on Information Processing in Sensor Networks (IPSN 2007)*. Stanford University, Stanford, CA, 360–369. DOI:http://dx.doi.org/10.1109/IPSN.2007.4379696

J. Henkel, L. Bauer, J. Becker, O. Bringmann, U. Brinkschulte, S. Chakraborty, M. Engel, R. Ernst, H. Hartig, L. Hedrich, A. Herkersdorf, R. Kapitza, D. Lohmann, P. Marwedel, M. Platzner, W. Rosenstiel, U. Schlichtmann, O. Spinczyk, M. Tahoori, J. Teich, N. When, and H. Wunderlich. 2011. Design and architectures for dependable embedded systems. In *Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2011)*. 69–78.

J. Henkel, A. Herkersdorf, L. Bauer, T. Wild, M. Hubner, R. K. Pujari, A. Grudnitsky, J. Heisswolf, A. Zaib, B. Vogel, V. Lari, and S. Kobbe. 2012. Invasive manycore architectures. In *Proceedings of the 17th Asia and South Pacific Design Automation Conference (ASP-DAC 2012)* 193–200. DOI:http://dx.doi.org/10.1109/ASPDAC.2012.6164944

T. Higuchi, Y. Liu, and X. Yao (Eds.). 2006. *Evolvable Hardware*. Springer Science+Media LLC, New York.

Eric Hoffman, Peter Martin, Thomas Pütz, Aymeric Trzmiel, and Karim Zeghal. 2007. Airborne spacing: Flight deck view of compatibility with continuous descent approach (CDA). *Interface (September)* (2007), 1–12.

H. Hoffmann. 2014. CoAdapt: Predictable behavior for accuracy-aware applications running on power-aware systems. In *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS 2014)*. 223–232. DOI:http://dx.doi.org/10.1109/ECRTS.2014.32

Henry Hoffmann, Jonathan Eastep, Marco D. Santambrogio, Jason E. Miller, and Anant Agarwal. 2010a. Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments. In *Proceedings of the 7th International Conference on Autonomic Computing*. ACM, 79–88.

H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. 2013. A generalized software framework for accurate and efficient management of performance goals. In *Proceedings of the 2013 International Conference on Embedded Software (EMSOFT)*. 1–10. DOI:http://dx.doi.org/10.1109/EMSOFT.2013.6658597

Henry Hoffmann, Martina Maggio, Marco D. Santambrogio, Alberto Leva, and Anant Agarwal. 2010b. Seec: A framework for self-aware computing. (2010).

Henry Hoffmann, Stelios Sidiroglou, Michael Carbin, Sasa Misailovic, Anant Agarwal, and Martin Rinard. 2011. Dynamic knobs for responsive power-aware computing. In *ACM SIGPLAN Notices*, Vol. 46. ACM, 199–212.

Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC'13)*. ACM, New York, NY, 15–20. DOI:http://dx.doi.org/10.1145/2491266.2491270

Toshiyuki INAGAKI. 2005. Design of human interactions with smart machines: Lessons learned from aircraft accidents. In *The 4th IARP/IEEE RAS/EURON, Keynote Lecture,* June 17, 2005, Nagoya (2005).

Canturk Isci, Gilberto Contreras, and Margaret Martonosi. 2006. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 359–370.

Syed M. A. H. Jafri, Liang Guang, Axel Jantsch, Kolin Paul, Ahmed Hemani, and Hannu Tenhunen. 2012. Self-adaptive NoC power management with dual-level agents: Architecture and implementation. In *Proceedings of the Conference on Self-adaptive Networked Embedded Systems*. Rome, Italy. http://web.it.kth.se/~axel/papers/2012/SANES-SyedJafri.pdf.

Axel Jantsch and Kalle Tammemäe. 2014. A framework of awareness for artificial subjects. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES'14)*. ACM, New York, NY, Article 20, 3 pages. DOI:http://dx.doi.org/10.1145/2656075.2661644

Brendan Jennings and Rolf Stadler. 2014. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management* (2014), 1–53. DOI:http://dx.doi.org/10.1007/s10922-014-9307-7

Hermann Kaindl, Mathieu Vallée, and Edin Arnautovic. 2013. Self-representation for self-configuration and monitoring in agent-based flexible automation systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43, 1 (January 2013), 164–175.

Eric Karl, David Blaauw, Dennis Sylvester, and Trevor Mudge. 2006. Reliability modeling and management in dynamic microprocessor-based systems. In *Proceedings of the 43rd Annual Design Automation Conference*. ACM, 1057–1060.

J. O. Kephart and others. 2003. The vision of autonomic computing. *Computer* 36, 1 (jan 2003), 41–50. DOI:http://dx.doi.org/10.1109/MC.2003.1160055

Jeffrey O. Kephart and David M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50.

V. B. Kleeberger, C. Gimmler-Dumont, C. Weis, A. Herkersdorf, D. Mueller-Gritschneder, S. R. Nassif, U. Schlichtmann, and N. Wehn. 2013. A cross-layer technology-based study of how memory errors impact system resilience. *IEEE Micro* 33, 4 (July 2013), 46–55. DOI:http://dx.doi.org/10.1109/MM.2013.67

Joonho Kong, Sung Woo Chung, and Kevin Skadron. 2012. Recent thermal management techniques for microprocessors. *ACM Computer Surveys* 44, 3, Article 13 (June 2012), 42 pages. DOI:http://dx.doi.org/10.1145/2187671.2187675

Georgios Kornaros and Dionisios Pnevmatikatos. 2013a. A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Transactions on Design Automation of Electronic Systems* 18, 2, Article 17 (April 2013), 38 pages. DOI:http://dx.doi.org/10.1145/2442087.2442088

Georgios Kornaros and Dionisios Pnevmatikatos. 2013b. A survey and taxonomy of on-chip monitoring of multicore systems-on-chip. *ACM Transactions on Design Automation of Electronic Systems* 18, 2, Article 17 (April 2013), 38 pages. DOI:http://dx.doi.org/10.1145/2442087.2442088

Rakesh Kumar, Keith I. Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. 2003. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36 2003)*. IEEE, 81–92.

Robert Laddaga. 2001. Active software. In *Self-Adaptive Software*. Lecture Notes in Computer Science, Vol. 1936. Springer, 11–26.

E. A. Lee. 2008. Cyber physical systems: Design challenges. In *ISORC, 2008*. 363–369. DOI:http://dx.doi.org/10.1109/ISORC.2008.25

Larkhoon Leem, Hyungmin Cho, Jason Bau, Quinn A. Jacobson, and Subhasish Mitra. 2010. ERSA: Error resilient system architecture for probabilistic applications. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*. IEEE, 1560–1565.

Charles R. Lefurgy, Alan J. Drake, Michael S. Floyd, Malcolm S. Allen-Ware, Bishop Brock, Jose A. Tierno, John B. Carter, and Robert W. Berry. 2013. Active guardband management in Power7+ to save energy and maintain reliability. *IEEE Micro* 33, 4 (2013), 35–45.

P. R. Lewis, A. Chandra, S. Parsons, E. Robinson, K. Glette, R. Bahsoon, J. Torresen, and Xin Yao. 2011. A survey of self-awareness and its application in computing systems. In *Proceedings of the 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2011)*. 102–107. DOI:http://dx.doi.org/10.1109/SASOW.2011.25

Peter R. Lewis, Arjun Chandra, Funmilade Faniyi, Kyrre Glette, Tao Chen, Rami Bahsoon, Jim Torresen, and Xin Yao. 2015. Architectural aspects of self-aware and self-expressive computing systems. *IEEE Computer* (August 2015).

Tuo Li, Muhammad Shafique, Jude Angelo Ambrose, Semeen Rehman, Jörg Henkel, and Sri Parameswaran. 2013. RASTER: Runtime adaptive spatial/temporal error resiliency for embedded processors. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 62.

Lennart Ljung. 1998. *System Identification*. Springer.

L. Madden, S. Tokmouline, T. Csail, M. Stoianov, and I. Nachman. 2007. PIPENET: A wireless sensor network for pipeline monitoring. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)*. Imperial College of London, 264–273. DOI:http://dx.doi.org/10.1109/IPSN.2007.4379686

Martina Maggio, Henry Hoffmann, Marco D. Santambrogio, Anant Agarwal, and Alberto Leva. 2011. Decision making in autonomic computing systems: Comparison of approaches and techniques. In *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC'11)*. ACM, New York, NY, 201–204. DOI:http://dx.doi.org/10.1145/1998582.1998629

Poirier Mathieu. 2013. In kernel switcher: A solution to support ARM's new big.LITTLE technology. https://events.linuxfoundation.org/images/stories/slides/elc2013_poirier.pdf.

Pietro Mercati, Andrea Bartolini, Francesco Paterna, Tajana Simunic Rosing, and Luca Benini. 2013. Workload and user experience-aware dynamic reliability management in multicore processors. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2.

Pietro Mercati, Andrea Bartolini, Francesco Paterna, Tajana Simunic Rosing, and Luca Benini. 2014a. A linux-governor based dynamic reliability manager for android mobile devices. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 1–4.

Pietro Mercati, Francesco Paterna, Andrea Bartolini, Luca Benini, and Tajana Simunic Rosing. 2014b. Dynamic variability management in mobile multicore processors under lifetime constraints. In *Proceedings of the 32nd IEEE International Conference on Computer Design (ICCD 2014)*. 448–455. DOI:http://dx.doi.org/10.1109/ICCD.2014.6974718

Subhasish Mitra, Kevin Brelsford, Young Moon Kim, H.-H. K. Lee, and Yanjing Li. 2011. Robust system design to overcome CMOS reliability challenges. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on* 1, 1 (2011), 30–41.

Subhasish Mitra, Kevin Brelsford, and Pia N. Sanda. 2010. Cross-layer resilience challenges: Metrics and optimization. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*. IEEE, 1029–1034.

Sparsh Mittal. 2014. A survey of techniques for improving energy efficiency in embedded computing systems. *International Journal of Computer Aided Engineering and Technology* (2014).

Alain Morin. 2006. Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. *Consciousness and Cognition* 15, 2 (2006), 358–371. DOI:http://dx.doi.org/10.1016/j.concog.2005.09.006

Lo. Motus, M. Meriste, and J. Preden. 2009. Towards middleware based situation awareness. In *Military Communications Conference (MILCOM)*.

Sani R. Nassif, Nikil Mehta, and Yu Cao. 2010. A resilience roadmap. In *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 1011–1016.

Ripal Nathuji and Karsten Schwan. 2007. Virtualpower: Coordinated power management in virtualized enterprise systems. In *ACM SIGOPS Operating Systems Review*, Vol. 41. ACM, 265–278.

Ulric Neisser. 1997. The roots of self-knowledge: Perceiving self, it, and thou. *Annals of the New York Academy of Sciences* 818 (June 1997), 19–33.

P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf. 1999. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and Their Applications* 14, 3 (May 1999), 54–62. DOI:http://dx.doi.org/10.1109/5254.769885

Organic Computing. Organic computing initiative. http://www.organic-computing.de/.

R. Culler, D. Polastre, and J. Szewczyk. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN 2005)*. Dept. of Computer Sciences, University of California, Berkeley, 364–369. DOI:http://dx.doi.org/10.1109/IPSN.2005.1440950

Jurgo Preden. 2014. Generating situation awareness in cyber-physical systems: Creation and exchange of situational information. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM, New York, NY.

J. Preden, J. Llinas, G. Rogava, R. Pathma, and L. Motus. 2013. On-line data validation in distributed data fusion. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV: SPIE Defense, Security and Sensing*, T. Pham, M. A. Kolodny, and K. L. Priddy (Eds.). SPIE - International Society for Optics and Photonics.

Jürgo-Sören Preden. 2012. *Enhancing Situation-Awareness, Cognition and Reasoning of Ad-Hoc Network Agents*. Ph.D. Dissertation. Tallinn University of Technology. Thesis on informatics and system engineering C56.

Jürgo-Sören Preden and J. Helander. 2006. Auto-adaptation driven by observed context histories. In *Proceedings of ECHISE (Exploiting Context Histories in Smart Environments) Workshop at UbiComp*.

Harald Psaier and Schahram Dustdar. 2011. A survey on self-healing systems: Approaches and systems. *Computing* 91, 1 (2011), 43–73.

Z. W. Pylyshyn. 1984. *Computation and Cognition* (2nd ed.). MIT Press.

Heather M. Quinn, Andre De Hon, and Nick Carter. 2011. *CCC Visioning Study: System-Level Cross-Layer Cooperation to Achieve Predictable Systems From Unpredictable Components*. Technical Report. Los Alamos National Laboratory (LANL).

Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P. Pipe, Thomas F. Wenisch, and Milo M. K. Martin. 2012. Computational sprinting. In *Proceedings of the 18th International Symposium on High Performance Computer Architecture (HPCA 2012)*. IEEE, 1–12.

Krishna K. Rangan, Gu-Yeon Wei, and David Brooks. 2009. Thread motion: Fine-grained power management for multi-core systems. In *ACM SIGARCH Computer Architecture News*, Vol. 37. ACM, 302–313.

Vijay Janapa Reddi, Meeta Sharma Gupta, Glenn Holloway, Gu-Yeon Wei, Michael D. Smith, and David Brooks. 2009. Voltage emergency prediction: Using signatures to reduce operating margins. In *Proceedings of the 15 International Symposium on High Performance Computer Architecture (HPCA 2009)*. IEEE, 18–29.

Vijay Janapa Reddi, Svilen Kanev, Wonyoung Kim, Simone Campanoni, Michael D. Smith, Gu-Yeon Wei, and David Brooks. 2010. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *MICRO*. 77–88.

Vijay Janapa Reddi, David Z. Pan, Sani R. Nassif, and Keith A. Bowman. 2012. Robust and resilient designs from the bottom-up: Technology, CAD, circuit, and system issues. In *ASP-DAC*. 7–16.

Semeen Rehman, Florian Kriebel, Duo Sun, Muhammad Shafique, and Jörg Henkel. 2014. dTune: Leveraging reliable code generation for adaptive dependability tuning under process variation and aging-induced effects. In *Proceedings of the the 51st Annual Design Automation Conference on Design Automation Conference*. ACM, 1–6.

Efraim Rotem, Alon Naveh, Doron Rajwan, Avinash Ananthakrishnan, and Eliezer Weissmann. 2012. Power-management architecture of the Intel microarchitecture code-named sandy bridge. *IEEE Micro* 32, 2 (2012), 0020–27.

Michael Rubenstein, Christian Ahler, and Radhika Nagpal. 2012. Kilobot: A low cost scalable robot system for collective behaviors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2012)*. IEEE, 3293–3298.

Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. 2014. Programmable self-assembly in a thousand-robot swarm. *Science* 345, 6198 (2014), 795–799.

Mazeiar Salehie and Ladan Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4, 2 (2009), 14.

M. G. Sánchez-Escribano and Ricardo Sanz. 2014. Emotions and the engineering of adaptiveness. In *Procedia Computer Science: Conference on Systems Engineering Research*, Vol. 28. Elsevier, 473–480. DOI:http://dx.doi.org/10.1016/j.procs.2014.03.058

Marco D. Santambrogio, Henry Hoffmann, Jonathan Eastep, and Anant Agarwal. 2010. Enabling technologies for self-aware adaptive systems. In *Proceedings of the 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 149–156.

Ricardo Sanz, Ignacio López, Manuel Rdoríguez, and Carlos Hernández. 2007. Principles for consciousness in integrated cognitive control. *Neural Networks* 20, 9 (11 2007).

Santanu Sarma and Nikil Dutt. 2014a. FPGA emulation and prototyping of a cyberphysical-system-on-chip (CPSoC). In *Proceedings of the International Symposium on Rapid System Prototyping (RSP)*.

Santanu Sarma and Nikil Dutt. 2014b. Minimal sparse observability of complex networks: Application to MPSoC sensor placement and run-time thermal estimation and tracking. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. 1–6. DOI:http://dx.doi.org/10.7873/DATE2014.342

Santanu Sarma, Nikil Dutt, P. Gupta, A. Nicolau, and N. Venkatasubramanian. 2014. On-chip self-awareness using cyberphysical-systems-on-chip (CPSoC). In *Proceedings of the 12th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*.

Santanu Sarma, Nikil Dutt, P. Gupta, A. Nicolau, and N. Venkatasubramanian. 2015. Cyberphysical-system-on-chip (CPSoC): A self-aware MPSoC paradigm with cross-layer virtual sensing and actuation. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2015*.

Santanu Sarma, Nikil Dutt, and Nalini Venkatasubramanian. 2012. Cross-layer virtual observers for embedded multiprocessor system-on-chip (MPSoC). In *Proceedings of the 11th International Workshop on Adaptive and Reflective Middleware (ARM'12)*. ACM, New York, NY, Article 4, 7 pages.

Santanu Sarma, Nikil Dutt, N. Venkatasubramaniana, A. Nicolau, and P. Gupta. 2013. *CyberPhysical-System-On-Chip (CPSoC): Sensor-Actuator Rich Self-Aware Computational Platform*. Technical Report CECS-TR-13-06. Center for Embedded Computer Systems, University of California, Irvine.

Santanu Sarma, T. Muck, L. A. D. Bathen, N. Dutt, and A. Nicolau. 2015. SmartBalance: A sensing-driven linux load balancer for energy efficiency of heterogeneous MPSoCs. In *DAC 2015*.

Ichiro Satoh. 2013. A framework for data processing at the edges of networks. In *Database and Expert Systems Applications*. 304–318.

Muhammad Shafique, Siddharth Garg, Tulika Mitra, Sri Parameswaran, and Jörg Henkel. 2014. Dark silicon as a challenge for hardware/software co-design: Invited special session paper. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 13.

Muhammad Shafique and Jörg Henkel. 2013. Agent-based distributed power management for kilo-core processors. In *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 153–160.

Muhammad Shafique, Benjamin Vogel, and Jörg Henkel. 2013. Self-adaptive hybrid dynamic power management for many-core systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 51–56.

Michael W. Shapiro. 2004. Self-healing in modern operating systems. *Queue* 2, 9 (Dec 2004), 66–75. DOI:http://dx.doi.org/10.1145/1039511.1039537

Victor Shnayder, Bor-rong Chen, Konrad Lorincz, Thaddeus R. F. Fulford Jones, and Matt Welsh. 2005. Sensor networks for medical care. In *SenSys*, Vol. 5. 314–314.

A. Singhee and R. Rutenbar. 2010. *Extreme Statistics in Nanoscale Memory Design*. Springer.

Ranjani Sridharan, Nikhil Gupta, and Rabi Mahapatra. 2008. Feedback-controlled reliability-aware power management for real-time embedded systems. In *Proceedings of the 45th ACM/IEEE Conference on Design Automation (DAC 2008)*. IEEE, 185–190.

S. Sarma, N. Dutt, and P. Gupta. 2014. *Strength of Diversity: Exploiting Cheap Heterogeneous Noisy Sensors for Accurate Full-Chip Thermal Estimation*. Technical Report CECS-TR-14-011. Univeristy of California Irvine.

Sujesha Sudevalayam and Purushottam Kulkarni. 2011. Energy harvesting sensor nodes: Survey and implications. *Communications Surveys & Tutorials, IEEE* 13, 3 (2011), 443–461.

Jin Sun, Avinash Kodi, Ahmed Louri, and Janet Meiling Wang. 2009. NBTI aware workload balancing in multi-core systems. In *Proceedings of the 2009 Quality of Electronic Design (ISQED 2009)*. IEEE, 833–838.

Jin Sun, Rui Zheng, Jyothi Velamala, Yu Cao, Roman Lysecky, Karthik Shankar, and Janet Roveda. 2013. A self-tuning design methodology for power-efficient multi-core systems. *ACM Transactions on Design Automation of Electronic Systems* 18, 1, Article 4 (Jan. 2013), 24 pages. DOI:http://dx.doi.org/10.1145/2390191.2390195

Dennis Sylvester, David Blaauw, and Eric Karl. 2006. Elastic: An adaptive self-healing architecture for unpredictable silicon. *IEEE Design & Test of Computers* 23, 6 (2006), 484–490.

E. Thelen and L. B. Smith. 1994. *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT Press, Cambridge, Massachusetts.

Akshat Verma, Puneet Ahuja, and Anindya Neogi. 2008. pMapper: Power and migration cost aware application placement in virtualized systems. In *Middleware 2008*. Springer, 243–264.

D. Vernon, G. Metta, and G. Sandini. 2007. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation* 11, 2 (April 2007), 151–180. DOI:http://dx.doi.org/10.1109/TEVC.2006.890274

Xiaorui Wang and Yefu Wang. 2011. Coordinating power control and performance management for virtualized server clusters. *IEEE Transactions on Parallel and Distributed Systems* 22, 2 (Feb. 2011), 245–259. DOI:http://dx.doi.org/10.1109/TPDS.2010.91

Yefu Wang, Kai Ma, and Xiaorui Wang. 2009. Temperature-constrained power control for chip multiprocessors with online model estimation. In *ACM SIGARCH Computer Architecture News*, Vol. 37. ACM, 314–324.

Qiang Wu, Philo Juang, Margaret Martonosi, and Douglas W. Clark. 2004. Formal online methods for voltage/frequency control in multiple clock domain microprocessors. *ACM SIGARCH Computer Architecture News* 32, 5 (2004), 248–259.

Qiang Wu, Margaret Martonosi, Douglas W. Clark, Vijay Janapa Reddi, Dan Connors, Youfeng Wu, Jin Lee, and David Brooks. 2005. A dynamic compilation framework for controlling microprocessor energy and performance. In *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 271–282.

Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. 2009. Hybrid cache architecture with disparate memory technologies. In *ACM SIGARCH Computer Architecture News*, Vol. 37. ACM, 34–45.

X. Yao and T. Higuchi. 1999. Promises and challenges of evolvable hardware. *IEEE Transactions on Systems* 29, 1 (February 1999), 87–97.

Juan Ye, Simon Dobson, and Susan McKeever. 2012. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing* 8, 1 (Feb. 2012), 36–66. DOI:http://dx.doi.org/10.1016/j.pmcj.2011.01.004

H. Zakaria, E. Yahya, and L. Fesquet. 2011. Self-adaption in SoCs. In *Autonomic Networking-on-Chip - Bio-Inspired Specification, Development, and Verification*, Phan Cong-Vinh (Ed.). CRC Press, Chapter 8.

Received June 2015; accepted October 2015