HIERARCHICAL AGENT MONITORING DESIGN PLATFORM
Towards Self-aware and Adaptive Embedded Systems

Liang Guang, Bo Yang, Juha Plosila, Jouni Isoaho and Hannu Tenhunen
Department of Information Technology, University of Turku, Turku, Finland

Keywords: Adaptiveness, Design approach, Embedded systems, Hierarchical agent monitoring, Self-awareness.

Abstract: Hierarchical agent monitoring design platform (HAM) is presented as a generic design approach for the emerging self-aware and adaptive embedded systems. Such systems, with various existing proposals for different advanced features, call for a concrete, practical and portable design approach. HAM addresses this necessity by providing a scalable and generically applicable design platform. This paper elaborately describes the hierarchical agent monitoring architecture, with extensive reference to the state-of-the-art technology in embedded systems. Two case studies are exemplified to demonstrate the design process and benefits of HAM design platform. One is about hierarchical agent monitored Network-on-Chip with quantitative experiments of hierarchical energy management. The other one is a projectional study of applying HAM on smart house systems, focusing on the design for enhanced dependability.

1 INTRODUCTION

The research on embedded computing has entered the era of massively parallel and distributed systems. For one thing, the number of components integrated on a single chip has been constantly increasing due to the continuous scaling of transistor sizes. For instance, an 80-tile 1.28TFLOPS single-chip processor is fabricated with 100-million transistors (Vangal et al., 2007). For another, more physically scattered entities are being integrated into distributed embedded systems, as indicated by the emerging platform of Cyber-Physical Systems (CPS) (Lee, 2008). As the complexity of embedded systems is steadily increasing, the concept of autonomic computing gets widely recognized, which refers to computing systems that can manage themselves given high-level objectives from administrators (Kephart and Chess, 2003). Since the release of IBM autonomic computing manifesto (Horn, 2001), many works have been developing the concept and its implementation, for instance the proposal of organic computing (Würtz, 2008).

Despite the diversity of works to promote the adaptiveness and self-management of computing systems, there remain major challenges to be tackled. In particular, generic and scalable design approaches are needed to reduce the design complexity. Previous works have proposed a list of appealing self-aware and adaptive functions, including self-configuration, self-optimization, self-healing and self-protection (Kephart and Chess, 2003). For instance, (Al Faruque et al., 2008) presents algorithms for run-time mapping on Network-on-Chip (NoC) with optimized computational effort. However, there are few works about generic design approaches to realize these functions. Considering the complexity and diversity of emerging parallel and distributed systems, designing adaptive features on these systems in an ad-hoc manner is very time-consuming and non-scalable.

This paper presents a generic design platform, hierarchical agent monitoring (HAM), for designing embedded system with self-aware and adaptive features. Our previous work (Guang et al., 2010) presented the formal specification of the design platform, while this work focused on the design theory and system architecture with studies on two specific architectures, NoC and smart houses. We will firstly present our conceptual definition of self-aware and adaptive systems (Section 2). Then we will extensively present the HAM platform (Section 3), including the benefits of platform-based design (Section 3.1) and the hierarchical agent monitoring architecture (Section 3.2). The design example of hierarchical agent monitored NoC is given in Section 4. The agent functional partition and architectural design are presented for hierarchical energy management. Section 5 projects the use of HAM design platform for distributed embedded systems, exemplifying the smart house system.

Guang L., Yang B., Plosila J., Isoaho J. and Tenhunen H..
HIERARCHICAL AGENT MONITORING DESIGN PLATFORM - Towards Self-aware and Adaptive Embedded Systems.
DOI: 10.5220/0003396805730581
In *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems* (SAAES-2011), pages 573-581
ISBN: 978-989-8425-48-5

The study will focus on the application of HAM to generalize the design process for any potential functions and enhanced dependability. The paper is concluded in Section 6 with discussions of future works.

2 SELF-AWARE AND ADAPTIVE SYSTEMS

Self-aware and adaptive systems originate from the classic autonomic computing concept, while focusing on the two distinctive aspects of system behaviors, sensing (for awareness) and reconfiguration (for adaptation).

We define a self-aware and adaptive embedded system as one that is monitoring its own state and the environment in order to achieve the expected performance under potential environmental changes. The performance refers to both functional (e.g. execution time) and non-functional metrics (e.g. energy efficiency), either as hard constraints (e.g. power budget) or soft requirements (e.g. with as low power as possible).

A self-aware and adaptive system can be abstracted as in Fig. 1. In terms of awareness, the system needs to be aware of its objectives, such as the power budget and dependability requirements, and the run-time status including its own status and the surrounding environment. In terms of adaptation, the system processes the gathered information based on intrinsic cost functions, and decides on the proper reconfiguration to achieve the objectives. Cost functions are pre-configured or dynamically reconfigurable models relating the status parameters to performance metrics. For instance, the energy consumption can be modeled as a function of input parameters including system activity ratio and supply voltage.

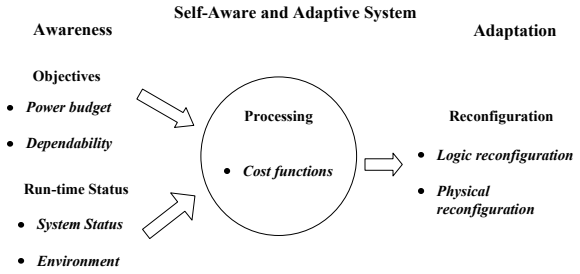

Figure 1: Concept of Self-Aware and Adaptive Systems.

Compared to the classic concept of autonomic computing with the four aspects of self-management (self-configuration, self-optimization, self-healing and self-protection (Kephart and Chess, 2003)), the notion of self-aware and adaptiveness focuses on the two distinctive phases, awareness and adaptation, which are universal for any type of self-management. As illustrated in Fig 2., awareness is the prerequisite for effective reconfiguration. A platform with a generic support for adaptation triggered by self-awareness is able to provide different types of autonomic operations, given proper configuration of the algorithms and microarchitectures.

The major obstacle of autonomic computing is the complexity (Horn, 2001). The development and innovation of design approaches have always been an essential enabler to effectively reducing the design complexity and time-to-market (Keutzer et al., 2000). For self-aware and adaptive systems, in order to make the design process generic and reusable for any type of adaptive functions, it is important to innovate on the design approach.

3 HIERARCHICAL AGENT MONITORING DESIGN PLATFORM

3.1 Platform-based Design

Platform-based design (Sangiovanni-Vincentelli and Martin, 2001; Keutzer et al., 2000) is an important design methodology addressing design productivity and reusability. Platform-based design abstracts the systems as platforms, which can be modeled at different levels of elaboration. With platform-based design, instead of building each component from scratch, the designers can reuse a proper platform as a foundation, and modify the components as needed. A concrete example applying the platform-based design is Network-on-Chip (NoC) (Rabaey, 2004) (Fig. 3), which provides a reusable platform for a large diversity of parallel applications.

Hierarchical agent monitoring design platform follows the platform-based design methodology by adding a new design dimension for monitoring and diagnostic services. As the unpredictableness and variations of system and circuit increase, it becomes more necessary to utilize various types of monitoring services, for instance power monitoring (Shang et al., 2003) and thermal monitoring (Shang et al., 2004). HAM design platform provides a design layer dedicated to such monitoring services, which integrates a generic monitoring architecture (Fig. 3). The monitoring architecture is built upon existing platforms with modularized computation and communication elements.

The term agent originally comes from artificial

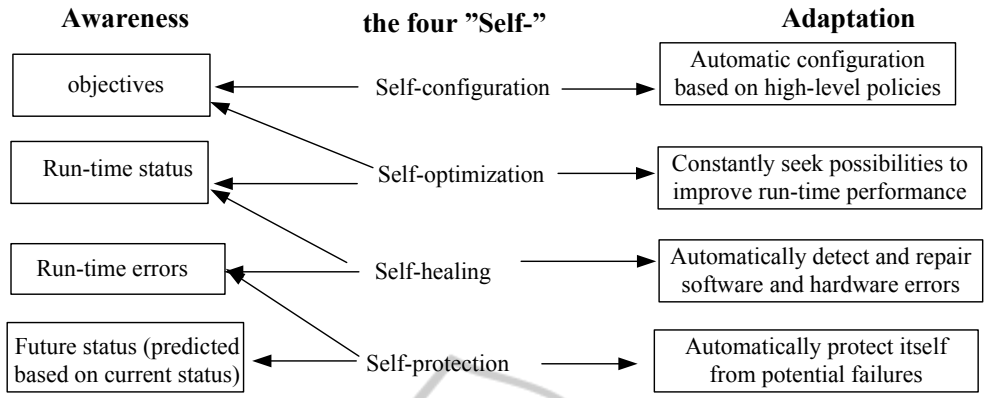

Figure 2: Self-Aware and Adaptive Systems: A Special Perspective on Autonomic Computing.

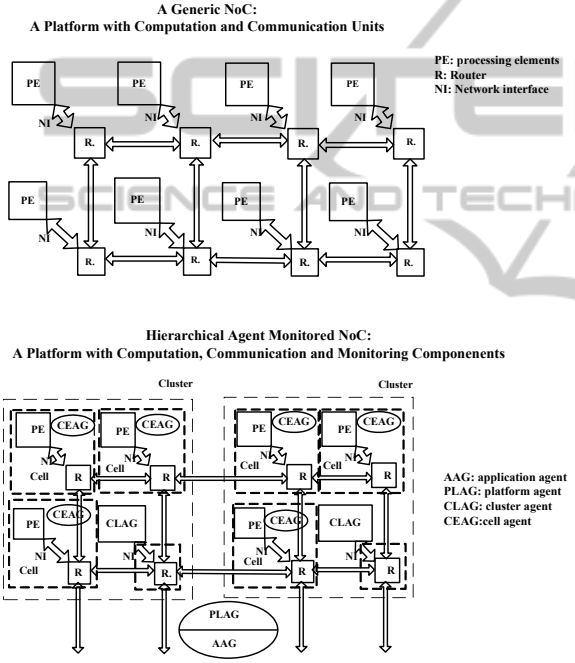

Figure 3: The Contrast between Conventional NoC Platform and Hierarchical Agent Monitored NoC Platform.

intelligence and software engineering, referring to a large diversity of intelligent components which are sensing, monitoring and reconfiguring the systems. From the design method's perspective, agent becomes a design abstraction, which may have different implementations, software, hardware or hybrid.

3.2 Hierarchical Agent Monitoring Architecture

A generic monitoring architecture, hierarchical agent structure, is provided by the HAM design platform (Fig. 4). Agents at different levels make a joint effort to offer self-aware and adaptive system features

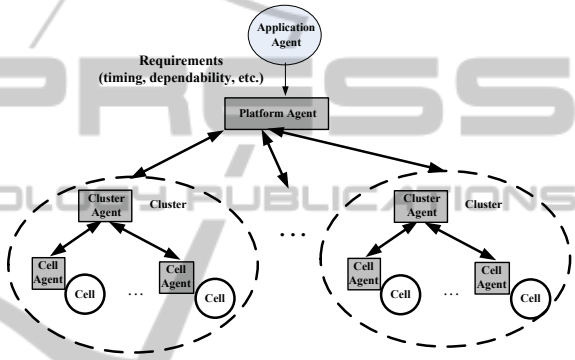

Figure 4: Hierarchical Agent Monitoring Architecture.

(Table 1).

The application agent is a module capturing the run-time requirements of the application, for instance timing constraints (e.g. soft or hard deadlines), and dependability requirements (e.g. mean-time-to-failure). In other words, the application agent enables the system to be aware of its objectives.

The platform agent is aware of the overall system performance, and makes major adaptation that affect the whole platform. For example, it is responsible for resource allocation, including mapping processors to application tasks, and network configuration. At run-time, it traces the overall system performance, for example the total power consumption or the general speed. When necessary the platform agent may reconfigure system settings to be applied to the whole platform, for instance the network topology.

Each cluster agent is responsible for a particular region (a cluster) in the platform. It follows the general settings as decided by the platform agent, while being able to configure its own cluster. Dynamic voltage and frequency scaling, for instance, can be applied to different voltage islands on a many-core platform (Ogras et al., 2009). Such regional services belong to the responsibilities of cluster agents.

Table 1: Hierarchical Agent Monitoring for Self-Aware and Adaptive Systems.

Agent Level	Main Features	Awareness	Adaptation
Application Agent	platform independent	system objectives	modify the system objectives
Platform Agent	coarse granularity, platform-wide influence, infrequent	overall system performance, cluster agents status, lower level components (when cluster agents fail)	configure general system setting, perform major system reconfiguration, monitor low level components (when cluster agents fail)
Cluster Agent	medium granularity, only influence one cluster	cluster performance, cell agents, cells (when cell agents fail)	configure cluster setting, re-configure clusters, monitor cells (when cell agents fail)
Cell Agent	fine granularity, quick response, often issued	cell performance, circuit state	reconfigure cell

Cell agents work on a fine granularity. Each cell agent tightly monitors the local hardware with very quick response. (Truong et al., 2009) presents a many-core platform where each core can adjust its own voltage and frequency quickly. In this case, the cell agent shall be responsible for voltage and frequency switching in each core.

As agents may also have errors themselves, each agent is also responsible for monitoring its lower-level agents. In case one agent fails, its supervising agent shall take over its resources until the failure is fixed.

4 HIERARCHICAL AGENT MONITORED NETWORK-ON-CHIP

NoC is a widely adopted scalable architecture for massively on-chip parallelism (Jantsch and Tenhunen, 2003). Due to the intensity and diversity of parallel computing on NoC, various run-time monitoring and optimization techniques have been proposed (Table 2). These monitoring services provide the NoC system with a diversity of self-aware and adaptive features. HAM platform can be conveniently applied to many-core NoC systems for the design of these monitoring services.

4.1 Exemplified Monitoring Operations for Energy Efficient Communication

Here we exemplify a set of monitoring operations on NoC to perform energy-efficient communication. This set of operations provide the awareness and adaptation at four agent levels (Table 3), in order to achieve the minimal communication energy while meeting the performance objective.

We consider a many-core NoC platform running multiple sub-applications. Each sub-application will run in a cluster, monitored by a cluster agent. The application agent is aware of the performance requirement of each sub-application. In this example, we exemplify the average communication latency as the performance requirement. The platform agent firstly transforms the communication latency requirements to thresholds of network load. The network load threshold is an indicator of network congestion, which is directly related to the average communication latency (Guang and Jantsch, 2006). In a buffered NoC architecture, the network load can be quantified by the buffer load, which is the percentage of occupied buffers out of the total number of buffers in the network. Given the network topology and flow control mechanism, we can simulate the proper buffer load threshold to provide the required communication latency before execution time (Table 4). The platform agent also performs energy-aware mapping for each sub-application, which minimizes the overall communication volumes between each two processes multiplied by their distances after mapping. While the design platform does not limit the choice of specific application mapping algorithm, here an in-house tree-based mapping algorithm is utilized ((Yang et al., 2010); Fig 5). Simply put, the network is abstracted into a tree structure. The process with the highest communication volume is mapped onto the root node in the tree. Then the algorithm searches for the process having the highest communication volume with the mapped nodes, and maps it onto the highest available node. The iteration continues until all the processes are mapped. The cluster level agent is performing DVFS (dynamic voltage and frequency scal-

Table 2: Representative Monitoring Operations on Many-core NoCs.

Monitoring Services	Awareness	Adaptation
Monitoring Performance (Hu and Marculescu, 2005)	communication speed	improve communication speed
Fault Tolerance (Lehtonen et al., 2007)	component (e.g. link) error	fix faults and errors
power efficiency (Shang et al., 2003)	network load	tradeoff communication performance with power efficiency
Thermal monitoring (Shang et al., 2004)	local temperature	adjust temperature evenness and avoid hotspot

Table 3: Exemplified Hierarchical Agent Monitoring on NoC for Energy Efficiency.

Level of Agent	Awareness	Adaptation
Application Agent	communication requirements	none
Platform Agent	communication requirements, application communication graph	transform performance requirements to network load threshold, energy-aware application mapping
Cluster Agent	cluster network load	intra-cluster DVFS
Cell Agent	local buffer load	report the load to the cluster agent

ing (Shang et al., 2003)) in each cluster, based on the network threshold dictated by the platform agent. The average network load in each cluster is calculated from the buffer load of each router, reported by every cell agent. The cell agent is dedicated hardware circuit inside each router, to calculate the buffer load. Fig. 5 illustrates the exemplified the hierarchical monitoring operations for energy efficiency.

4.2 Experimental Setting

Several synthetic and real traces were used to demonstrate the operations (Table 4). A cycle-accurate simulator is utilized to simulate an 8*8 mesh NoC, assuming 65*nm* technology. Each channel on the NoC is 1 *mm* long and 64-bit wide (32-bit per direction). For DVFS, two voltage and frequency levels, (1.2V,2GHz) and (0.8V, 1GHz) are adopted. The energy consumption on routers and links is estimated from Orion 2.0 (Kahng et al., 2009).

4.3 Quantitative Evaluation

The average per-flit (a 32-bit word) energy consumption for each traffic trace, as well as the average per-flit latency, are reported in Table 5. We can observe that the hierarchical monitoring operations effectively achieve energy optimization under the performance constraint. Compared to static high voltage and frequency supply, platform-level application mapping reduces the energy consumption for sub-applications with traffic spatial unevenness (H.264 and hotspot). When application mapping does not provide significant energy reduction, cluster-level DVFS still offers improved energy efficiency. When hierarchical monitoring operations are performed, we achieve considerable energy saving for all studied traffic traces (19%-64%). In all cases, the average communication latency is well below the upper boundary set by the application agent.

The platform agent is realized by software with the application mapping algorithm written in C code of around 600 lines. Cluster agents and cell agents are realized as hardware circuits, with standard synthesis flow. Major accessory circuits to support dynamic voltage and frequency scaling are DC-DC converters and PLLs (phase-locked loop). We estimated that, on an 8*8 NoC platform, the hardware overhead of hierarchical agents and accessory circuits is less than $2mm^2$, which is very small compared to common die sizes (for instance $275mm^2$ as in (Vangal et al., 2007)). The areas for DC converters and PLLs are scaled from (Wibben and Harjani, 2007) and (Tierno et al., 2008) respectively.

5 HIERARCHICAL AGENT MONITORED SMART HOUSE: A PROJECTION

Hierarchical agent monitoring design platform is generic for any type of parallel and distributed embedded system. The more widespread and heterogeneous the system is, the more effective HAM platform is compared to ad-hoc designs. In this section, we an-

Table 4: Setting of Traffic Traces Mapped on an 8*8 NoC.

Trace	Features	Maximal Average Latency	Mapping	Cluster Load threshold
linearly changing traffic (I)	traffic injection linearly increases	10*ns*	3*3 mesh	0.3
hotspot traffic (II) (Lu et al., 2008)	4 processing elements receive 50% traffic	20*ns*	5*5 mesh	0.15
MP3 trace (III) (Truscan et al., 2008)	audio decoder	15*ns*	5*3 mesh	0.18
H.264 trace (IV) (Latif et al., 2008)	video encoder	15*ns*	5*3 mesh	0.06

Table 5: Experiment Results of Energy Efficiency and Communication Latency.

Trace	Average energy with high voltage and frequency e-11J	Average Energy with application mapping e-11J	Average Energy with hierarchical operations e-11J	Latency Boundary *ns*	Average Latency with hierarchical operations *ns*
I	4.17	4.17	3.37	10	9.78
II	6.12	4.91	4.19	20	15.20
III	4.01	4.2	2.22	15	7.15
IV	5.50	3.47	1.98	15	6.17

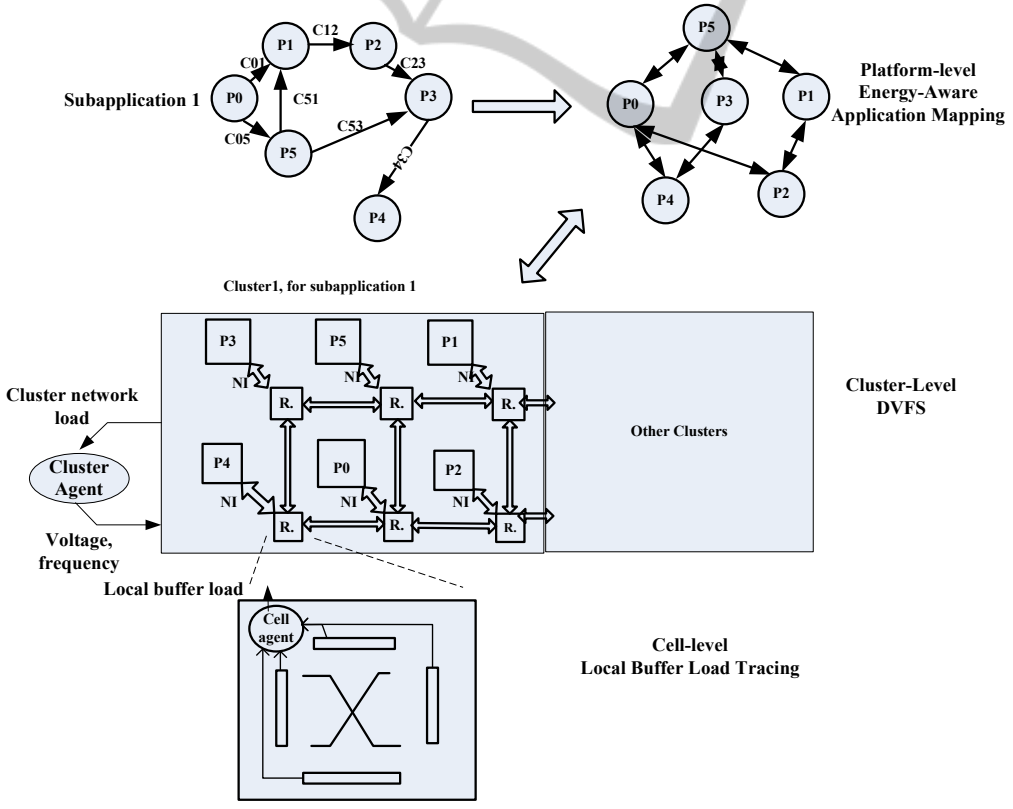

Figure 5: Hierarchical Agent Monitoring Operations for Energy-Efficient Communication.

alyze the application of HAM on smart houses, as an example of distributed embedded systems.

5.1 Smart House: A Hierarchical View

The concept of autonomic computing is applied on smart houses to facilitate human daily life. A diver-

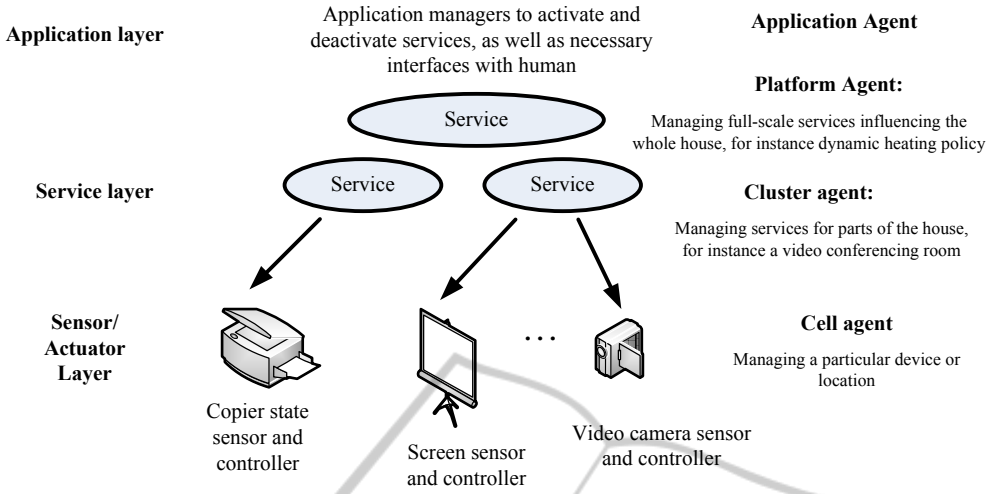

Figure 6: Hierarchical View of Smart House (smart house hierarchy adapted from (Helal et al., 2005)).

sity of self-aware and adaptive functions have been proposed (Helal et al., 2005; Stefanov et al., 2004), in order to save domestic energy consumption, improve dependability against various failures (e.g. power failures), or help patients and people with disabilities.

Existing works have focused on the proposal of more functions, for instance run-time video conferencing to enable distant dining (Helal et al., 2005) or robotic systems for moving assistance (Stefanov et al., 2004). While these features seem to be technology feasible, we still need a scalable design approach so that new features can be efficiently added with an increasing scale of distribution. (Helal et al., 2005) proposes a hierarchical view for the large diversity of control and monitoring components in a smart house, which seamlessly fits into the hierarchical agent monitoring architecture (Fig. 6).

The sensor and actuator layer is the one directly interfacing with the physical world. For instance, on video cameras, autonomous sensors can be installed to identify any suspicious intruders. Cell agents shall be responsible for this layer, since they are local monitors based on the HAM design platform. The service layer defines all types of basic or composite services provided in the smart house. On HAM design platform, these services shall be assigned to either the platform agent or the cluster agent, depending the scope of influences. Services which are managed centrally with a system-wide influence, such as the universal heating policy, is the responsibility of the platform agent. Other services which affect part of the houses are managed by the cluster agent, for instance the management of a video conferencing room. Some services can be provided by either the platform or cluster agent. For instance, if each room may adjust its own temperature, then heating will be managed by the cluster agent. The application layer includes the application manager to activate or deactivate the services and the interface to external monitors, for instance designers. This layer shall be managed by the application agent, which decides if certain services are needed for the smart house.

5.2 HAM Design Platform for Self-Aware and Adaptive Smart Houses

HAM design platform, in addition to being a generic platform for designing control and monitoring smart houses, provides a monitoring architecture which offers self-awareness and adaptiveness systematically in the smarthouse, as illustrated by Table 6.

One of the major benefits of hierarchical agent monitoring in providing self-aware and adaptiveness is the enhanced dependability. Existing research is mostly concerned with the fault or errors of original devices, for instance the camera, copier or the heating system. However, the added monitors, either hardware or software, may have faults themselves. For example, the adaptive sensor on the camera itself may be broken, or the software managing the heating system may have errors. Hierarchical agent monitoring architecture relies on the hierarchical supervision of agents to minimize the possibility of undetected monitor errors, as illustrated in Table 7. Simply put, each level of agent needs to regularly check the state of the lower-level agents. At the highest level, the human may need to check if the platform agent is in a healthy state, although such checking is very infrequent and only applied on the platform agent. One convenient way to perform such state checking is to

Table 6: Example Services in Self-Aware and Adaptive Smart Houses with Hierarchical Agent Monitoring .

Agent Level	Awareness	Adaptation
Application Agent	Required Services	Activate or Deactivate Services
Platform Agent	the availability of power supply, the average temperature in the house, any intrusion	switch to power shut-down mode, adjust the house temperature, switch to guard mode
Cluster Agent	the temperature of the green house, the quality of the video conferencing	adjust the temperature, adjust the general setting of video conferencing (camera, projector, loudspeaker, etc.)
Cell Agent	the state of the surveillance camera, the state of the copier machine	adjust the setting of the camera and report intrusions to the platform agent, adjust the setting of copier machines

Table 7: Enhanced Dependability in Hierarchical Agent Monitored Smart Houses.

Agent Level	Monitoring for dependability	Example
Application Agent	the state of platform agent	the application agent tests if all functions of the platform agent work normally
Platform Agent	the state of the platform, the state of cluster agent	check the temperature and power supplies of the house, check if the cluster agent monitoring the video conferencing room works normally
Cluster Agent	the state of the cluster, the state of cell agents	check if the temperature of a particular room remains normal, check if the cell agent monitoring the surveillance camera works normally
Cell Agent	the state of the cell	check if the surveillance camera works normally

regularly send testing signals. A healthy agent should be able to respond with the correct reply within an expected time period.

6 CONCLUSIONS AND FUTURE WORK

This paper presented hierarchical agent monitoring (HAM) design platform for parallel and distributed embedded systems. HAM approaches the goal of autonomic computing by providing a scalable and portable design framework, while focusing on the potential integration of various self-aware and adaptive features. The paper extensively presented the functional partition among agents and their interactions to fulfill the intended monitoring services. We demonstrated the design platform on the many-core NoC architecture, exemplifying the run-time energy management. With quantitative evaluations using real and synthetic benchmarks, we observed the functional effectiveness and low physical overhead of HAM. A projectional study on smart houses illustrates how HAM can be applied to distributed embedded system, for instance to provide enhanced dependability.

At current stage, the development of HAM prioritizes the study of generic design approach over specific algorithms and implementations. We will continue to research on more detailed architectures with algorithm and implementation exploration. In particular, we are building elaborated simulation for a virtual smart house model, using high-level tools for instance Matlab. The study will focus on the system dependability against potential errors, for instance processor errors, power failure or heavy wireless channel noise.

ACKNOWLEDGEMENTS

The work is supported by the Foundation of Nokia Corporation.

REFERENCES

Al Faruque, M. A., Krist, R., and Henkel, J. (2008). Adam: run-time agent-based distributed application mapping for on-chip communication. In *DAC '08: Proceedings of the 45th annual Design Automation Conference*, pages 760–765, New York, NY, USA. ACM.

Guang, L. and Jantsch, A. (2006). Adaptive power management for the on-chip communication network. In *Proc. of Euromicro DSD'06*, pages 649–656.

Guang, L., Plosila, J., Isoaho, J., and Tenhunen, H. (2010). 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):86–105.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. (2005). The gator tech smart house: A programmable pervasive space. *Computer*, 38:50–60.

Horn, P. (2001). Autonomic computing: Ibm's perspective on the state of information technology. online.

Hu, J. and Marculescu, R. (2005). Energy and performance-aware mapping for regular noc architectures. *IEEE Transactions on CAD*, 24(4):551–562.

Jantsch, A. and Tenhunen, H. (2003). *Networks on Chip*. Kluwer Academic Publishers.

Kahng, A., Li, B., Peh, L.-S., and Samadi, K. (2009). Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proc. DATE '09*, pages 423–428.

Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Keutzer, K., Newton, A., Rabaey, J., and Sangiovanni-Vincentelli, A. (2000). System-level design: orthogonalization of concerns and platform-based design. *IEEE Transactions on CAD*, 19(12):1523–1543.

Latif, K., Niazi, M., Tenhunen, H., Seceleanu, T., and Sezer, S. (2008). Application development flow for on-chip distributed architectures. In *Proceeding of SOC Conference, 2008 IEEE International*.

Lee, E. A. (2008). Cyber physical systems: Design challenges. Technical Report UCB/EECS-2008-8, EECS Department, University of California, Berkeley.

Lehtonen, T., Liljeberg, P., and Plosila, J. (2007). Online reconfigurable self-timed links for fault tolerant noc. *VLSI Design*, 2007:13.

Lu, Z., Jantsch, A., Salminen, E., and Grecu, C. (2008). Network-on-chip benchmarking specification part 2: Microbenchmark specification version 1.0. Technical report, OCP International Partnership Association.

Ogras, U., Marculescu, R., Marculescu, D., and Jung, E. G. (2009). Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Transactions on VLSI*, 17(3):330–341.

Rabaey, J. M. (2004). *Interconnect-centric Design for Advanced SoC and NoC*, chapter System-on-chip challenges in the deep-sub-micron era, a case for the network-on-a-chip, pages 3–24. Kluwer Academic Publishers.

Sangiovanni-Vincentelli, A. and Martin, G. (2001). Platform-based design and software design methodology for embedded systems. *IEEE Des. Test*, 18(6):23–33.

Shang, L., Peh, L., Kumar, A., and Jha, N. (2004). Thermal modeling, characterization and management of on-chip networks. In *Proc. 37th International Symposium on Microarchitecture MICRO-37 2004*, pages 67–78.

Shang, L., Peh, L.-S., and Jha, N. (2003). Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proc. of HPCA 2003*, pages 91–102.

Stefanov, D. H., Bien, Z., and Bang, W.-C. (2004). The smart house for older persons and persons with physical disabilities: structure, technology arrangements, and perspectives. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12:228–250.

Tierno, J., Rylyakov, A., and Friedman, D. (2008). A wide power supply range, wide tuning range, all static cmos all digital pll in 65 nm soi. *IEEE Journal of Solid-State Circuits*, 43(1):42–51.

Truong, D., Cheng, W., Mohsenin, T., Yu, Z., Jacobson, A., Landge, G., Meeuwsen, M., Watnik, C., Tran, A., Xiao, Z., Work, E., Webb, J., Mejia, P., and Baas, B. (2009). A 167-processor computational platform in 65 nm cmos. *IEEE Journal of Solid State Circuits*, 44(4):1130–1144.

Truscan, D., Seceleanu, T., Lilius, J., and Tenhunen, H. (2008). A model-based design process for the segbus distributed architecture. In *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, pages 307–316. IEEE Computer Society.

Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y., and Borkar, N. (2007). An 80-tile 1.28tflops network-on-chip in 65nm cmos. In *Proc. Digest of Technical Papers. of ISSCC 2007*, pages 98–589.

Wibben, J. and Harjani, R. (2007). A high efficiency dc-dc converter using 2nh on-chip inductors. In *Proc. IEEE Symposium on VLSI Circuits*, pages 22–23.

Würtz, R. P., editor (2008). *Organic Computing*. Springer.

Yang, B., Guang, L., Canhao, X. T., Yin, A. W., Tero Säntti, T., and Plosila, J. (2010). Multi-application multi-step mapping method for many-core network-on-chips. In *Proc. of Norchip 2010*.