A DESIGN SPACE EXPLORATION FRAMEWORK FOR AUTOMOTIVE EMBEDDED SYSTEMS AND THEIR POWER MANAGEMENT

Gregor Walla
Walter Stechele
Andreas Herkersdorf
Institute for Integrated Systems
Technische Universitaet Muenchen
Munich, Germany
Email: gregor.walla@tum.de

Zaur Molotnikov
Andreas Barthels
Institute for Informatics
Technische Universitaet Muenchen
Munich, Germany

Hans-Ulrich Michel
BMW Group
Research and Technology
Munich, Germany

KEYWORDS

Automotive, E/E architecture, embedded, modeling, simulation, design space exploration, power consumption

ABSTRACT

The E/E (electric/electronic) architecture of a modern vehicle is a complex distributed system, where up to 80 electronic control units (ECUs), interconnected by several communication buses, need to collaborate with each other in order to implement the various comfort and safety features.

The presented E/E design space exploration framework supports engineers during the development process of new E/E architectures by providing a graphical modeling and simulation environment with an interactive visualization of simulation-based results. Furthermore, it contains an advanced business logic which administrates the modeling, storing, retrieving and cloning of evaluation sessions consisting of complex experiments. Due to a high-level modeling approach, future architectures can be evaluated in respect to power consumption and performance values already in an early stage of the design process and design alternatives can be easily compared with each other.

INTRODUCTION

The E/E (electric/electronic) architecture of a modern vehicle consists of a large number of electronic control units (ECUs) forming a complex distributed embedded system. Sensors detect the surrounding environment, computational units process the obtained information and actuators control the behavior of the vehicle.

With every vehicle generation the demand for safety and comfort features is increasing, resulting in an ever-increasing complexity of the overall system. Various subsystems need to collaborate with each other, which makes the design of the E/E architecture a major challenge. Furthermore, the power consumption of the electric and electronic components is also becoming a subject of consideration. It negatively affects the fuel consumption of vehicles with combustion engines or the maximum range for electric vehicles.

In order to address this situation, tool support for modeling and simulation in an early phase of the development process is necessary. The engineer needs to evaluate design alternatives and check whether all design goals will be achieved, e.g. functional requirements, performance constraints or power budgets.

The commercially available tool PREEvision allows model-based E/E development (Vector Informatik GmbH, 2013). It provides several abstraction layers for modeling a system from different architectural viewpoints, e.g. requirements, logical architecture, hardware component architecture. Through the usage of different metrics, it is possible to evaluate static properties of the system, such as communication dependencies. However, it is not possible to simulate the models and gain information about the dynamic behavior.

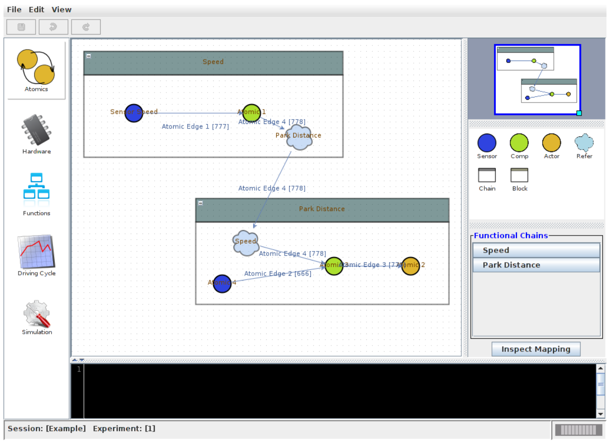
Another CASE tool for the model-based development of embedded systems is AutoFOCUS 3 which is specialized for reactive and embedded systems (Hölzl and Feilkas, 2011). It supports different layers of models and allows for timing simulation and verification of the modeled application.

The tool box UPPAAL is used for modeling, simulation and verification of real-time systems. It focuses on model checking of applications that can be modeled as networks of timed automata extended with integer variables, structured data types, user defined functions, and channel synchronization (Behrmann et al., 2004).

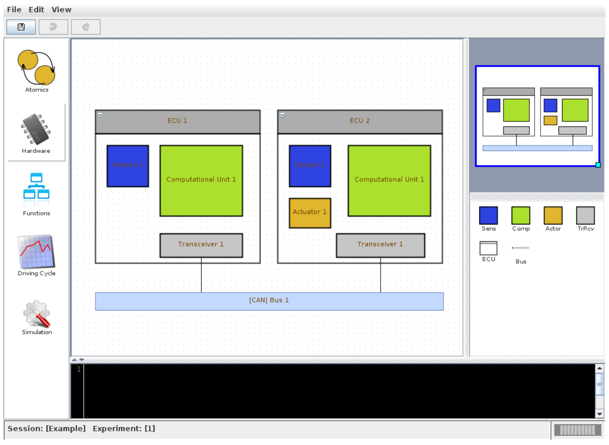
The performance and timing analysis tool SymTA/S uses formal scheduling analysis techniques and symbolic simulation in order to determine system-level performance data such as end-to-end latencies, bus and processor utilization, and worst-case scheduling scenarios (Henia et al., 2005). It can be used for design space exploration and verification of heterogeneous architectures.

In this paper, we propose E/E-MaSiF (E/E Modeling and Simulation Framework), a domain specific framework, specialized for the evaluation of E/E architectures. The novelty is the focus on the design space exploration with respect to power consumption in an early phase of the development process. The framework provides a visual modeling environment and supports the engineer through basic consistency checks and automated genera-

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)



(a) Functional Network Consisting of Two Functional Chains and 5 Functional Blocks



(b) Technical Architecture Consisting of Two ECUs Interconnected by a Communication Bus

Figure 1: Screenshots of the Visual Modeling Environment (*VME*)

tion of parts of the model.

Once an E/E architecture is fully specified, the framework can simulate it with respect to power consumption and performance values and provide a graphical representation of the results. Due to the high-level modeling approach, the simulation results are only approximated values, but it makes it possible to compare design alternatives already in an early stage of the development process. The engineer can clone and reuse existing E/E models, modify them, and evaluate the behavior of the changed system.

The rest of this paper is structured as follows: the next section introduces the modeling framework and the overall UI concept. Later on, the system architecture is described. As one building block of the architecture the business logic is highlighted and described in more detail. Afterwards, an example of the visualization of the simulation-based results is presented. Finally, the last section concludes the paper and gives an outlook on future work.

DESIGN SPACE EXPLORATION

The presented framework is built around the ITE-Sim simulator (Walla et al., 2012b) and is used to evaluate design alternatives in an early stage of the E/E architecture development process. The framework automatically generates the necessary input files for the simulator and provides a visual representation of the simulation results to the user.

In order to cope with the overall complexity, model-based development has to be applied. Figure 1 shows two screenshots of the modeling environment. The user can switch between the different views of the application by using the navigation bar on the left-hand side.

As can be seen in figure 1a the different functions of a vehicle can be modeled as directed graphs called functional chains, where the nodes represent functional blocks and the edges indicate the communication flow (Hillenbrand and Muller-Glaser, 2009). This abstraction indicates the logical architecture of the application, since the actual implementation is not yet available during that design phase. However, the internal behavior of each functional block is further specified through a separate trace primitive description. This concept was introduced in (Walla et al., 2012b). It allows an annotation of each functional block with it's computation and communication properties, i.e. data to be processed when a block is executed or the amount of data to be transmitted when a block is communicating with other blocks. Through this abstraction of the real implementation, the modeling of the functions of a vehicle is independent from the underlying hardware.

Figure 1b shows the modeling view of the technical architecture of the vehicle, where the hardware layout of the distributed system can be specified. The user can drag and drop new ECUs into the view and connect them via communication buses. Furthermore, each ECU is described in more detail by defining sub-components inside it, i.e. computational units, communication transceivers, sensor interfaces, or actuators. The sub-components are assumed to be operated in various power states with different power and performance properties.

Having modeled the functional chains and the technical architecture, the designer needs to decide which functional block will be executed on which hardware component. This mapping is restricted by various functional constraints, e.g. location of sensor/actuators or available hardware resources, but is mainly based on the knowledge of previous vehicle generations.

However, since power consumption is becoming increasingly important, the partitioning needs to be done also with energy-efficiency considerations in mind, e.g. power savings due to temporarily switching off unused functions or ECUs is highly dependent on the deployment of the various vehicle functions (Walla et al., 2012a).

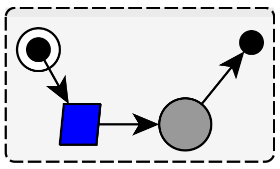


Figure 2: Exemplary Power Management Plan, Starting up the Hardware, then Running Functional Block

In order to evaluate the power consumption of the system, the power management strategy needs to be defined. This is done by modeling a set of power management plans (PMP) for each ECU. A PMP is a directed graph over power states of the sub-components of an ECU and functional blocks mapped on the ECU together with timing conditions (Barthels et al., 2011). On the one hand, this graph indicates the scheduling of the functional blocks, and on the other hand, it describes the change of the operating points of the sub-components, e.g. switching to an idle-mode when all required functional blocks have been executed.

An example of a PMP is shown in figure 2. The concept of PMPs enables the evaluation of various power management strategies, like Partial Networking (Fuchs et al., 2010) or Pretended Networking (Schmutzler et al., 2010).

With this information it is possible to evaluate the modeled E/E architecture with respect to power consumption. The user defines a specific driving scenario (Samuel et al., 2002), which corresponds to the input for the sensors over time, and can simulate the execution of the vehicle functions.

In order to be able to perform a design space exploration, it is possible to clone a modeled E/E architecture, perform modifications to them, for example, change the mapping of functional blocks or the power management strategy, and repeat the simulation. This enables the designer to compare the power related consequences of different experiment configurations in a short period of time.

SYSTEM ARCHITECTURE

The overall system architecture is shown in figure 3 in form of components. The components are arranged in tiers. They have data flow between them, user and simulator interaction, and a database for persistency. The rectangular components encapsulated in the tiers together represent a standalone system, a visual modeling environment (VME). The VME supports the user on the way towards the full experiment configuration, needed to perform simulation, serving as an intermediary between the simulator and the user.

On the presentation tier the user interacts with a graphical user interface, built with Swing and JGraphX technologies. The graphical user interface invokes processing functions of the business logic (BL) component in the logic tier. The business logic contains the data model of

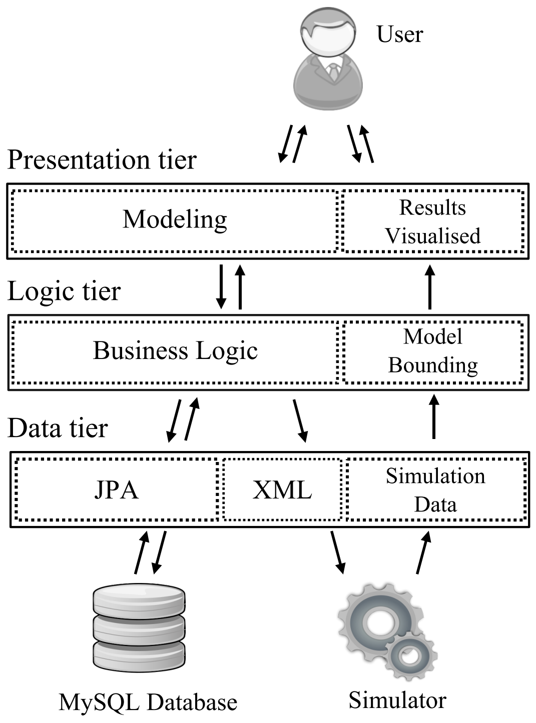


Figure 3: Schematical System Architecture Diagram

the experiment as well as domain relevant functionality, which is discussed separately in the following section.

Persistence for the configuration is supported in the data tier via Java Persistence API (JPA), which interfaces a MySQL database.

When the modeling is finished, the business logic component performs post configuration checks and calls an XML generator, to transform the configuration data into XML input to the simulator. The external simulator component is described well in (Walla et al., 2012b). After getting the input from the VME, the simulator communicates back to the VME via standard output and a specialized communication protocol based on sockets. The simulation data component is responsible to receive the simulator output and pass it one level up, where the data is associated back again with the model in the BL component, from which the simulator input generation has been performed.

Later on, the result visualization component presents to the user the simulation data in a graspable and analyzable form. All components of the VME are clearly separated, which allows for reuse and ease of modifications.

BUSINESS LOGIC

As described in the section on the system architecture, the business logic component is a central component for the system, meaning that all other modules communicate to it and use functions provided by it. Among the BL component functionality most of the domain oriented functions are concentrated. These domain oriented functions are discussed next.

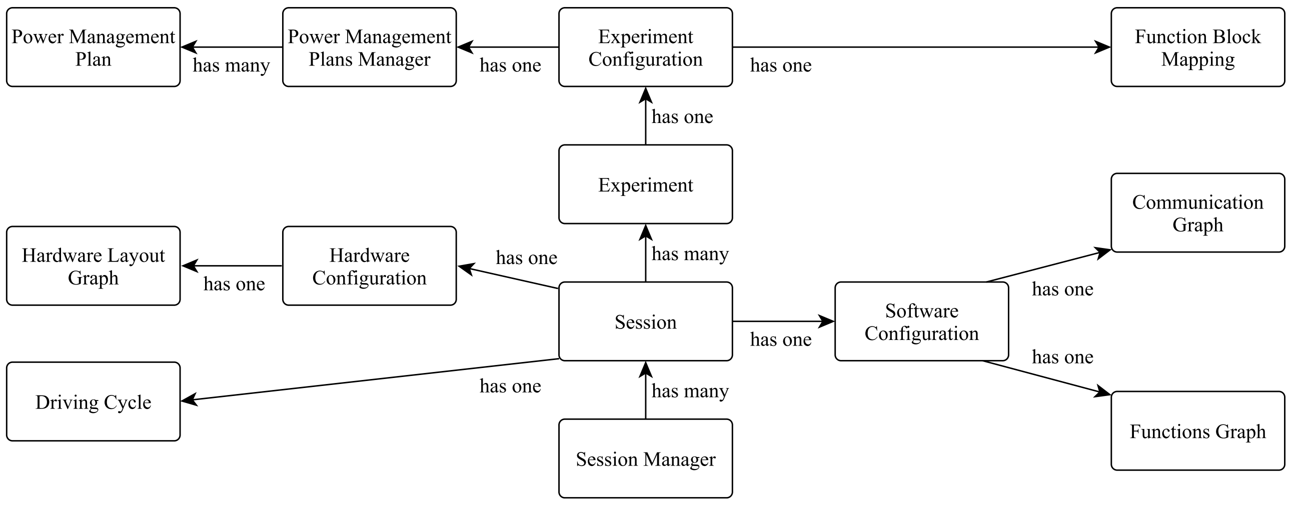


Figure 4: The Data Structure of the Business Logic Component

Experiment Cloning
To support the reuse of configuration parts, some of the parts are stored in the Session object, and only the parts specific for one particular experiment are stored in the Experiment object.

When experimenting with different power management strategies, and different functional block mappings, it is handy to be able to reuse the pure hardware and software configuration parts. This is performed by placing exactly this information into the Experiment objects, which can be many in one session.

In order to practically benefit from the information sharing in the Session object among several Experiment objects, one should create several experiments, which differ. The difference, however, between two experiments, can be "minor", e.g. one functional block remapped, and the power management plans (PMPs) modified accordingly.

After the first experiment has been configured, the second and further experiments, can be created as a clone of the existing one. Starting from the Experiment object in the structure shown in figure 4, all objects allow to create a clone of themselves.

The task of cloning starts to be less trivial, when it comes to cloning a PMP, which is a BL object containing a graph together with the JGraphX visual model of it. Cloning of such a structure must preserve the coherence of the new graph clone (BL model) and its JGraphX model newly acquired clone. The corresponding isomorphism is computed during the cloning.

Correct by Construction Principle
Manipulating on the BL objects, the user has to maintain the configuration in a correct state. Correctness here is taken in a most general, or weak, sense, which does not include the configuration completeness, and rather means mechanical consistency of the model.

The BL component takes care, that after each operation starting from a correct state, the user ends up in a new state which is also correct (in a sense as above). The system architecture is designed in such a way, that more complex routines to support correctness (in a stronger sense) can be integrated.

All operations on the data model within the BL component provide information on their success or failure, and automations are performed to keep the correct state.

As an example here we can consider a deletion of a functional block. It should not be possible to delete a functional block and to keep all edges leading to it in the visual representation. Furthermore, the deleted functional block has to be removed from the software-to-hardware mapping, and all other parts referencing it. The BL component takes care of the deletion, performing routine clean-up operations.

After the deletion of a functional block from the configuration, one of the PMPs, as a graph, can turn to be disjoint, which is logically incorrect (unreachable nodes appear). Post configuration checks are implemented in the BL component to filter out this and similar situations.

Post-Configuration Checks Infrastructure
The weakness of the correct by construction approach gives birth to a need for the post-configuration checks. Trivial examples of a not valid configuration would be a not complete configuration, where the user does not specify some vital configuration details, e.g. newly added functional blocks not mapped on the hardware components or PMPs. The post-configuration checks are implemented in each of the configuration objects and allow to test the object by itself, as well as the valid state of its descendants as on figure 4.

In the area of configuration validation and verification, the current system can be significantly improved, running checks dynamically (in the time of modeling) as well as making the checks more complete and complex. A good example of a useful checks could be PMP timing checks, as a check for schedulability, with all timing properties preservation.

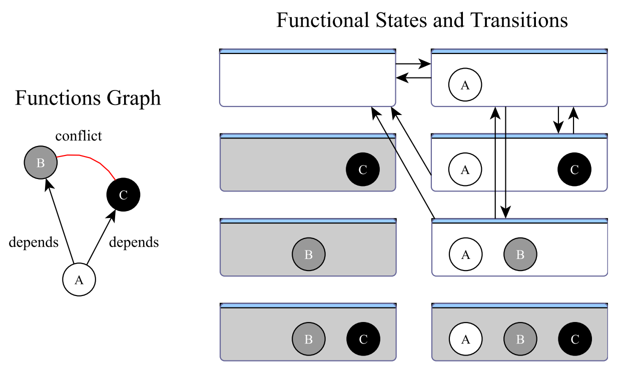


Figure 5: Functional State Modeling

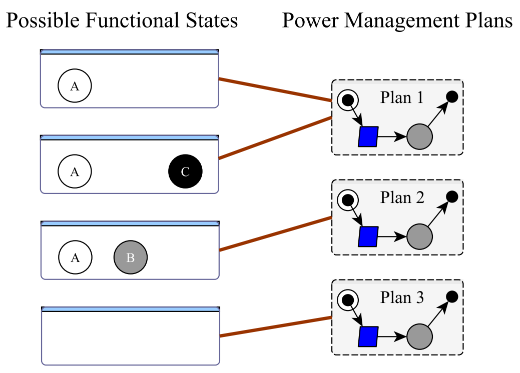


Figure 6: Power Management Plan Matching

Functional States Generation, Power Management Plans Matching

Here the term *function* means a feature of the system being modeled. Applying to the automotive domain examples could be: ignition ready, engine running, parking sensor working, wind screen wipers cleaning and similar. A function is defined as a set of functional blocks, supporting its operation. In the related work, common concepts are also denoted as feature (Metzger, 2004), or service interaction (Broy, 2005).

The interrelation of functions is shown figure 5. On the left, a so-called functions graph (FG) is shown. Nodes in the graph correspond to system functions. A function *B* can have a dependency on another function *A* (an edge from *A* to *B*), which means that *B* can only be activated (at a run time) after *A* has already been activated. Two functions can be denoted as conflicting, when they can not be active together at the same time.

The modeled system (vehicle) can have different states of operation. A set of functions, activated together at some point of time, we name here a functional state (FS). Figure 5 shows 8 functional states in the right part as rectangles. If there are N functions in the system, the amount of the FSs, in general, can be 2^N. The amount of transitions can have then the order of $2^N * (2^N - 1)$. Later in this work we call FSs and their transitions Functional State Graph (FSG).

Of the 8 functional states depicted in figure 5, only 4 stay to be valid (rectangles with the white background) due to the conflicts and dependencies defined by the FG on the left-hand side. Among the valid FSs, 8 transitions exist, which corresponds to switching particular functions on or off.

A full experiment configuration must include all possible FSs. For every FS the configuration has to specify transition conditions (under which circumstances the system reaches the FS) as well as a PMP for each of the ECUs. These procedures are automated in E/E-MaSiF.

For the user of E/E-MaSiF, the task to model the FSs and match them to corresponding PMPs is simplified to providing the functions graph, as in figure 5.

The FSG generation starts from the empty FS. Then all the functions are activated one by one, respecting dependencies and conflicts. New FSs are acquired, and the transitions to them are collected. Each of the FSs then participates again in the process of adding new functions. Set equivalent FSs are merged, combining the respective transition sets. When no more functions are available to extend the FSs, one by one the functions are switched back off from each FS, generating new transitions. The FSs merging stage is repeated after each FS generation step.

After the FSs and transitions are generated, the matching of the PMPs is performed (figure 6) for each ECU. The aim of this procedure is to identify, which PMP an ECU has to run, when the system enters a given FS.

For each of the ECUs a set of user-composed PMPs is specified together with the generated FSG. For each FS a set of supporting PMPs are defined, as PMPs containing all the FS required by the functional blocks. The *minimal* PMP of all matching PMPs is assigned to be the PMP to activate, when the system enters a given FS.

The minimality criteria can vary, depending on the user preferences. Currently, it means simply the lowest number of functional blocks present in each PMP.

The user of the system is verbosely informed, if the matching can not be performed due to lack compatible artifacts for specific FS.

SIMULATION VISUALIZATION

Once the user has modeled a concrete instance which passes all checks of the business layer, the model is translated to a simulation specification using XML. This specification contains all generated model properties like concrete functional states and matched power management plans. The simulator runs this specification and yields power and performance values for the architecture at hand.

Concrete model instances can be analyzed and visualized with the tool. Different specifications can be compared and visualized in a standardized way. Figure 7 depicts the aggregated consumptions per ECU and per ECU Component Type for an exemplary system specification. It shows on the left hand-side, that within the current experiment, a quarter of the total system's consumption is

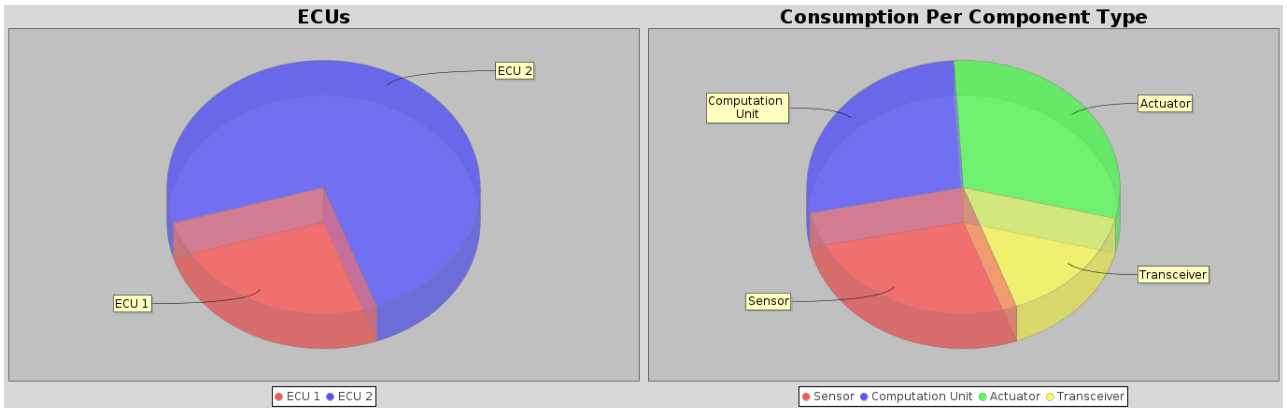


Figure 7: Overall Power Consumption per ECU and per ECU Component Type for one Experiment

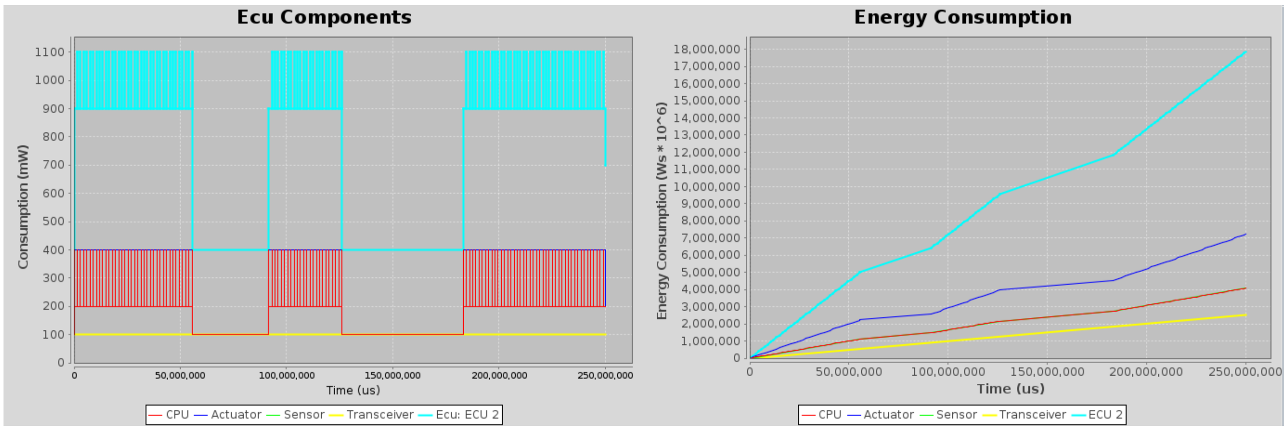


Figure 8: Instantaneous Power Consumption and Accumulated Energy Consumption per ECU Component of an ECU

amounted to ECU 1 while three quarters are amounted to ECU 2. On the right side, the total consumption during the experiment is amounted to the type of components. It is visible that in the example at hand, all component types except the Transceivers roughly amount for an equal share of the total consumption.

Selecting a specific ECU, the user can visualize detailed consumptions. This type of chart is depicted in figure 8. The charts are showing the dynamics of the system. On the left hand-side, the instantaneous power consumption of the selected ECU are given, while on the right hand-side, the accumulated energy consumption can be analyzed.

The possibility to open, maintain, and visualize different experiments at the same time enables the user to explore the design space and to evaluate different specifications of complex automotive systems.

CONCLUSION

This paper presented a new modeling and simulation framework for the evaluation of automotive embedded systems. It supports engineers in an early phase of the development by providing simulation-based information about the power consumption and performance of future E/E architectures.

The business logic of the framework provides basic consistency checks and assists the user through automated generation of some parts of the model. Furthermore, it administrates the storing, retrieving and cloning of evaluation sessions.

Simulating different variants in an early stage can help the developer understand the interplay of different vehicle functions and their power management and energy consumption.

Future work includes an OpenGL extension to the visualization of the simulation-based results. The user will be able to understand and to analyze the behavior of the vehicle in respect to a certain power management strategy, e.g. temporary switching of unused ECUs, in a 3D environment.

ACKNOWLEDGMENT

This work is part of an interdisciplinary research project of the Technische Universitaet Muenchen and BMW Group Research and Technology. It is organized under the CAR@TUM initiative, where the decades longstanding cooperation between the Technische Universitaet Muenchen and the BMW Group is intensified and restructured.

REFERENCES

Barthels, A., Ruf, F., Walla, G., Fröschl, J., Michel, H., and Baumgarten, U. (2011). A model for sequence based power management in cyber physical systems. *Information and Communication on Technology for the Fight against Global Warming*, pages 87–101.

Behrmann, G., David, A., and Larsen, K. (2004). A tutorial on uppaal. *Formal methods for the design of real-time systems*, pages 33–35.

Broy, M. (2005). Service-oriented systems engineering: Specification and design of services and layered architectures. *Engineering Theories of Software Intensive Systems*.

Fuchs, M., Scheer, P., and Grzemba, A. (2010). Selektiver teilnetzbetrieb im fahrzeug: Eine realisierung für den can-bus und adaption auf andere bussysteme. *GMM-Fachbericht-AmE 2010-Automotive meets Electronics*.

Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., and Ernst, R. (2005). System level performance analysis-the symta/s approach. In *Computers and Digital Techniques, IEE Proceedings-*, volume 152, pages 148–166. IET.

Hillenbrand, M. and Muller-Glaser, K. (2009). An approach to supply simulations of the functional environment of ecus for hardware-in-the-loop test systems based on ee-architectures conform to autosar. In *Rapid System Prototyping, 2009. RSP'09. IEEE/IFIP International Symposium on*, pages 188–195. IEEE.

Hölzl, F. and Feilkas, M. (2011). Autofocus 3-a scientific tool prototype for model-based development of component-based, reactive, distributed systems. *Model-Based Engineering of Embedded Real-Time Systems*, pages 317–322.

Metzger, A. (2004). Feature interactions in embedded control systems. *Computer Networks*, 45(5):625–644.

Samuel, S., Austin, L., and Morrey, D. (2002). Automotive test drive cycles for emission measurement and real-world emission levels-a review. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 216(7):555–564.

Schmutzler, C., Kruger, A., Schuster, F., and Simons, M. (2010). Energy efficiency in automotive networks: Assessment and concepts. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 232–240. IEEE.

Vector Informatik GmbH (2013). Preevision: Model-based electric/electronic development from architecture design to series-production readiness. *http://www.vector.com*.

Walla, G., Barthels, A., Ruf, F., Dörfel, R., Michel, H., Fröschl, J., Baumgarten, U., Herzog, H., Stechele, W., and Herkersdorf, A. (2012a). Aspects of function partitioning in respect to power management. *2nd International Energy Efficient Vehicles Conference (EEVC), Dresden, Germany*.

Walla, G., Gabriel, D., Barthels, A., Ruf, F., Michel, H., and Herkersdorf, A. (2012b). Ite-sim: A simulator and power evaluation framework for electric/electronic architectures. *The 8th IEEE Vehicle Power and Propulsion Conference, Seoul, Korea,*.

AUTHOR BIOGRAPHIES

GREGOR WALLA is a research staff member at the Institute for Integrated Systems at the Technische Universitaet Muenchen. He received the Dipl.-Ing. degree in Systems of Information and Multimedia Technology from the Friedrich-Alexander-Universitaet Erlangen-Nuernberg, Germany, in 2010.

ZAUR MOLOTNIKOV is a graduate student of the faculty of Informatics at the Technische Universitaet Muenchen, Master of Science Informatics program. He received a Univ. Dipl. in Applied Mathematics and Informatics at Southern Federal University, Rostov-on-Don, Russia, in 2009.

ANDREAS BARTHELS is a research staff member at the Institute for Informatics at the Technische Universitaet Muenchen, where he also received a Master's degree in 2009. Previously he received a Bachelor's degree in Informatics and Mathematics from Heinrich-Heine-Universität Düsseldorf in 2006.

HANS-ULRICH MICHEL received a Dipl.-Phys. degree in Physics from the Technical University in Darmstadt, Germany. After joining the BMW Group he has been responsible for the development of aftersales E/E subsystem products. After changing to BMW Group Research and Technology, he represented BMW in standardisation activities like the OSGi consortium, where he was a board member and chair of the vehicle expert group. Currently he is working on various research projects in the area of advanced telematic concepts with a special focus on the in-vehicle architecture.

WALTER STECHELE is an Academic Director at the Institute for Integrated Systems at the Technische Universitaet Muenchen. He received the Dipl.-Ing. and Dr.-Ing. degrees in Electrical Engineering from the Technische Universitaet Muenchen, Germany, in 1983 and 1988, respectively. In 1990 he joined the Kontron Elektronik GmbH in Germany, where he was responsible for the ASIC and PCB design department.

ANDREAS HERKERSDORF is a Full Professor and Chair of the Institute for Integrated Systems at the Technische Universitaet Muenchen. He joined the IBM Zurich Research Laboratory as a PhD student in 1988. In 1991, he became a Research Staff Member in the Communications Systems department of the IBM Zurich and in 2000 manager of the network processor hardware group. He received the Dipl.-Ing. degree from the Technische Universitaet Muenchen in 1987 and the Dr. techn. degree from the ETH Zurich (Swiss Federal Institute of Technology), Switzerland, in 1991, both in Electrical Engineering.