Electric Circuit- and Wiring Harness-Aware Behavioral Simulation of Model-Based E/E-Architectures at System Level

Harald Bucher, Jürgen Becker
Institute for Information Processing Technologies (ITIV)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{bucher, becker}@kit.edu

***Abstract*—To cope with the rising complexity of automotive electric/electronic architectures (EEA), model-based development at system level is well-established and typically realized in architecture description languages (ADLs) and high-level tools. In this paper, we extend a previously developed approach for automated cross-domain simulation synthesis of model-based EEA descriptions enabling system-level evaluation by a behavioral specification layer. The key contributions of this work are modeling extensions applied to a state-of-the-art EEA ADL to refine specified behavior during synthesis with electric circuits including wiring harness details modeled at the hardware layer. Preliminary experiments show that the novel combination of quantization- and SPICE-based synthesized circuit simulation, conducted in a discrete-event manner and applied to a buck converter, a typical device in an automotive EEA, increases simulation efficiency up to a factor of 2.0 compared to other state-of-the-art tools while preserving accuracy. Finally, another example EEA hardware network, modeling the dynamic current consumption of an Electric Power Steering actuator, applied to a realistic vehicle topology model demonstrates the impact of wiring harness refinements.**

***Index Terms*—Automotive, E/E-Architectures, ADL, Wiring Harness, MBSE, QSS, Modeling and Simulation, SPICE, Ptolemy II, PREEvision**

I. Introduction

Nowadays the EEA of modern vehicles are a distributed system of up to 100 Electronic Control Units (ECUs), sensors and actuators communicating over various bus systems and gateways though there is a shift towards hierarchical and centralized architectures [1], [2]. The latest innovations like driver assistance systems and the trend towards autonomous driving are the main reasons for this development. To cope with that complexity at system level, model-based architecture description languages (ADLs) and tools have been established in recent years such as the EAST-ADL [3], EEA-ADL [4] (realized in the tool PREEvision [5]) and Vehicle Systems Architect [6] which are compliant to the AUTOSAR [7] standard methodology. The ADLs and tools each provide sophisticated modeling capabilities for several aspects of an EEA such as requirements, functional network, hardware/software architecture and their mappings to each other. However, most of the ADLs and system level tools offer purely static modeling. The major drawback therefore is the lack of *executable* behavior specification integrated in the EEA model which is a fundamental property to perform system evaluation by means of high-level simulation in an early design stage [8] and to provide a functional behavior model independent of the running platform in order to transfer it to other vehicle product lines without changes ("separation of concerns") [9]. Consequently, to perform system level simulation of EEA, there need to be a linkage of the executable behavior specification to lower layer aspects [10] e.g., the deployment of functions to ECUs and the utilized network protocol between them or even ECU internal components.

Particularly, an ECU, sensor or actuator typically not only consist of one or more micro-controllers or comparable processing units executing some software functions but also hardware modules containing electric circuits doing, for instance, input/output signal conditioning for the processing units or actuators [11]. Its impact on the specified abstract behavior within the ECU and even on subsequent ECUs of the EEA therefore needs to be evaluated as early as possible to detect possible behavioral deviations at system level. Moreover, another very important part in an EEA is the wiring harness including its ground network. Due to high currents of several components within an EEA, they can induce a significant impact on voltage stability e.g., voltage drops on supply pins of hardware components. Supply voltages below a certain level may lead to several failures such as malfunctioning and ECU resets [12]. This is especially critical for safety-related ECUs. Therefore, the impact of several current consumers and the wiring harness on conventional or supplying connections also needs to be considered as early as possible in terms of the EEA system simulation model. PREEvision, for instance, provides capabilities to analyze the static current consumption of an EEA but, due to its static ADL modeling nature, cannot consider the dynamic current profile of components. Also the wiring harness and its ground network are neglected during analysis. [5, Sec. 10.3] Thus, the former and the latter can only be considered by additional simulation capabilities.

A novel approach was proposed in [13] to address the mentioned challenges by means of a cross-domain Ptolemy II [14] (PtII) simulation model synthesis out of an EEA system-level model and is described in more detail in Sec. III-A. However, ECU internal components in terms of electric circuits and especially the wiring harness between hardware components are not yet regarded. Against this background, this paper delivers the following contributions:

1) Modeling extensions applied to the state-of-the-art EEA-ADL regarding electric circuit refinement of logical signals specified at the logical architecture layer as well as static and behaviorally specified dynamic current consumers.
2) Automatic synthesis of a unified simulation model combining a behavioral specification refined by ECU internal electric circuits and especially wiring harness modeling refinements including its ground network.
3) Discrete-event-based simulation of the electric circuits in a single model by combining quantization-based integration methods with SPICE [15] based circuit simulation.
4) Decoupling of the abstract behavior and the refinements by using aspect-oriented simulation.
5) Preliminary experiments demonstrating simulation accuracy and efficiency compared to other state-of-the-art tools and the proof-of-concept of wiring harness impacts using a realistic vehicle topology model.

Note that the electric circuit refined models are not intended to replace

doi: 10.1109/SysEng.2018.8544434

detailed small-signal circuit simulation with expert tools, but rather to support EEA system-level evaluation at an early design stage by large-signal behavior of circuits interwoven with behavioral specifications.

II. Background

A. Quantized-State System Methods

In contrast to classical numerical integration solvers where time is discretized, quantized-state system (QSS) methods quantize the state trajectory. Regarding an ordinary differential equation (ODE) system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$ where $\mathbf{x}(t)$ is the state vector and $\mathbf{u}(t)$ a known input vector. Classical variable step-size solvers like Runge-Kutta determine sample times at which the sample values are computed for *all* states in the model [16].

The first-order QSS method [17] approximates the ODE equation with the *quantized state vector* $\mathbf{q}(t)$ resulting in $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{u}(t), t)$. A so called *hysteretic quantization function* connects each quantized state variable $q_i(t)$ and its corresponding state variable $x_i(t)$ by

$$q_i(t) = \begin{cases} x_i(t) & \text{if } \left|x_i(t) - q_i(t^-)\right| \geq \Delta Q_i, \\ q_i(t^-) & \text{otherwise.} \end{cases} \quad (1)$$

where ΔQ_i is called *quantum*. This means that $q_i(t)$ only changes its value when it differs from $x_i(t)$ by more than the quantum and immediately after the change $q_i(t) = x_i(t)$ applies. From Eq. (1) it follows that $\mathbf{q}(t)$ is piecewise constant and assuming that $\mathbf{u}(t)$ is also piecewise constant it follows that $\mathbf{x}(t)$ follows piecewise linear trajectories [17]. This shows an important advantage compared to classical time-step solvers in that each q_i is updated individually and *asynchronously* at different instants of time dependent on their specified quantum and the speed of crossing it. Since $\mathbf{x}(t)$ is piecewise linear, the time instants at which the quantum is reached can be computed analytically without iterations. Therefore QSS methods are particularly suited to efficiently simulate across discontinuities [16].

Second- and third-order QSS methods (QSS2 [18] and QSS3 [19]) share the same properties and advantages of QSS1 but exhibit quantized states following piecewise linear and piecewise quadratic trajectories respectively. They offer better accuracy without significantly increasing the number of quantization events. Because the QSS methods can rarely handle stiff systems (simultaneous slow and fast dynamics), the QSS family was extended by *Linear Implicit QSS* methods (LIQSS1-3) [20] since QSS1-QSS3 methods exhibit high frequency oscillations with most stiff systems. Another important property of QSS methods is their easy integration into discrete-event (DE) simulation engines, where changes of the quantized states are represented by a sequence of discrete events thus enabling hybrid simulation of continuous dynamics and DE behavior in a single model. In a DE model a QSS is split into n so called *static functions* and n *quantized integrators* for each quantized state and optionally m event sources which represent the input vector $\mathbf{u}$(t) [16]. Within higher-order QSS methods the events do not carry the actual value only but also the slope (QSS2) and the second derivative (QSS3) of the current trajectory segment.

B. Modeling and Simulation with Ptolemy II

PtII is an open-source modeling and simulation framework for heterogeneous embedded systems with focus on concurrent components as well as the deterministic use and composition of heterogeneous *Models of Computation* (MoC). PtII follows an actor-oriented approach. Actors are components that execute concurrently and communicate with each other via ports. They can be atomic or composite. Atomic actors cannot be refined whereas composites enable hierarchical nesting of actors. The semantics for the execution of and communication between actors is governed by a specific MoC. The MoC within the model of a composite actor is realized by a component called *Director*. Distinct directors can be composed hierarchically in a single model at each level of the hierarchy. There are a variety of MoCs supported by PtII including DE, which is especially suitable to model complex and large-scale discrete systems like hardware architectures or communication networks. The DE MoC in PtII supports a sound semantics incorporating a deterministic model of time called *superdense time*. Besides DE, there exist several other MoCs like continuous time, various data flow MoCs for signal processing and finite state machines. A *concrete syntax* to represent models in PtII is the XML-based MoML (Modeling Mark-up Language) and can be edited in the GUI Vergil. [21]

Recently, QSS methods were integrated into PtII realized by several components [22]: (1) a *QSS Director* is an extended version of the DE Director ensuring the correct processing of events in time stamp order and provides default parameters for all QSS Integrators. (2) *QSS Integrators* realize the quantized integrators as described in Sec. II-A and provide parameters e.g., for setting the QSS solver or quanta individually. (3) a special token called *Smooth Token* is introduced, which is an extended version of a primitive double valued token. It is a discrete event that can carry not only the actual real value, but also one or more (currently up to three) derivative values. Thus, piecewise smooth signals like piecewise linear or quadratic trajectories can be exchanged between actors that can either use the real value only or also the derivative values. This is a prerequisite for higher-order QSS integrators and actors that serve as static functions (see Sec. II-A). Downstream actors that receive smooth tokens but require a value between two consecutive smooth token events - which indicate a significant change in the piecewise smooth signal - extrapolate the latest smooth token to the current time using its derivatives. Currently QSS1-QSS3 methods are supported by PtII but are still experimental.

III. Automated EEA Simulation Model Synthesis

In this section we first give a brief introduction to the already developed baseline methodology [13] shown in Fig. 1. Afterward, we present in detail the modeling extensions applied to the EEA-ADL to extend the baseline approach by refined electric circuit simulation in a DE manner streamlined with the specified behavior.

A. Cross-Domain Simulation Synthesis: Baseline Methodology

The starting point for cross-domain simulation of model-based EEA is a data model, which captures all relevant information about an EEA. To handle the complexity of such architectures, this is done in a model-based fashion e.g., with sophisticated EEA architecture description languages like the EEA-ADL which is realized in the architecture design and analysis tool PREEvision. In PREEvision the EEA is split into seven abstraction layers (without the green extensions) each having its own viewpoint covering a specific engineering domain as shown in Fig. 1. They stretch from the requirements specification, over the functional network (Logical Architecture (LA)), the System Software Architecture and the Hardware Topology of ECUs and buses down to detailed models of ECU internal Electric Circuits (EC), Wiring Harness (WH) artifacts such as wires and splices as well as geometrical Topology information like installation locations and branch-offs. Cross-layer mappings establish a seamless traceability across all layers. The main idea proposed in [13] is to use the already available EEA data model information in order to synthesize an integrated, unified and executable high-level simulation model which is capable of linking the functional network specified at the LA layer with lower level implementation details e.g., the network communication

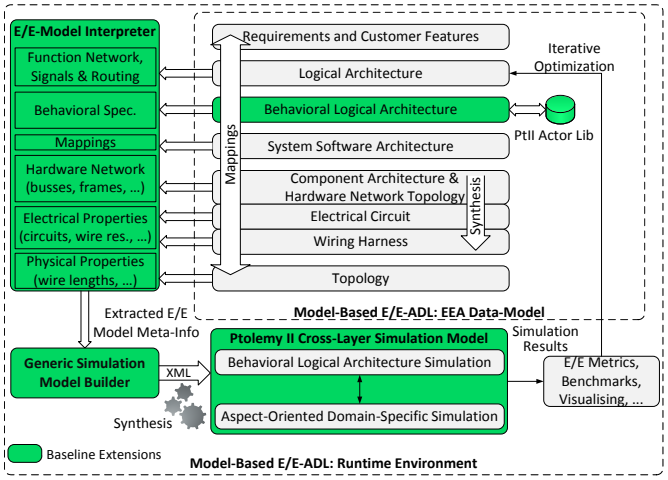

Fig. 1: Overview of the baseline methodology for cross-domain EEA simulation model synthesis [13]

of the underlying hardware topology and execution time of functions. The necessary extensions made to achieve this are shown in Fig. 1.

Because of the lack of support of existing ADLs and tools, including PREEvision, for modeling *executable* behavior integrated within the EEA system model, a new layer called *Behavioral Logical Architecture* (BLA) was introduced that refines the static logical blocks (what the system does) with detailed behavior (how the abstract functions work) by reusing actors from the *PtII Actor Library*. The latter is imported as a separate logical block type library into PREEvision where the types are used to instantiate actors at the BLA layer. Additional mappings of LA atomic logical functions to BLA building blocks containing the actors as well as between their port prototypes are introduced. The latter enable automatic connection of the BLA top-level blocks thus their connections need not be modeled. In combination with mappings from the LA layer to lower layers they provide the connection of the behavioral blocks of the BLA to domain-specific information at lower layers enabling the early stage cross-domain simulation.

The *E/E-Model Interpreter* extracts all necessary information from the relevant layers of the underlying EEA data model including the mappings as well as signal routing information. It serves as a front-end to interpret the underlying EEA data model and stores the meta-info belonging to the functional network in an internal database. The *Generic Simulation Model Builder* serves as a back-end, it maps EEA meta-model artifacts to the PtII meta-model and synthesizes the unified cross-domain simulation model in terms of a PtII MoML description using the specified LA/BLA behavioral network and the extracted meta-info. Note that due to the layered approach, the target behavioral model is not limited to PtII but can be extended to another target language like MATLAB/Simulink by importing its library and implementing an appropriate back-end. The synthesized *PtII Cross-Domain Simulation Model* can either be conducted within PREEvision or externally by means of the generated MoML file and is twofold: (1) it contains the behavioral simulation specified at the BLA layer and (2) it performs the aforementioned lower layer domain-specific and non-functional simulation. The latter simulations are performed in an aspect-oriented way along with the behavioral simulation. More details can be found in [13].

B. Modeling Extensions for EC- and WH-Aware Simulation

In the baseline approach, executable behavior is fully specified at the BLA layer by means of the PtII actors. ECU internal components

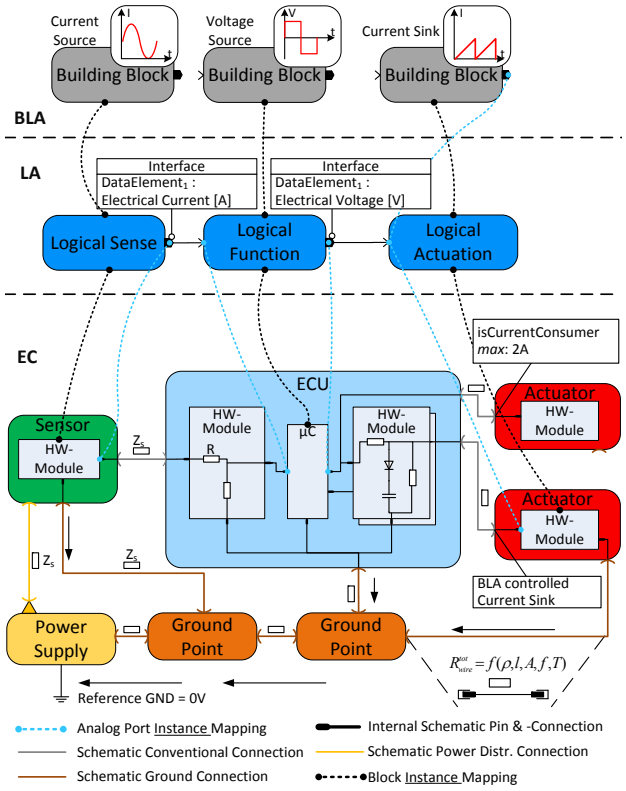

Fig. 2: EEA modeling extensions for electric circuit refined simulation model synthesis

at the EC layer and wiring harness artifacts at the WH layer were not yet regarded during synthesis. An example EC layer is shown in Fig. 2. There, a simple voltage divider before the input pin of a micro-controller (where behavior is mapped to) and a half-way rectifier after an output pin are shown. In the following we refer to these components as *internal HW components*. To identify which signals of a LA logical function i.e., BLA building block actors, shall include electric circuits we introduce two EEA modeling extensions: (1) special LA port interface data types and (2) a new set of cross-layer port mappings.

1) Logical Port Interface Data Types: A logical port instantiates a port prototype and is attached to an interface containing data elements. Data elements are the actual information sent by the logical ports via signals. Here we specify that a port having exactly one data element either of type *Electrical Voltage* or *Electrical Current* is identified as an analog port. An actor output at the BLA layer thus is either interpreted as a voltage or current source during synthesis respectively. This is illustrated at the top of Fig. 2.

2) Analog Port Mappings: Each logical function is typically mapped to exactly one internal HW component, which abstracts the functional behavior by a BLA building block. Individual logical ports can now serve as an actor-oriented electrical source stimulating connected successor electrical circuits or can serve as an electrical sink e.g., if no electric circuits are modeled in a receiving hardware module. The profile of the latter is then specified by a behavioral actor network (cf. the Actuator at the bottom in Fig 2). To identify which intermediate electrical circuits need to be considered between a logical sender port and one or more connected logical receiving ports, a source and a destination pin have to be specified at the EC layer serving as start and end point of the electric circuit parsing

during synthesis. This is necessary, since a processing unit provides several pins and in case the logical function is mapped to more than one internal HW component. The mapping is - similar to the port mappings between LA and BLA layer - accomplished via *Analog Port Mappings* between exactly one logical port instance and exactly one corresponding *Internal Schematic Pin*. Mapped pins between two logical functions can be spread across ECUs. The mappings are illustrated by the dotted blue lines in Fig. 2.

3) Current Sinks: As mentioned, a logical receiver port can serve as an electrical sink. Two types of sinks are introduced: *static* and *dynamic* current consumers. At the EC layer there already exists an attribute called *isCurrentConsumer* annotated on each schematic pin of a HW component like ECU or actuator. It provides three parameters *Minimum*, *Maximum* and *Typical* to model a static current consumer. Besides the analog port mapping we reuse this attribute as a necessary modeling requirement to identify a logical receiver port as a current consumer, if the corresponding schematic pin is connected to the mapped analog internal schematic pin. At the same time, a set *isCurrentConsumer* attribute signals an end point in the electric circuit parsing. In this way, we can include further HW components which are either not fully modeled with internal devices to achieve a valid circuit or do not have a corresponding mapped logical function along the path. The *isCurrentConsumer* attribute modeling is illustrated by the top text box at the Actuator in Fig. 2. To distinguish a dynamic BLA controlled current sink from a static one, a logical receiver port at the LA layer has - besides the internal schematic pin mapping - an additional analog port mapping to a logical sender port of the corresponding mapped BLA building block (cf. Fig. 2). This is necessary since PtII follows an actor-oriented approach where each event or token has to be communicated via an output port to its receivers. The BLA building block serving as a dynamic current consumer can specify an arbitrary current profile by its internal actor network. The dynamic current consumer case is illustrated by the Actuator and text box at the bottom of Fig. 2.

4) Wiring Harness: So far, we have regarded the internal HW components' electrical circuits as refinement of the abstract behavior and neglected the *Conventional Connections* and *Power Distribution Connections* between them as well as their *Ground Connections*. At the EC layer they are abstracted through plain connections. However, at the WH layer they are represented by a set of wires, splices, wiring harness connectors etc. and the schematic pins are refined by wire and header pins. This is shown by the example of a ground connection in Fig. 2. As introduced in Sec. I the voltage stability at supply pins of ECUs influenced by the wiring harness is very critical. In the EEA model each wire has a specific *wire type* which offers electrical and physical properties like the specific resistance ρ and cross-section A. Via cross-layer mappings to the topology, the total length l_{tot} of a wire is obtained which consists of several *Topology Segments* each having an individual length. If this information is available along the path between electric circuit refined logical functions, wire resistances are calculated with the formula $R^{tot}_{wire} = \rho \frac{l_{tot}}{A}$ and included in the netlist.

The circuits as well as the current consumers discussed so far were assumed to have a ground reference equals to $0V$. However, HW components have ground connections to a specific *Ground Point* in the chassis of the vehicle body and therefore cause additional voltage drops until the reference ground of the power supplying component is reached. This can cause, for instance, undesired DC offsets in the signals of the modeled electric circuits which could significantly influence the overall circuit behavior. For these reasons, we include the ground network in our electric circuit simulations to enable this

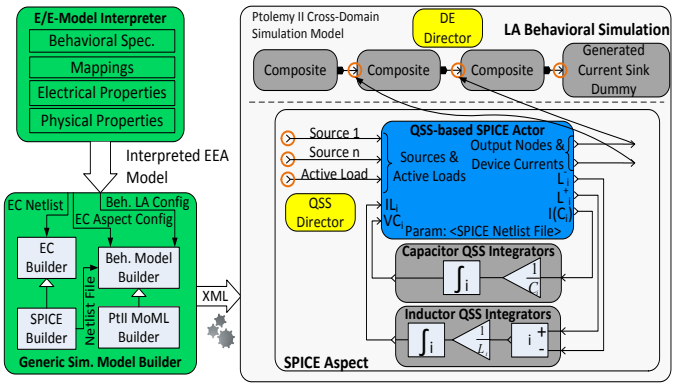

Fig. 3: Electric circuit aspect synthesis

analysis as early as possible. For the sake of simplicity, only the power supply connection to the sensor is shown. However, in order to get a valid circuit, each HW component must have a connection to a supplying component. Otherwise, the ground network cannot be considered. Note that a wire actually exhibits a frequency dependent wire impedance Z_S rather than a resistance only. In future work we plan to provide further generic modeling extensions to support wire impedance at least for supplying connections, since several simplifications can be made concerning frequency dependency and the RLC wire model without a significant accuracy loss [12].

C. Aspect-Oriented Electric Circuit Synthesis

The foundations of the EC synthesis are the components *E/E-Model Interpreter*, the *Generic Simulation Model Builder* and the aspect-oriented approach of the baseline methodology. In Fig. 3 the further realized components are shown. In the baseline approach, aspects are also used to refine behavior via so called communication aspects. A token arriving at a receiving port which is decorated with such an aspect is rerouted to a sub-model which processes/delays these tokens and finally sends them back to the originating rerouted receiver port. The structure of the behavioral model is not touched. The same approach is used for electric circuit simulation, where tokens from analog ports are rerouted to a composite aspect performing the electric circuit simulation. The rerouting is illustrated by the circles around the analog receiving ports in Fig. 3.

1) E/E-Model Interpreter: The E/E-Model Interpreter aggregates the behavior specified at the BLA layer with the introduced modeling extensions from the EC, WH and Topology layer to synthesize the PtII model. It fulfills three main tasks: (1) Parsing the modeled electric circuits as well as its connecting wiring harness and generating an EC netlist; (2) generating a *BLA configuration* data structure used to synthesize the behavioral simulation decorated with the electric circuit aspect refinement and (3) generating a *EC Aspect configuration* data structure used to synthesize the electric circuit aspect itself. To include the ground network into the EC netlist, the following methodology is applied: For each external ground schematic pin of a HW component, the shortest path over *Ground Point* components and its connecting ground wires is traced back until its energy source (cf. Fig. 2). The latter is assumed to be the reference ground of $0V$.

2) Generic Simulation Model Builder: The behavioral simulation model is synthesized directly as modeled at the BLA layer by the engineer. This is shown in Fig. 3 where each PtII composite corresponds to a BLA building block. To increase productivity and the reuse of BLA building blocks, analog sender ports which should serve as dynamic current consumers and its corresponding receiving BLA

building blocks are generated automatically based on the provided configuration data (cf. Fig. 3). The top-level director of the PtII model is DE, since the used aspects rely on timed events. Consequently, the user is responsible for modeling a sufficiently high sampled signal or a sequence of smooth tokens specifying piecewise smooth signals at analog output ports in order to achieve accurate input signals to the electric circuit aspect simulations. The electric circuit aspect is synthesized by means of the generated EC netlist and the EC Aspect configuration data. The *EC Builder* shown in Fig. 3 provides generic interfaces to build a specific electric circuit netlist. In this work, we implemented a SPICE [15] netlist builder which outputs a SPICE-based netlist file used by the behavioral model builder finally generating the SPICE aspect. The latter is illustrated together with its main components in Fig. 3 and is outlined in the following.

3) SPICE Aspect: The key components of this composite aspect are the *QSS-based SPICE Actor*, composites containing *QSS Integrators* for capacitors and inductors as stored in the EC Aspect configuration and the *QSS Director* governing the DE-based SPICE simulation of the EC netlist. The basic simulation principle is to perform a transient analysis of the EC netlist stimulated by the voltage/current sources and current sinks as modeled in the composites. Every time an event is sent to the aspect or an event is received by the QSS integrators, the QSS-based SPICE actor calculates a single transient analysis step by solving the DC operating point (DCOP) of the EC netlist until the SPICE engine converges i.e., the results of two successive DCOPs deviate by less than a parametrizable tolerance.

a) QSS-based SPICE Actor: The SPICE engine used for the new actor is JSpice [23], an extensible SPICE-based circuit simulator implemented in Java and focusing on rapid prototyping by using simple linear companion models of non-linear circuit devices like MOSFETs and diodes. The fact that the simulator is implemented in Java enables the easy integration of circuit simulation into PtII as a separate actor with only slight changes to the engine's interface and without the need to perform co-simulation with external tools. The main parameter of the actor is the *SPICE Netlist File* to be simulated as shown in Fig. 3. The circuit system equation formulation of JSpice is based on the *Modified Nodal Analysis* (MNA), the commonly used method in many circuit simulators. Each circuit device's contribution to the MNA called *stamp* is done after [24]. Dependent on the parametrized QSS solver, smooth tokens with no derivative (QSS1) up to the second derivative (QSS3) are produced at the node voltage/device current outputs of the actor.

b) QSS Integrators: In transient analysis, capacitors and inductors are typically piecewise linearized around a DCOP through an equivalent voltage and current source with an equivalent series and parallel resistance respectively. The equivalent circuit and its values are obtained by classical time-discretized numerical integration of their $I-V$ relationship. For this reason, the system matrix stamps of the equivalent linear elements depend on the current time-step of the applied time-discretized numerical solver [24]. To eliminate the time-step dependency, we extract the frequency dependent parts $I_C(t) = C\frac{dV_C(t)}{dt}$ and $V_L(t) = L\frac{dI_L(t)}{dt}$ of the MNA equation and solve them by QSS integrators individually. This approach has several advantages: only a frequency independent linear equation system is to be solved by the SPICE actor every time a source or QSS integrator generates an event. Thus there is no need to have knowledge about the present signal frequencies in the model, but the SPICE actor is triggered asynchronously in a DE manner. Finally, a distinct director for solving algebraic loops as proposed in [22] is not necessary, since they are implicitly broken by the SPICE actor by only producing an event if the DCOP is converged.

IV. Experimental Simulation Results

This section presents and discusses the simulation results obtained by our approach. Two scenarios are pursued: (1) evaluation of the accuracy and run-time efficiency of the automatically synthesized model by simulating a buck converter, a DC/DC voltage switching circuit converting an input DC voltage to a lower one; (2) simulation of electric circuit refined behavior regarding wiring harness and current consumers. All experiments were conducted on a 64 bit Windows 8.1 machine running an Intel Core i5-4300U at $2.49GHz$ and providing $12GB$ RAM and a SSD hard drive.

A. Benchmark Setup

Regarding the evaluation of the accuracy and run-time performance, the buck converter circuit simulation is compared to several state-of-the-art tools namely *MATLAB/Simscape Power Systems R2015b* [25], *PowerDEVS* [26] and *LTSpice* [27]. MATLAB Simscape Power Systems is widely used in industry for modeling and simulation of control systems and electrical power systems using sophisticated classical ODE solvers; PowerDEVS is a tool focusing on hybrid system simulation of power systems implementing the complete family of QSS solvers; LTSpice is a SPICE derivative with focus on power switching regulators like DC/DC converters.

Concerning MATLAB and PowerDEVS we reused their buck converter demo example and adapted them to meet our applied parameters. For LTSpice, we used our generated SPICE netlist and adapted it by simply searching and replacing the QSS related voltage/current sources with corresponding capacitors and inductors. To evaluate the accuracy we simulated two reference trajectories of the capacitor and inductor state variables using the MATLAB ode23tb solver and the PowerDEVS LIQSS3 solver with a tight relative error tolerance of $1e^{-9}$. The simulated trajectories exhibited unequally spaced sample points where the LIQSS3 solution produced about 55000 points and the MATLAB solution about 11800 points. In order to calculate the error between the two reference trajectories, we linearly interpolated the MATLAB trajectories and compared them at the time points chosen for LIQSS3 trajectories. Since the resulting relative errors between the reference trajectories were in the order of 10^{-5} and 10^{-6} for the inductor and capacitor respectively, we only report the error of the remaining solutions compared to the LIQSS3 solution. For all buck converter simulations we applied the following parameters: (1) QSS3 was used for all QSS integrators in PowerDEVS and PtII. (2) The relative error tolerance as well as the absolute and relative quanta ΔQ_i related to QSS solutions are set to $1e^{-3}$. (3) A simulation time of $0.01s$ is conducted and the simulation run-time is reported while suppressing all plots and outputs. (4) The mean absolute error and the relative error compared to the reference are reported by the following formulas:

$$MAE = \frac{1}{|\mathbf{t_s}|}\sum_{i=1}^{|\mathbf{t_s}|} abs\left(y_i^{ref} - y_i^{sim}\right) \; and$$
$$NMAE = \frac{MAE}{mean\left(abs\left(\mathbf{y^{ref}}\right)\right)} \tag{2}$$

B. Buck Converter

In order to simulate switched circuits, we added models for both externally (voltage-controlled) and internally controlled switches into JSpice by implementing the models proposed in [28]. The latter provide better run-times in the order of one magnitude compared to conventional SPICE switch models. For both types of switches, a switch is represented by a resistor R_S with a low value R_{ON} or high value R_{OFF} depending on its state. Diodes are implemented

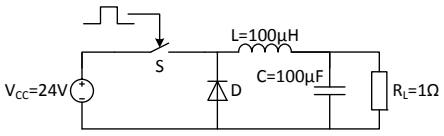

Fig. 4: Buck converter circuit

TABLE I: Additional buck converter circuit parameters

Parameter	Value
$R_{ON}^{Diode} = R_{ON}^{Switch}$	$10e^{-6}\Omega$
$R_{OFF}^{Diode} = R_{OFF}^{Switch}$	$10e^{6}\Omega$
Switching Frequency	$10kHz$
V_{ctrl}^{Switch}	$1V$
V_{thres}^{Switch}	$0.5V$

as internal switches according to [29]. The circuit's schematic and device values are shown in Fig. 4. Further applied parameters are summarized in Tab. I. Power switching circuits exhibit high frequency discontinuities as well as stiff system properties and thus are a good choice to benchmark the QSS performance [29], [30]. The DC voltage source and the switch control signal were modeled according to Sec. III-B in a BLA building block with two analog ports spread over two internal HW components and a BLA block receiving the output voltage with its HW counterpart within one ECU.

The relative error of the inductor current is plotted in Fig. 5. Tab. II compares results obtained with the different tools and the run-time necessary to simulate them. Regarding the LTSpice solution one can observe that the error of the capacitor trajectory significantly deviates from the other solutions, while the inductor trajectory is acceptable. PowerDEVS provides the best accuracy with one order of magnitude less than all other solutions coming at the cost of a higher run-time. Despite we can observe a few outliers in the relative error of the inductor current of our synthesized model, overall it meets the specified relative quantum of 10^{-3} though, having a mean relative error one order of magnitude below the specified quantum like the MATLAB and LTSpice solution. Especially in the beginning, their errors reach its maximum due to the high current gradient. However, for rapid prototyping purposes at system level, our resulting error is still acceptable while simulation performance is enhanced.

Considering the comparison of the run-time performance we report two types of simulation run-time for our approach: SPICE simulation with and without aspect. The latter means that the analog composite, SPICE actor and QSS integrators are directly governed by a QSS Director and located at the same hierarchy level without rerouting to an aspect composite. This provides a fairer comparison to the other tools, since they do not offer an aspect mechanism. Though we also

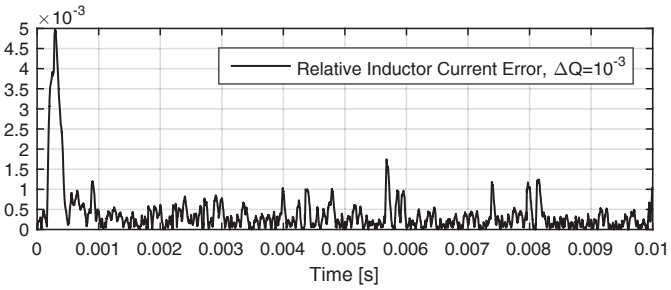

Fig. 5: Relative error of the inductor trajectory of the synthesized buck converter

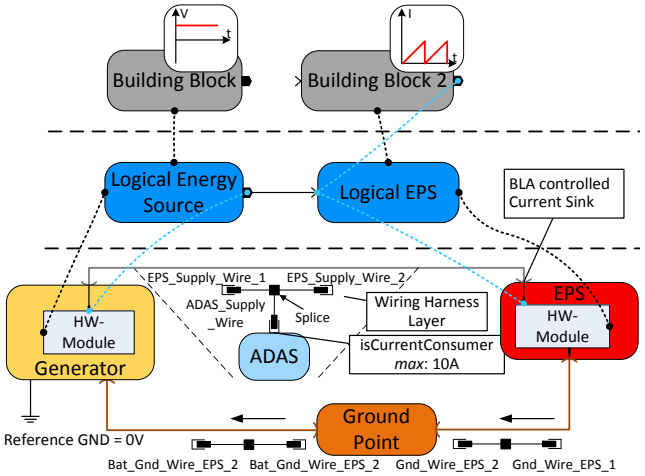

Fig. 6: Wiring harness and dynamic current consumer setup of an EPS actuator network

compare the aspect solution, since it is an essential part of our work. An interesting result is that the MATLAB and PowerDEVS solutions perform nearly equally fast, despite PowerDEVS uses a QSS3 solver. This can be explained by the fact that QSS3 actually is an explicit non-stiff solver while the buck converter exhibits stiff properties, where LIQSS methods would perform much more efficiently [29]. However, even without LIQSS our synthesized PtII model performs faster than MATLAB and PowerDEVS by a factor of 2.0 without SPICE aspect and about 1.19 and 1.17 respectively with SPICE aspect while preserving accuracy. This shows the efficiency of our QSS- and SPICE-based approach by means of the asynchronous calculation of the SPICE circuit while embedding the static functions of the QSS integrators into it. This significantly reduces the number of actors and therefore event traffic between actors and the coordinating director. The aspect reduces performance, since it infers additional rerouting overhead but increases modularity and reuse by separation of concerns.

C. Wiring Harness and Current Consumers

Given the correct and accurate simulation of our synthesized electric circuit refined model shown in Sec. IV-B, we want to demonstrate the correct simulation of wiring harness parts as well as static and dynamic BLA controlled current consumers. The considered EEA model is shown in Fig. 6. It models an *Electric Power Steering* (EPS) actuator supplied over a conventional $12V$ generator assuming an internal resistance of $1m\Omega$. Both are connected to the same ground point. At the LA layer, two logical functions are modeled which are mapped to their corresponding HW counterparts. Here, the *Logical EPS* consumes a specific current profile which is deposited and realized within the *EPS Current Profile* building block at the BLA layer. Similarly, the generator is modeled as a constant supply voltage of $12V$. Analog port mappings are performed according to Sec. III-B. In addition, at the EC layer the wiring harness refinements with the represented wire names of the conventional and ground connections are shown. The ADAS ECU is modeled as a static current consumer sinking $10A$ and not mapped to a logical function. For the sake of simplicity, all wires are assigned to the same wire type *AWG4* (American Wire Gauge).

The described model is embedded into a complete demo EEA model within PREEvision, which provides a realistic vehicle topology model to which the hardware and wiring harness artifacts are mapped. Since the topology layer model is large and complex, it is not shown in Fig. 6 for space reasons. After mapping our EPS sub-model extensions to

TABLE II: Simulation results comparison of the buck converter circuit

Integration Method	MAE		NMAE		Simulation run-time [ms]
	V(C) [V]	I(L) [A]	V(C)	I(L)	
PtII QSS3 ($\Delta Q_i = 10^{-3}$)	$5.55e^{-3}$	$4.51e^{-3}$	$4.69e^{-4}$	$3.75e^{-4}$	211 (w/o SPICE aspect)
	$5.55e^{-3}$	$4.51e^{-3}$	$4.69e^{-4}$	$3.75e^{-4}$	355 (with SPICE aspect)
PowerDEVS QSS3 ($\Delta Q_i = 10^{-3}$)	$4.45e^{-4}$	$6.64e^{-4}$	$3.77e^{-5}$	$5.61e^{-5}$	416
MATLAB ode23tb ($rel.\,tol. = 10^{-3}$)	$3.69e^{-3}$	$4.07e^{-3}$	$3.15e^{-4}$	$3.42e^{-4}$	422
LTSpice mod. trap. ($rel.\,tol. = 10^{-3}$)	$8.46e^{-3}$	$2.95e^{-3}$	$7.13e^{-4}$	$2.47e^{-4}$	205

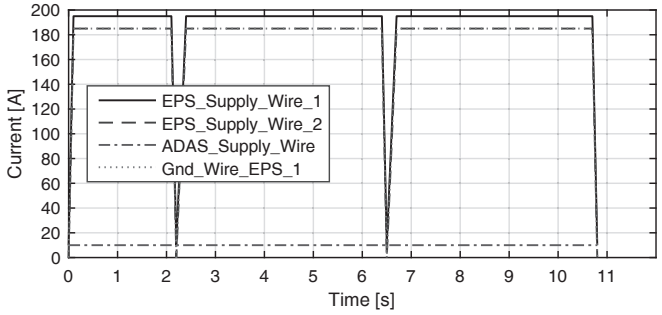

Fig. 7: Currents through EPS supply and ground wires

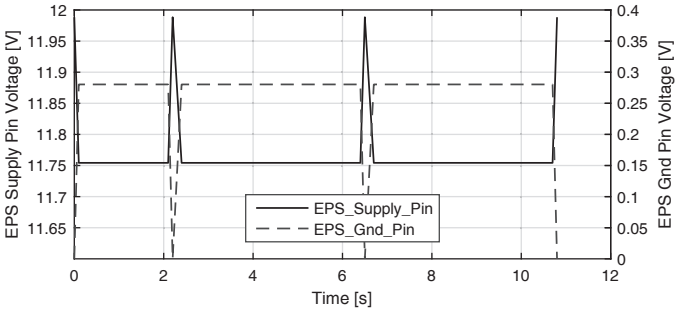

Fig. 8: Voltage at the EPS supply pin resulting from generator and wire voltage drops and at the EPS ground pin

the topology layer, the *Wiring Harness Router* of PREEvision [5, Sec. 10.4] was used to obtain the shortest wire paths and thus realistic wire lengths for our simulation synthesis. The obtained wire resistances lie in the range of several hundred $\mu\Omega$. Wire and connector pin resistances are neglected but can also be incorporated during synthesis from the EEA model. The simulation results of the synthesized model are shown in Fig. 7 and Fig. 8. The current through *EPS_Supply_Wire_2* represents the EPS current profile as specified at the BLA layer. Among others, the profile is gathered from an EPS specification document regarding the vehicle power network and is based on real-world driving maneuvers and measurements of a German OEM. The profile represents a parking scenario. This raised a requirement of a max. current of $185A$. *ADAS_Supply_Wire* illustrates the correctly synthesized static current consumption of $10A$ of the ADAS ECU related to reference ground $0V$ (since no ground network is modeled). Fig. 8 shows the voltage drops caused by the wiring harness at the EPS supply pin and the voltage at the ground pin. For the applied wire type AWG4 ($A = 21.15mm^2$) and since the signal frequency is below $11Hz$, a R_{DC} wire model is appropriate according to [12] without a significant accuracy loss.

V. Conclusions and Future Work

In this work, we presented a set of modeling extensions applied to the state-of-the-art EEA-ADL and its EEA design and analysis tool PREEvision. They connect abstract behavioral blocks mapped to ECUs or similar hardware components with internal electric circuits. Especially, a unique feature is the consideration of wiring harness modeling refinements between ECUs including their ground network as well as static and dynamic current consumers. The presented modeling extension concepts can be transferred to other EEA ADLs provided that they offer a comparable level of detail for the necessary electrical and physical properties such as hardware devices, wiring harness, wire or pin resistances. The synthesized simulation is capable of performing DE-based hybrid simulation of abstract behavior specified at the logical layer enhanced by the hardware layer details. Preliminary experiments show that for simple examples the novel combination of QSS methods and the QSS-based SPICE actor performs up to a factor of 2.0 faster than comparable tools like MATLAB and PowerDEVS while preserving accuracy. Even with rerouting overhead induced by aspects it performs about 16% and 15% faster respectively while offering separation of concerns. LTSpice performs slightly faster than our approach without aspects which can be explained by its highly optimized solver for power switching circuits. However, it is limited to circuit simulation and does not offer integration into system-level modeling with aspect-oriented decoupling like in our approach. Nevertheless, the generated SPICE netlist additionally increases reuse and modularity since it can be exchanged with any SPICE compatible circuit simulator with only slight adaptations as done with LTSpice. Furthermore, an EC Builder software component offers abstract interfaces to implement other EC target simulation models which could be investigated.

Finally, an example logical and hardware network of an EPS actuator, applied to a realistic vehicle topology model, demonstrated the consideration of the wiring harness and current consumers. Its impact in terms of voltage drops on supplying ECU pins and voltages at ground pins as well as currents through individual wires was shown, which is important for voltage stability and wire dimensioning analysis respectively. Of course the power distribution within a real EEA is much more complex containing e.g., fuse-relay-boxes and further power conditioning and balancing components. However, the simulations demonstrated the proof-of-concept of the basic functionality to include the wiring harness and arbitrary current consumers. More complex models can be incorporated and evaluated based on this foundation.

Future work includes the implementation of LIQSS3 methods to increase simulation efficiency and stability for more stiff systems, modeling extensions to consider wire impedance, and the evaluation of more complex EEA models regarding scalability and simulation efficiency.

VI. Related Work

A review of system modeling tools can be found in [8]. The tool *System Desk* from dSpace can model software architectures and functional networks. Simulation and verification of functions and component diagrams has to be performed offline and it does not

offer linkage to lower layer EEA artifacts. *Volcano Vehicle Systems Architect* from Mentor is similar to PREEvision and offers design and management of hardware and software systems according to AUTOSAR and EAST-ADL and provides cross-layer artifacts mapping capabilities. However, requirements and lower layers such as the wiring harness or topology are not captured and thus is not suitable for our approach. In addition, functional behavior simulation has to be exported to another external tool format. *Rhapsody Designer* from IBM is a well-established tool to design UML-based system models and provides simulation and code generation capabilities for activity and state diagrams but only captures functional behavior.

Works dealing with simulation model synthesis from EAST-ADL models are, for instance, [31]–[33]. In all approaches, the *FunctionalBehavior* or *HWComponentType* blocks from the EAST-ADL specification are referenced to externally deposited (outside the EAST-ADL system model) simulation model counterparts and are not embedded within the EEA model description. The target simulation models are either SystemC-based or Simulink models combined with Functional Mock-up Units (FMUs). The import of the latter is also supported by PtII [22]. Particularly, the authors in [32] leverage SystemC-AMS enabling analog/mixed-signal modeling and simulation. Unfortunately, its capabilities cannot be fully exploited since the EAST-ADL does not provide the level of detail to synthesize hardware internal circuits and wiring harness details such as splices or wire resistances. Moreover, SystemC-AMS does not incorporate QSS solver methods compared to our approach. Further references and more detailed descriptions of the mentioned works can be found in [13].

Concerning systems engineering in actor-oriented models, the work in [34] briefly reviews existing aspect-oriented approaches and provides a detailed evaluation of various aspects applied to a robotic swarm. It includes network fabrics, robot dynamics, abstract CPU models, fault models and error handling, contract modeling and logging aspects. Aspects for electric circuit simulation, especially combined with QSS methods, are not mentioned. Our approach of simulating electric circuits with QSS methods was inspired by Lee et al. [22]. In our work, however, circuits are generated automatically as SPICE netlist from naturally non-causal representations within the EEA model and not in terms of an actor network. The encapsulation in a composite aspect also differentiates our approach.

DE simulation of hybrid systems in general and electric circuits specifically using QSS methods has been studied in recent works such as [26], [29]. The authors in [30] generate DE models for the tool *PowerDEVS* [26] from Modelica models. In [35] the authors developed a stand-alone QSS solver from a Modelica language subset called μ-Modelica significantly improving the run-time efficiency of QSS simulations. However, the textual, equation-based μ-Modelica modeling is neither applicable to EEA ADLs nor to our actor-oriented behavioral specification.

References

[1] C. Buckl *et al.*, "The software car: Building ict architectures for future electric vehicles," in *Electric Vehicle Conference (IEVC), 2012 IEEE International*, Mar. 2012, pp. 1–8.

[2] W. Stolz *et al.*, "Domain control units - the solution for future e/e architectures?" in *SAE Technical Paper 2010-01-0686*. SAE International, Apr. 2010.

[3] EAST-ADL Association. (2016) East-adl domain model specification v2.1.12. [online; http://www.east-adl.info/Specification, last accessed 19.07.2018].

[4] J. Matheis, "Abstraktionsebenenübergreifende Darstellung von Elektrik/Elektronik-Architekturen in Kraftfahrzeugen zur Ableitung von Sicherheitszielen nach ISO 26262." Ph.D. dissertation, 2010.

[5] Vector Informatik GmbH, *PREEvision Version 8.5 Manual*, 2017.

[6] Mentor Graphics. (2018) Volcano vehicle systems architect. [online; https://www.mentor.com/products/vnd/, last accessed 07.01.2018.

[7] AUTOSAR Consortium. (2018) Autosar 4.3 (automotive open system architecture) specifications. [online; https://www.autosar.org, last accessed 07.01.2018].

[8] P. Waszecki *et al.*, "How to engineer tool-chains for automotive e/e architectures?" *SIGBED Rev.*, vol. 10, no. 4, pp. 6–15, Dec. 2013.

[9] A. Sangiovanni-Vincentelli *et al.*, "Embedded system design for automotive applications," *Computer*, vol. 40, no. 10, pp. 42–51, 2007.

[10] P. Derler *et al.*, "Modeling cyber-physical systems," *Proceedings of the IEEE (special issue on CPS)*, vol. 100, no. 1, pp. 13 – 28, Jan. 2012.

[11] H. Eki *et al.*, *Fail-operational EPS by distributed architecture*. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, pp. 421–441.

[12] R. Gehring *et al.*, "Modeling of the automotive 14 v power net for voltage stability analysis," in *2009 IEEE Vehicle Power and Propulsion Conference*, Sep. 2009, pp. 71–77.

[13] H. Bucher *et al.*, "An integrated approach enabling cross-domain simulation of model-based e/e-architectures," in *SAE Technical Paper 2017-01-0006*. SAE International, 2017.

[14] J. Eker *et al.*, "Taming heterogeneity - the Ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[15] P. W. Tuinenga, *SPICE: A Guide to Circuit Simulation and Analysis Using PSpice*. Prentice Hall, 1995.

[16] E. Kofman, "Discrete event simulation of hybrid systems," *SIAM Journal on Scientific Computing*, vol. 25, no. 5, pp. 1771–1797, 2004.

[17] E. Kofman *et al.*, "Quantized-state systems: A devs approach for continuous system simulation," *Trans. Soc. Comput. Simul. Int.*, vol. 18, no. 3, pp. 123–132, Sep. 2001.

[18] E. Kofman, "A second-order approximation for devs simulation of continuous systems," *Simulation*, vol. 78, no. 2, pp. 76–89, 2002.

[19] ——, "A third order discrete event simulation method for continuous system simulation," *Latin American Applied Research*, vol. 36, no. 2, pp. 101–108, 2006.

[20] G. Migoni *et al.*, "Linearly implicit quantization-based integration methods for stiff ordinary differential equations," *Simulation Modelling Practice and Theory*, vol. 35, no. Supplement C, pp. 118 – 136, 2013.

[21] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.

[22] E. A. Lee *et al.*, "Modeling and simulating cyber-physical systems using cyphysim," in *2015 International Conference on Embedded Software*, Oct. 2015, pp. 115–124.

[23] T. Molter. (2017) Jspice. [online; https://github.com/knowm/jspice, last accessed 02.01.2018].

[24] J. Vlach *et al.*, *Computer Methods for Circuit Analysis and Design*, 2nd ed. Springer, 1993.

[25] Mathworks. (2015) Simscape power systems r2015b. [online; https://www.mathworks.com/products/simpower.html, last accessed 04.01.2018].

[26] F. Bergero *et al.*, "Powerdevs: a tool for hybrid system modeling and real-time simulation," *SIMULATION*, vol. 87, no. 1-2, pp. 113–132, 2011.

[27] Linear Technology. (2017) Ltspice xvii. [online; http://www.linear.com/designtools/software/\#LTspice, last accessed 04.01.2018].

[28] V. Litovski *et al.*, "Ideal switch model cuts simulation time," *IEEE Circuits and Devices Magazine*, vol. 22, no. 4, pp. 16–22, 2006.

[29] G. Migoni *et al.*, "Quantization-based simulation of switched mode power supplies," *SIMULATION*, vol. 91, no. 4, pp. 320–336, 2015.

[30] X. Floros *et al.*, "Automated simulation of modelica models with qss methods - the discontinuous case -," in *8th International Modelica Conference*, Mar. 2011, pp. 657–667.

[31] G. Weiss *et al.*, "Approach for iterative validation of automotive embedded systems," in *Models 2010 ACES-MB Workshop Proceedings*, 2010, pp. 69–83.

[32] R. Weissnegger *et al.*, "Simulation-based verification of automotive safety-critical systems based on east-adl," *Procedia Computer Science*, vol. 83, pp. 245–252, 2016.

[33] R. Marinescu *et al.*, "Analyzing industrial architectural models by simulation and model-checking," in *Formal Techniques for Safety-Critical Systems*. Springer International Publishing, 2015, pp. 189–205.

[34] I. Akkaya *et al.*, "Systems engineering for industrial cyber-physical systems using aspects," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 997–1012, May 2016.

[35] J. Fernandez *et al.*, "A stand-alone quantized state system solver for continuous system simulation," *Simulation: Transactions of the Society for Modeling and Simulation International*, vol. 90, no. 7, pp. 782–799, 2014.