

## Cyber-Physical Co-Simulation Framework for Smart Cells in Scalable Battery Packs

SEBASTIAN STEINHORST, MATTHIAS KAUER, ARNE MEEUW,  
SWAMINATHAN NARAYANASWAMY, MARTIN LUKASIEWYCZ, TUM CREATE Limited  
SAMARJIT CHAKRABORTY, TU Munich

This paper introduces a Cyber-physical Co-Simulation Framework (CPCSF) for design and analysis of smart cells that enable scalable battery pack and Battery Management System (BMS) architectures. In contrast to conventional cells in battery packs, where all cells are monitored and controlled centrally, each *smart cell* is equipped with its own electronics in the form of a Cell Management Unit (CMU). The CMU maintains the cell in a safe and healthy operating state, while system-level battery management functions are performed by cooperation of the smart cells via communication. Here, the smart cells collaborate in a self-organizing fashion without a central controller instance. This enables maximum scalability and modularity, significantly simplifying integration of battery packs. However, for this emerging architecture, system-level design methodologies and tools have not been investigated yet. By contrast, components are developed individually and then manually tested in a hardware development platform. Consequently, the systematic design of the hardware/software architecture of smart cells requires a cyber-physical multi-level co-simulation of the network of smart cells which has to include all the components from the software, electronic, electric and electrochemical domains. This comprises distributed BMS algorithms running on the CMUs, the communication network, control circuitry, cell balancing hardware and battery cell behavior. For this purpose, we introduce a CPCSF which enables rapid design and analysis of smart cell hardware/software architectures. Our framework is then applied to investigate request-driven active cell balancing strategies that make use of the decentralized system architecture. In an exhaustive analysis on a realistic 21.6 kW h Electric Vehicle (EV) battery pack containing 96 smart cells in series, the CPCSF is able to simulate hundreds of balancing runs together with all system characteristics, using the proposed request-driven balancing strategies at highest accuracy within an overall time frame of several hours. Consequently, the presented CPCSF for the first time allows to quantitatively and qualitatively analyze the behavior of smart cell architectures for real-world applications.

CCS Concepts: •Computer systems organization → Embedded and cyber-physical systems; Distributed architectures; •Hardware → Batteries; •Computing methodologies → Discrete-event simulation;

Additional Key Words and Phrases: Smart Battery Cells, Battery Management, Co-simulation, Active Cell Balancing, Cell Balancing Strategy

### ACM Reference Format:

Sebastian Steinhorst, Matthias Kauer, Arne Meeuw, Swaminathan Narayanaswamy, Martin Lukasiewicz, and Samarjit Chakraborty, 2016. Cyber-Physical Co-Simulation Framework for Smart Cells in Scalable Battery Packs. *ACM Trans. Des. Autom. Electron. Syst.* 0, 0, Article 00 (2016), 26 pages.  
DOI: <http://dx.doi.org/10.1145/0000000.0000000>

---

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) program.

Author's addresses: Sebastian Steinhorst (corresponding author), Matthias Kauer, Arne Meeuw, Swaminathan Narayanaswamy, Martin Lukasiewicz, TUM CREATE Limited, Singapore; Samarjit Chakraborty, TU Munich, Germany.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2016 ACM. 1084-4309/2016/-ART00 \$15.00  
DOI: <http://dx.doi.org/10.1145/0000000.0000000>

## 1. INTRODUCTION

Electric Vehicles (EVs) and smart energy grids are entering the mass market. For the progress of these technologies, Electrical Energy Storages (EESs) in form of Lithium-Ion (Li-Ion) battery packs play a central role. They are gaining importance due to their favorable energy and power density dominating all other battery chemistries. Li-Ion battery packs cover a wide range of applications from a few watt-hours of energy storage capacity in small laptop computers to several megawatt-hours in large stationary EESs for backup supplies or peak demand/supply management of the highly volatile renewable energy sources.

Design of Li-Ion battery packs for a certain energy and power requirement comprises the choice of a cell type, which is then arranged in a series-parallel architecture tailored to the application. Despite all their benefits, including a negligible memory effect and long lifetime, Li-Ion battery cells are highly sensitive to certain operating parameters. Exceeding specified cell voltage or temperature thresholds causes irreversible damage to the cells which negatively influences their capacity and cycle lifetime up to a complete loss of function and, in the worst case, a thermal runaway which can lead to an explosion of the cell, possibly causing a chain reaction in a battery pack. Consequently, Battery Management Systems (BMSs) are a mandatory component of such battery packs. Their task is to keep all cells in the battery pack in a safe and healthy operating state by managing the flow of energy in and out of the battery pack while charging or discharging. Furthermore, the BMS controls the air or liquid cooling of the pack such that all parameters are constantly maintained in the specified range.

Traditionally, BMS architectures are designed in a centralized fashion such that a central master controller processes all information and manages a whole battery pack, often with slave controllers in a hierarchical setup [Brandl et al. 2012]. Design and integration of battery packs is a highly customized process and for each new battery pack, a huge hardware/software integration effort is required. Recently, there is a trend towards decentralized BMS architectures, which provide increased scalability and reduce the integration effort [Baronti et al. 2012; Otto et al. 2012; Steinhorst et al. 2014]. Towards this, a natural trend is to build in more intelligence into each cell by adding its own electronics in form of a CMU, creating a *smart cell* which monitors and manages itself. In a battery pack assembled from smart cells, the smart cells coordinate their actions using a communication interface and cooperate to provide pack-level battery management in a completely decentralized, self-organizing fashion.

However, designing the architectures and management techniques of such battery packs requires an appropriate simulation framework that combines the (i) continuous dynamics arising from the physical and electrochemical characteristics of the cells, and (ii) the discrete nature of the management algorithms running in software on the CMUs of each cell. Designing such a Cyber-physical Co-Simulation Framework (CPCSF) is challenging because of the multiple models required for the components of the architecture. These models significantly differ regarding their modeling approach and time constants required for an accurate system-level simulation. We address these challenges by combining models for all layers of the architecture in a tightly coupled but modular discrete event simulation framework that, for the first time, enables system-level simulation with high accuracy of large battery packs with decentralized battery management.

### 1.1. Contributions of this Paper

This paper introduces a CPCSF built upon validated models for all components of the smart cell architecture to enable fast system-level battery pack simulation of long-time behaviors such as active cell balancing. Active cell balancing is an emerging approach

to equalize the charge of individual cells in a battery pack efficiently by transferring charge between them. The requirement for the co-simulation framework is motivated from decentralized active balancing strategies which have been developed in the scope of this paper and which cannot be exhaustively analyzed with existing software tools or hardware development platforms. In addition to decentralized battery management and smart cell architectures, which are discussed in Section 2, the contributions of this paper are as follows.

- The smart cell architecture enables cooperative and communication-based decentralized battery management algorithms and influences the hardware/software co-design. Consequently, we present an approach to designing active cell balancing control algorithms for smart cells. In this context, we propose four request-driven balancing strategies in Section 3 that cannot be analyzed on system level with existing tools.
- To enable system-level analysis of smart cell architectures, we propose a CPCSF in Section 4. Here, efficient and accurate validated modeling approaches for battery cell characteristics, active cell balancing hardware behavior, BMS algorithms and the communication network to cover all layers of the smart cell architecture are developed and then combined in the CPCSF.
- We apply the proposed framework in Section 5 to analyze how the request-driven balancing strategies proposed in this paper can be simulated and evaluated in a long-term simulation of several hours system time, covering the complete behavior of a 21.6 kW h EV battery pack consisting of 96 cells in series.

Related work in the domain of battery management, cell balancing, component modeling and system-level BMS simulation is discussed in Section 6. Finally, conclusions are drawn in Section 7.

## 2. SCALABLE BATTERY MANAGEMENT WITH SMART CELL ARCHITECTURES

In contrast to traditional hierarchical or centralized BMS architectures [Brandl et al. 2012] where a master controller performs all control and monitoring tasks, smart battery cells manage themselves individually at cell level. Beyond that, they coordinate cooperative actions via a communication interface to perform pack-level tasks such as cell balancing or State-of-Charge (SoC) determination. This decentralized approach fundamentally changes the implementation of BMS algorithms which now need to be developed in a distributed fashion. Consequently, besides the hardware architecture discussed in the following, new algorithmic approaches such as those introduced in Section 3.2 will be required.

A smart cell provides the functionality to monitor and control itself, as well as to send and receive messages for organizing pack-level functions in cooperation with other smart cells. The CMU attached to the cell is powered by the cell itself. Figure 1 illustrates a battery pack formed by five smart cells consisting of battery cells and their dedicated CMUs. Here, the CMU comprises

- a Sensor and Balancing Module (SBM) to acquire the parameters (voltage, temperature, current) of the cell and perform cell balancing,
- computation capabilities in form of a microcontroller and
- a communication interface to exchange information between smart cells.

The amount of control circuitry to be embedded into the CMU depends on the choice which kind of cell balancing shall be supported.

Figure 2 shows a smart cell development platform for five smart cells which was used as an initial proof of concept in [Steinhorst et al. 2014]. While the feasibility of decentralized battery management has been successfully shown with this platform, it does not allow to develop and analyze system-level aspects for larger battery packs built

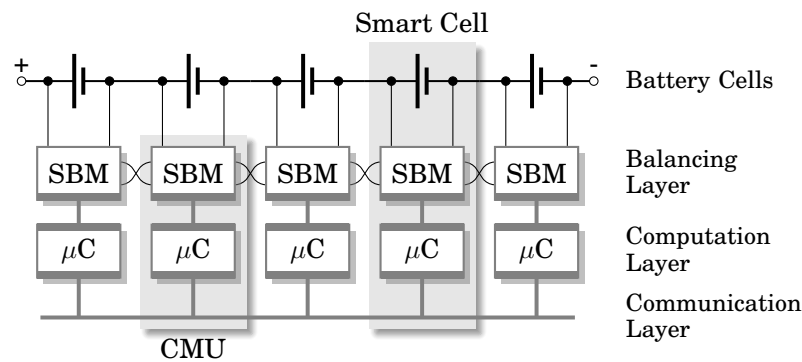


Fig. 1. Five smart cells forming a battery pack. Each smart cell consists of a battery cell and its dedicated Cell Management Unit (CMU). The CMU contains a Sensor and Balancing Module (SBM), a microcontroller and a communication interface, forming the balancing, computation and communication layers of the architecture.

from smart cells. Therefore, this development platform will be used in the remainder of this paper for model development and validation in order to enable a co-simulation framework that can answer questions for a full EV battery pack. EV battery packs usually contain 96 cells in series to achieve the required power output at feasible voltages and currents. Examples of properties to analyze are the charge transfer efficiency or required time for performing active cell balancing with the new request-driven strategies that will be introduced in the next section.

In the remainder of this section, the requirements and design characteristics of the SBM, the computational layer and the communication interface are summarized. Finally, a discussion regarding the possible integration of the components of the CMU into an Integrated Circuit (IC) in the form of a System-on-Chip is given.

### 2.1. Sensing and Balancing

The basic properties of a cell are its voltage and temperature at any point in time. Furthermore, beyond the pack current, the individual balancing currents into and out of the cell have to be monitored for SoC estimation. These measurements can be easily performed by an integrated multiplexed Analog-Digital Converter (ADC) that is connected to the cell terminals, to a shunt resistor or hall effect sensor for balancing current sensing and to a thermistor or resistance temperature detector.

For charge equalization, all smart cells that form a battery pack must have the same type of balancing capabilities. For passive cell balancing, where energy from cells with a higher SoC is dissipated until they reach the SoC of the cell with the lowest charge, the CMU must contain a switchable resistor that can individually dissipate energy stored in the cell. For active cell balancing, where charge is transferred between cells, a modular architecture consisting of homogeneous modules that can be integrated into each smart cell, such as the ones proposed in [Lukasiewicz et al. 2014], [Kauer et al. 2013] or [Kutkut 1998], are suitable. Details on active balancing will be discussed in Section 3.

### 2.2. Computation

As a core feature of the smart cell architecture, the CMU performs computational tasks to process local information from the sensors of the smart cell, as well as data received via the communication channel. Consequently, the CMU performs the management and control of its connected individual cell as well as contributes to the cooperative

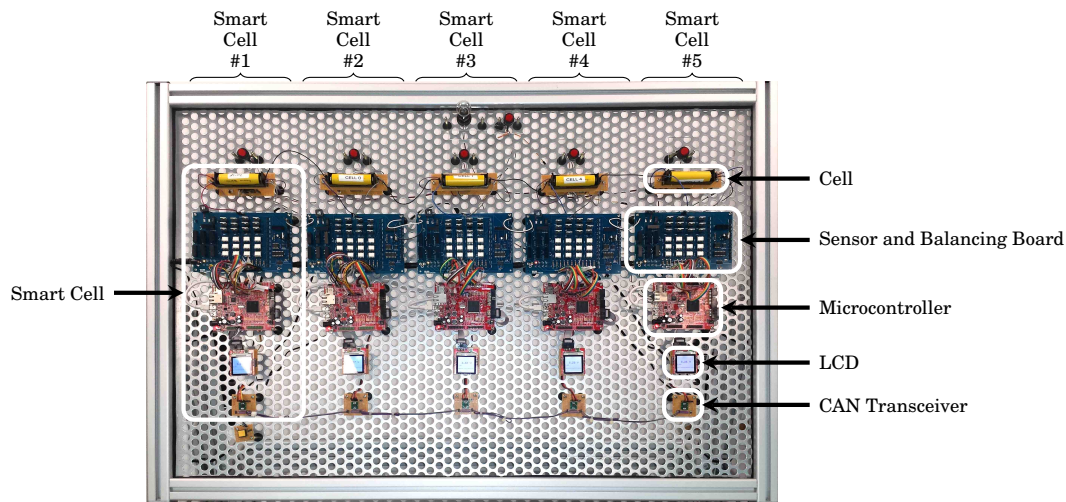


Fig. 2. Development platform from [Steinhorst et al. 2014] showing five smart cells with individual SBMs and microcontrollers which communicate via a CAN bus.

system-level functionality. Being powered directly by the cell, the computational core of the CMU requires low-power processing capabilities and efficient power management. In this context, it has to be considered that the computational performance has to satisfy the requirements of local calculation of the SoC and State-of-Health (SoH) of the cell with a sampling rate of the sensor data of up to 100 Hz. Furthermore, the communication with other smart cells has to be provided. Here, in certain high-activity periods such as performing active cell balancing during peak load current, the cell may have to receive and process hundreds of messages per second with status and control information from other cells when considering an EV battery pack where usually 96 smart cells would be connected in series.

### 2.3. Communication

The communication architecture chosen for the development platform is a wired CAN bus as this is a reliable and well-established standard. Here, the emphasis is on broadcast messages with only one smart cell transmitting to the bus at a time. Consequently, hardware message filtering has to be performed such that smart cells only have to process messages containing information which is relevant to them. Alternatively, a daisy chain topology could be considered. Here, broadcasts are expensive as messages have to be relayed across all nodes. Local communication with neighboring nodes can, by contrast, be performed concurrently, allowing for local parallel communication.

### 2.4. Integration into a System-on-Chip

The overall goal of the smart cell architecture is to come up with a single mixed-signal integrated circuit per cell, comprising the whole functionality of the CMU. Integration of the computational layer and the communication layer as well as most parts of the SBM is possible with a minimal footprint, low power consumption and cheap production costs. In case of passive cell balancing chosen as the mode of cell balancing, the complete CMU could be integrated into a single chip. A typical 32-bit microcontroller (AT32UC3A), which matches the computational requirements for the CMU, weighs less than two grams. While this hardware is added at cell level, the smart cell architecture reduces

the weight of a central controller and might also require less wiring for voltage and temperature sensing.

For modular active cell balancing architectures, one inductor or transformer would be required for each smart cell as will be discussed in detail in Section 3. Such components, however, cannot be efficiently integrated in a single chip together with the rest of the components of the CMU. Packaging an inductor together with the CMU chip on a small Printed Circuit Board (PCB) would still allow highly compact smart cells with a negligible volume and weight of less than 50 g added to the cell which weighs almost 2 kg per cell in state-of-the-art vehicles such as the BMW i3 battery pack with 60 A h prismatic cells. For further details, a discussion of suitable component choices for active balancing circuits can be found in [Narayanaswamy et al. 2014]. As conventional BMS components would be replaced by the CMU of the smart cells, the final volume and weight balance would be very similar to conventional solutions, but with the scalability and integration benefits coming from the smart cell architecture and the energy efficiency of active cell balancing.

### 3. ACTIVE BALANCING STRATEGIES FOR SMART CELLS

Considering the special characteristics of the smart cell architecture introduced in Section 2, in this section we will discuss the properties of active cell balancing that will be modeled in the CPCSF proposed in Section 4.

During charging and discharging of the series-connected cells in a battery pack, cells tend to, over time, have different capacities and resulting SoCs. This distribution is due to variations in manufacturing and operating temperature that influence the capacity of each individual cell and the variation grows with every charging or discharging of the cells. Furthermore, different self discharge rates of individual cells can cause imbalances even when the pack is not operated. As Li-Ion cells are very sensitive to minimum and maximum charge levels, certain thresholds for the SoC have to be maintained for every cell. If the SoC variation increases over time (i.e., no countermeasures are taken), the usable SoC of the battery pack decreases.

The cell with the lowest charge will require the whole discharging of the battery to be stopped once it reaches its lower discharging threshold. The same applies to charging where the charging process has to be stopped when the first cell reaches its upper SoC threshold. Therefore, cell balancing is performed to equalize the SoC of all cells in order to maximize the usable capacity of the battery pack. In the case of passive cell balancing, only the upper SoC threshold of cells is considered. Therefore, this approach is only applicable when charging the cells. In order to maximize the battery pack-SoC, the cells that have a higher individual SoC than others are discharged over a controlled resistor such that ideally all cells can reach their maximum SoC during charging.

When a battery pack reaches its lower SoC threshold, there may be, however, a huge number of cells in the battery pack that have a slightly higher SoC. Although there is still energy available in these cells, the pack cannot be further discharged. Active cell balancing architectures can utilize this remaining energy as charge can be transferred between cells. In contrast to passive cell balancing, where the SoC of cells can only be decreased, active cell balancing can increase the SoC of the battery pack.

The generation of Pulse Width Modulation (PWM) signals required for control of active cell balancing is a challenging task for a centralized BMS, as the signals have to be processed concurrently with high accuracy while other tasks are performed. As a result, passive balancing still dominates the market, despite its obvious deficits, due to its simpler implementation and significantly cheaper manufacturing cost. Smart cells, however, allow to generate the PWM signals directly at cell level individually for each active balancing circuit module which might, together with future cost reductions in active cell balancing hardware, make this approach economical.

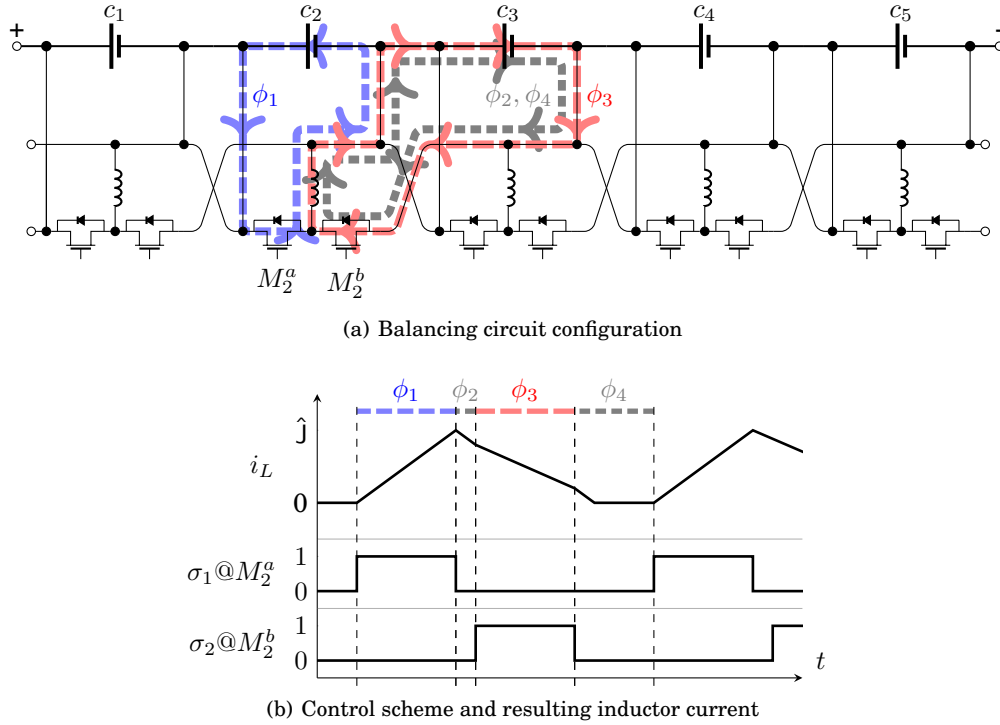


Fig. 3. Balancing architecture configuration for charge transfer between neighboring cells via an inductor. The transfer is carried out in four phases  $\phi_1$  to  $\phi_4$  where appropriate PWM signals  $\sigma_1$  and  $\sigma_2$  are applied to the MOSFETs.

Figure 3(a) shows a modular inductor-based active cell balancing hardware architecture that enables charge transfer between neighboring cells that is adapted from [Kutkut 1998]. It consists of two Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs) and one inductor per cell. The balancing process requires the control of some MOSFETs using PWM signals. Figure 3(b) illustrates the required periodically occurring phases  $\phi_1$  to  $\phi_4$  of the non-overlapping PWM signals which control the MOSFET switches. Here, in phase  $\phi_1$ , the inductor is charged from cell  $c_2$  by closing  $M_2^a$  and discharged into cell  $c_3$  in phases  $\phi_2$  to  $\phi_4$ . Phases  $\phi_2$  and  $\phi_4$  are created by the non-overlapping behavior of the PWMs, using the body-diodes in the MOSFETs to prevent short circuits when  $M_2^b$  is closed.

Charge transfer can be performed concurrently between several pairs of cells in a battery pack. However, depending on the balancing hardware architecture, certain restrictions regarding concurrent charge transfers can be present. In the architecture illustrated in Figure 3(a), the cells neighboring those performing a charge transfer should not participate in a charge transfer themselves concurrently for safety reasons. Consequently, a charge transfer between two cells  $c_i$  and  $c_{i+1}$  requires to also block cells  $c_{i-1}$  and  $c_{i+2}$  from engaging in a charge transfer themselves.

### 3.1. Optimization Criteria for Active Balancing

Active balancing enables the transfer of charge between cells until an equilibrium is reached in the battery pack. However, there needs to be an algorithmic strategy to determine the source and destination cells for charge transfers, as well as the order in which the transfers are performed. Furthermore, the strategy has to consider the

specific characteristics of the underlying balancing hardware architecture. The signals to control the balancing have to be generated such that the hardware can perform the charge transfer in a safe fashion. Short circuits or charged inductors without freewheeling path have to be avoided at all cost as these situations damage cells and balancing hardware and might even result in a hazardous thermal runaway of the cells. Approaches to formalize and verify the safety of switching patterns for active balancing circuits have been developed in [Baronti et al. 2014a; Lukasiwycz et al. 2014].

There are four main criteria characterizing the quality of a charge equalization process, directly influenced by the chosen strategy:

- (1) *Minimize charge losses during the balancing:* While active cell balancing transfers charge between cells, the process is not completely lossless. The resistance of the components in the current path creates a small energy dissipation. When charge is transferred between neighboring cells in one direction, a later transfer should not require charge flow in the opposite direction, as this would render the initial transfer inefficient. Consequently, the SoC change of each cell should be monotonic except the case that a cell is used as a *shuttle* cell where it receives charge from a neighbor to hand it over to the other neighbor. Furthermore, the global direction of charge transfers should always be monotonic, hence working towards the global equalization goal. We will formalize this requirement in Subsection 3.2.
- (2) *Minimize balancing time until an equilibrium is reached:* In scenarios where the balancing process is performed for the battery pack of an EV, the balancing time is relevant as it is usually constrained. Both the balancing strategy as well as the charge transfer current influence the time required to reach an equilibrium in the pack. For any chosen strategy, increasing the balancing current reduces the balancing time as more charge is transferred within a certain amount of time. Increasing the balancing current, however, will also increase resistive losses in the balancing architecture, as the relation between resistive losses and balancing current is quadratic. By choosing a strategy that maximizes concurrent transfers and does not perform inefficient transactions, the balancing time can be minimized for a chosen current.
- (3) *Maximize the usable pack-SoC as early as possible during balancing:* While a completely balanced battery is required in order to achieve a maximum pack-SoC during charging, there are scenarios where the usable SoC of a pack shall be increased as fast as possible. Here, a strategy that achieves the overall pack balance fast, but falls short of increasing the SoC of the cell with the lowest charge for a long time, may not be beneficial. Instead, it might be better to utilize a balancing strategy which is tailored to increase the SoC of the cells with the lowest charge as fast as possible while being overall slightly less efficient regarding charge transfer losses or taking longer to achieve an overall balanced battery pack. The same considerations apply for fully charging the battery pack, where the cells with the highest SoC might need to have their SoC reduced quickly such that the charging process does not need to be stopped due to cells reaching their upper charge threshold.
- (4) *Minimize the stress on cells induced by balancing:* While the goal of cell balancing is to equalize the SoCs of cells in the battery pack, there is a trade-off between balancing and the wear exerted on the cells by the charge transfer. Keeping the cell balancing process active all the time and, hence, cells in balance, might have an adverse effect on the lifetime of the battery pack as the cells are subject to much more cycling than in a scenario where balancing is only performed when it is really required, such as charging to full capacity. In this case, any benefit of energy conservation by active balancing might be dominated by the reduced lifetime of the battery cells, leading to a negative economical balance. Whether



continuous balancing is beneficial depends on several factors such as the specific cell characteristics, balancing strategy and application scenario. Note that a lower energy loss of a balancing process also results in a lower stress on the cells as the overall amount of lost charge is proportional to the amount of transferred charge. Inefficient charge transfer strategies transfer charge in more steps, hence losing more energy on the way and consequently also stressing the cells more compared to a transfer process which achieves the equalization with less charge transfer steps.

### 3.2. Request-driven Charge Transfer Strategies

The main feature of smart cells is that they maintain their safe and healthy operating state on their own, cooperating with other cells via communication in order to perform pack-level functionality. This design paradigm affects the implementation of BMS algorithms such as active cell balancing strategies. Conventional centralized approaches where a global computation-centric algorithm decides the balancing strategy are replaced by distributed communication-based methods. In this context, active cell balancing can be considered as a negotiation-based task such that each cell achieves its target to be as close as possible to the average pack-SoC. To achieve this goal, each cell either requires to receive charge from its peers in case its SoC is below the pack average, or it has to give charge to its peers in case it is above the pack average. Independent from any specific strategy with whom and when to exchange charge, the individual target of achieving a SoC as close as possible to the pack average is the main objective for each cell. When every cell has its SoC corresponding to the average pack-SoC, the pack is balanced. Consequently, cell balancing can be performed by letting each cell work towards its goal to reach the pack average individually by negotiating favorable charge transfers with its peers by either requesting to receive charge from or acknowledging to give charge to them. With this general concept of *request-driven strategies*, it has to be discussed how cells can negotiate favorable charge transactions with their peers, fulfilling requirements of efficiency and speed while reaching the global goal. Here, we assume knowledge of the average pack-SoC in each smart cell, as broadcast-based bus communication makes this information available without additional cost. Extension of the algorithms to operate completely with local information would be favorable for a communication architecture solely based on a daisy-chain and will be addressed in future work.

All following considerations assume a neighbor-only charge transfer architecture such as the one presented in Figure 3. Furthermore, each charge transfer between two cells is performed for a specified time period  $T_m \in \mathbb{R}$ . After this time period has passed, the cells go again into the negotiation phase. Requests and hence the balancing are stopped if the SoC range  $\delta \in \mathbb{R}$  across all of the cells is smaller than an equalization threshold value  $\epsilon \in \mathbb{R}$ .

In order to achieve a balanced pack, all cells have to bring their SoC  $z_i \in \mathbb{R}$  towards the pack average  $\bar{Z}$  defined as:

$$\bar{Z} = \frac{1}{n} \cdot \sum_{j=1}^n z_j \quad (1)$$

Cells are indexed from 1 to  $n$  and connected in series with cell  $c_1$  being the cell at the positive terminal and cell  $c_n$  being the cell at the negative terminal of the battery pack. For a cell  $c_i$  whose SoC  $z_i$  is below the pack average  $\bar{Z}$ , it is obvious that  $z_i$  has to be increased by requesting charge from either the upper or lower neighboring cell  $c_{i-1}$  or  $c_{i+1}$ , respectively. Note that we refer to a neighboring cell as upper neighbor if it is closer to the positive terminal of the battery pack and as lower neighbor otherwise.

Furthermore, we denote with  $\bar{Z}_i^\uparrow$  the SoC average of the subset containing all cells above cell  $c_i$  and with  $\bar{Z}_i^\downarrow$  the SoC average of the subset containing all cells below  $c_i$ :

$$\bar{Z}_i^\uparrow = \frac{1}{i-1} \cdot \sum_{j=1}^{i-1} z_j \quad (2)$$

$$\bar{Z}_i^\downarrow = \frac{1}{n-i} \cdot \sum_{j=i+1}^n z_j \quad (3)$$

Generally, charge transfers are globally efficient and contribute to an equalization of the battery pack if, from the local perspective of a cell  $c_i$ , charge is moved in the direction of the subset with the smaller  $\bar{Z}_i$ . Hence, a cell  $c_i$  will request charge from its upper neighbor if  $\bar{Z}_i^\uparrow \geq \bar{Z}_i^\downarrow$  and from its lower neighbor otherwise. This policy is beneficial as it reduces the imbalance between  $\bar{Z}_i^\downarrow$  and  $\bar{Z}_i^\uparrow$ . Therefore, the respective neighbor of cell  $c_i$  to request charge from can be identified by  $c_{i+\lambda}$  with:

$$\lambda = \text{sgn}(\bar{Z}_i^\downarrow - \bar{Z}_i^\uparrow) \quad (4)$$

In the following, four strategies are presented which all operate based on request and acknowledge processes. Their respective policies under which condition to request and acknowledge charge transfers are summarized in Table I. Note that  $Z_i^{(min)}$  defines the set of the  $i$  smallest elements in  $Z = \{z_1, \dots, z_n\}$ . Algorithm 1 illustrates the request and acknowledgment processes running on each CMU with the respective policies for the chosen strategy. To ensure exclusive transfers between pairs of cells, a cell only requests charge or acknowledges requests if it is in the status *idle*. A status different from *idle* is assigned to smart cells during the time they are involved in an actual charge transfer or while being blocked due to architectural requirements during charge transfer of their neighbors.

*Below Average Strategy.* Each cell  $c_i$  in the battery pack compares its own SoC  $z_i$  with the pack average  $\bar{Z}$  and only requests charge from the neighbor  $c_{i+\lambda}$  if it is below the pack average  $\bar{Z}$ . A request is made in form of a CAN message and the receiver of the message (identified by an individual node ID) decides whether or not to acknowledge the request. The request is only acknowledged if the SoC of the requested cell  $c_{i+\lambda}$  is above the pack average. It will ignore the request otherwise. This strategy ensures that cells with a below-average SoC will request charge and cells with an above-average SoC will acknowledge requests and send charge to the requester. As charge requests are always made in the direction of the higher  $\bar{Z}_i$ , charge is never transferred in the wrong direction and hence the monotonicity criterion for the efficiency of the charge transfer is satisfied. We call this the *Below Average* strategy as requests are made if a cell is below average.

*Minimum Strategy.* While the *Below Average* strategy by design achieves a high level of concurrency and works towards a fast equalization of the overall pack, it is not necessarily increasing the effective pack-SoC as fast as possible. Bringing up the effective pack-SoC as fast as possible can be beneficial when the maximum possible energy shall be drawn from the pack, such as in conditions where the unbalanced SoC may not be sufficient to reach the next charging station for an EV. This has been discussed in the optimization criteria for cell balancing in the previous subsection. By contrast, balancing with the strategy presented in the following will increase the pack-SoC as fast as possible and hence the effective driving range of an EV. For this purpose, we focus on increasing the SoC of the cells with the lowest charge in the

Table I. Request and acknowledge policies for balancing strategies.

Strategy	$r = c_i$ requests from $s = c_{i+\lambda}$ if	$s = c_i$ acknowledges to $r = c_{i-\lambda}$ if
Below Average	$z_r < \bar{Z}$	$z_s > \bar{Z}$
Minimum	$z_r \in Z_{n/2}^{(min)}$	$(z_s - \Delta \geq z_r + \Delta) \vee (z_r = \min(Z))$
Maximum	$z_r < \max(Z)$	$((z_s \in Z_{n/2}^{(max)}) \wedge (z_s - \Delta \geq z_r + \Delta)) \vee (z_s = \max(Z))$
Min-Max	$z_r < \max(Z)$	$(z_s - \Delta \geq z_r + \Delta) \vee (z_r = \min(Z)) \vee (z_s = \max(Z))$

pack. Here, each cell  $c_i$  determines if it is among the  $n/2$  cells with the lowest SoCs in the pack. In this case, the cell requests charge from its neighbor  $c_{i+\lambda}$ . A request is acknowledged by  $c_{i+\lambda}$  if the charge transfer satisfies the requirement that, after the transfer, the SoC of  $c_{i+\lambda}$  will not become smaller than the one of  $c_i$ :

$$z_{i+\lambda} - \Delta \geq z_i + \Delta \quad (5)$$

Here,  $\Delta$  is a conservative estimate of the change in SoC by the transferred charge which can be easily calculated, considering the average balancing current and the balancing time. With this requirement, we ensure that the overall SoC of the pack is monotonically increasing. Furthermore, charge transfers to the cell with the lowest SoC in the pack are generally acknowledged in order to ensure that there always is a feasible transfer. We call this the *Minimum* strategy as it focuses on increasing the minimum SoC boundary of the pack.

*Maximum Strategy.* Instead of increasing the SoC of the cell with the lowest charge in the pack, we can alternatively decrease the SoCs of the cells with the highest charge in the pack. This approach may be beneficial if the goal is to fully charge the battery pack. As charging of the pack has to be stopped if the first cell reaches its upper SoC threshold, prioritizing charge transfers from the cells with the highest individual SoC allows to achieve a fully charged battery pack in a shorter period of time. For this purpose, each cell  $c_i$  except the one with the highest SoC in the pack always requests charge from its neighbor  $c_{i+\lambda}$ . A request is acknowledged by  $c_{i+\lambda}$  if it is among the  $n/2$  cells with the highest SoC in the battery pack and if the charge transfer satisfies the requirement that, after the transfer, the SoC of  $c_i$  will not become larger than the one of  $c_{i+\lambda}$  as stated in Eq. (5). With this requirement, we ensure that the upper bound of the individual cell SoCs in the pack is monotonically decreasing. Furthermore, charge transfers from the cell with the highest SoC in the pack are generally acknowledged in order to ensure that there always is a feasible transfer. We call this the *Maximum* strategy as it focuses on decreasing the maximum SoC boundary of the pack.

*Min-Max Strategy.* Combining both the *Minimum* and the *Maximum* strategy, we can monotonically increase the effective pack-SoC determined by the cell with the lowest charge in the pack and monotonically decrease the upper bound of the range of the SoCs of the cells in the pack. Consequently, requests are made by each cell except the one with the highest SoC in the pack. Requests are acknowledged as long as the requirement in Eq. (5) is satisfied. Furthermore, requests are generally acknowledged if the requester is the cell with the lowest charge or the requested cell has the highest charge in the pack in order to ensure that transfers are always feasible. We call this the *Min-Max* strategy as it is a combination of the *Minimum* and *Maximum* strategies.

---

**Algorithm 1:** Request and acknowledge processes of request-driven balancing strategies for cell  $c_i$ . Request policy and Acknowledge policy are true or false corresponding to the respective condition in Table I.

---

```

1 Request (to receive charge) process()
2   if Request policy and status=idle then
3     | request charge from  $s = c_{i+\lambda}$ ;
4   end
5 Acknowledge (to send charge) process()
6   if Acknowledge policy and status=idle then
7     | success=block_cells( $s, r$ );
8     | if success then
9       | acknowledge request from  $r$ ;
10      | transfer charge to  $r$  for time  $T_m$ ;
11     | end
12     | unblock_cells( $s, r$ );
13   end

```

---

#### 4. CYBER-PHYSICAL CO-SIMULATION FRAMEWORK

In order to design and analyze smart cell architectures presented in Section 2, including the request-driven balancing strategies introduced in Section 3, we propose a Cyber-physical Co-Simulation Framework (CPCSF) in this section which covers all hierarchy levels of the system architecture. The cyber-physical characteristics of the battery pack architecture using smart cells are determined by the interaction of the electrochemical battery cells with the electrical components of the SBM. The SBM is controlled by the algorithms running on the computational platform, using a communication network between the smart cells to achieve system-level functionality. For this purpose, we model the behavior of battery cells and the balancing architecture with high accuracy, allowing to analyze the individual SoC of the battery cells and the charge transfer efficiency with all relevant electrical and timing parameters down to the level of PWM signals controlling the balancing circuits. Our model of the CMU enables implementation of request-driven charge transfer strategies and contains a CAN bus communication interface where the coordination between the smart cells is performed with individual CAN messages modeled with typical timing characteristics.

The multiple components of our simulation require properties of the charge transfer PWM signals and the CAN messages to be performed at a time resolution in the micro- to milliseconds range. On the other hand, the charge equalization process for a complete EV battery pack can be in the domain of several hours, depending on the hardware parameters and the battery cell variation. This multi-timescale characteristic of the system is a challenging scenario for developing a simulator. Either runtimes tend to be unacceptable with conventional simulation approaches progressing the whole simulation in the finest time resolution, or the simulation accuracy significantly suffers from approximations and simplifications. Consequently, an architecture for the simulation has to be chosen which enables high accuracy, not abstracting away the effects of the smallest time scale operations, while on the other hand allowing to analyze the system-level performance. Although other implementation platforms such as MATLAB/Simulink could be considered, we specifically want to leverage the benefits of an object-oriented implementation without relying on commercial software, such that the Open Source release of our framework allows immediate access and can easily be applied and extended without licensing restrictions. Hence, we chose a process-based Discrete-Event Simulation (DES) architecture for our CPCSF using the

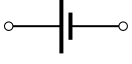
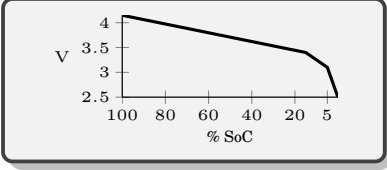

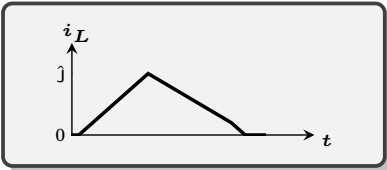


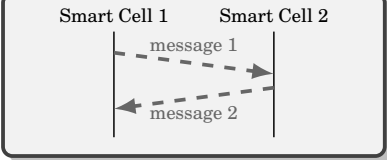
Layer	Architecture	Simulation Model
(1) Cell		
(2) Balancing		
(3) Computation		<pre>function request(sender) ... end function</pre>
(4) Communication		

Fig. 4. For each layer of the smart cell architecture, a corresponding simulation model is developed for the CPCSF. Each of the four layers is discussed in the respective subsections 4.1 to 4.4.

SIMPY FRAMEWORK FOR PYTHON [Team SimPy 2015]. In DES, the simulation flow evolves solely based on events that occur. This is in contrast to continuous simulation where for each time slice of the finest granularity the system development is tracked. For efficient DES, a good balance between simulation accuracy and abstraction of continuous properties to discrete events has to be found. In the remainder of this section, we introduce the modeling approaches for each layer of our CPCSF and discuss how we obtain a controlled balance between simulation speed and accuracy. The layers considered in the simulation are the

- (1) cell model and parameters (Section 4.1),
- (2) balancing architecture model and parameters (Section 4.2),
- (3) battery management algorithms of the CMU (Section 4.3),
- (4) CAN bus communication model (Section 4.4).

Figure 4 illustrates the relation between these layers of the smart cell architecture and the respective simulation models used for the CPCSF. All components are represented in separate modules such that they can be replaced by other models. Each modeling level is implemented independently from the others and, due to the object-oriented software engineering concept, the CPCSF is therefore easy to modify or extend. As we have designed the system for modularity, the CAN communication could, e.g., easily be replaced by FlexRay or Automotive Ethernet without affecting other parts of the implementation. Similarly, we have structured balancing architecture implementations

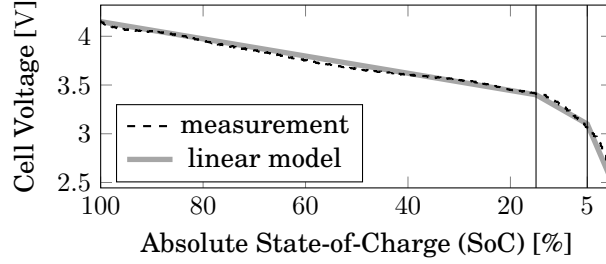


Fig. 5. Relation between cell voltage and SoC for a INR18650-25R cell from SAMSUNG, approximated by a piecewise-linear model.

to be a combination of the equivalent circuit hardware modeling, as discussed in Section 4.2, and a capabilities definition, which is then utilized by the balancing strategies to determine allowed transactions. This modular approach allows to adapt the CPCSF to other BMS architectures or to implement other balancing architectures and cell chemistries. We are currently preparing an open-source release of the CPCSF to make all implementation details available to the scientific community. In the following, the modeling and simulation aspects for each layer are discussed.

#### 4.1. Battery Cell Simulation

The characteristics of Li-Ion cells comprise a non-linear correlation between the charge stored in a cell, usually described by a SoC value in percent, and the open-circuit terminal voltage of the cell. Consequently, in order to simulate operations on the battery pack such as cell balancing with a quantitative model for the transfers that we identified in Figure 3(a), we need a model for the battery cells themselves. Figure 5 shows the voltage evolution and the corresponding SoC during a slow complete discharge (250 mA  $\approx 0.1C$ ) of a single INR18650-25R cell from SAMSUNG. The  $C$ -rate determines the discharge current with respect to the capacity of a cell.  $1C$  means that a current is chosen such that the cell is discharged within one hour,  $0.1C$  refers to a discharge rate such that the cell is discharged within 10 hours. Balancing is supposed to eliminate small imbalances in the pack and high currents are thus mostly not required. If behavior at higher currents becomes relevant, modeling approaches such as proposed in [Petricca et al. 2013] could be considered. For instance, the large-scale simulation in Section 5 operates only at an average current per cell of about  $0.05C$ . The voltage evolves linearly even beyond the range of the SoC in use (roughly 20% – 80%). Note that often this range is used for the usable SoC of battery cells, hence the absolute range between 20% and 80% translates into a usable SoC range between 0% and 100%, respectively. As cells are significantly stressed in the lowest and highest 20% of the absolute SoC range, these regions are therefore commonly not used in applications as the cycle life requirements for the cells usually dominate. There are two kinks around 15% and 5%. This behavior can be captured well by a piecewise linear model:

$$V(Q) = \begin{cases} V_{\zeta,0} + \zeta_0(Q - 0) & \text{if } Q < Q_1 \\ V_{\zeta,1} + \zeta_1(Q - Q_1) & \text{if } Q_1 \leq Q < Q_2 \\ V_{\zeta,2} + \zeta_2(Q - Q_2) & \text{if } Q_2 \leq Q < Q_{\max} \end{cases} \quad (6)$$

For our measurement, we have  $[Q_1 \ Q_2] = [0.05Q_{\max} \ 0.15Q_{\max}]$  and  $[V_{\zeta,0} \ V_{\zeta,1} \ V_{\zeta,2}] = [2.5 \ 3.1 \ 3.4]$ .  $\zeta_i$  are then calculated such that the curve is continuous. We use this piecewise linear model in our framework for the voltage-SoC relation of the simulated battery cells in the pack.

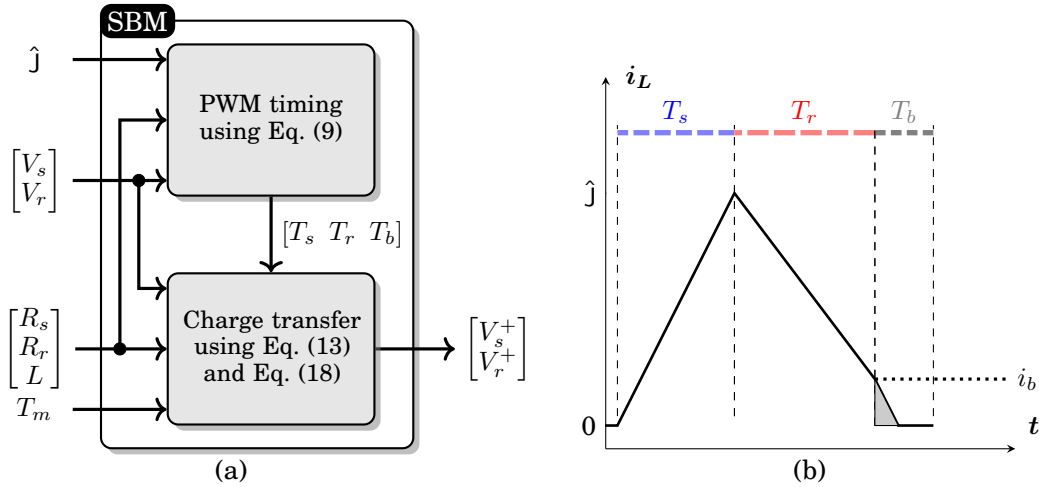


Fig. 6. The simulation back-end (a) for the SBM first calculates the PWM timing from the present voltages, the circuit parameters and the desired peak current. It then simulates transferring charge for a macro step of  $T_m$  – typically over  $10^5$  cycles – and yields the ensuing voltages. During each cycle (b), the transmitting cell charges an inductor for a period of  $T_s$ . After switching over – a brief moment that we treat as switching loss – the inductor then discharges into the receiving cell for a period of  $T_r$ . Finally, there is break period of  $T_b$  where no transistor is conducting and the inductor discharges very briefly over the diode of a MOSFET before the cycle repeats.

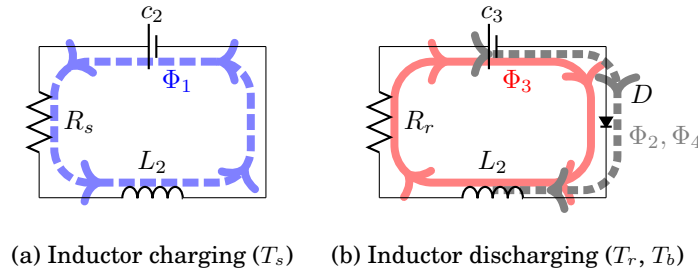


Fig. 7. By collecting resistances along the current path for configurations  $\Phi_1$ ,  $\Phi_3$  and  $\Phi_2, \Phi_4$  shown in Figure 3(a), the charge transfer can be captured by equivalent circuits. As we will show, they are structurally identical for all possible transfers and can thus form the backbone of an analytic charge transfer model.

## 4.2. Charge Transfer Simulation

For the analysis of pack-level active cell balancing performance, we want to perform long-term balancing simulations (several hours) of large battery packs ( $\approx 100$  cells) with high accuracy. To that end, we first outline an efficient short-term model for the behavior within individual PWM cycles in Section 4.2.1. This recursive model outperforms general purpose simulation approaches and can perform small balancing simulations. With many large balancing scenarios in mind, we further improve the simulation speed by deriving a closed-form solution for this recursion in Section 4.2.2. The performance of that model, summarized in Figure 6, is evaluated in Section 4.2.3.

*4.2.1. Short-term transfer model.* Once the transfer partners are identified, corresponding equivalent resistances can be established. In the circuit from Figure 3(a), we can

calculate them as

$$R_s = R_M + R_L + R_C \quad R_r = R_M + R_L + R_C \quad (7)$$

where  $R_M$ ,  $R_L$ ,  $R_C$  refer to the resistance of the involved MOSFET, inductor and battery cell, respectively. Aggregated in this fashion, the transfer dynamics can then be calculated based on the equivalent circuits in Figure 7.

*Transfer losses.* The cell voltage is assumed to remain constant during individual PWM cycles. This is justified by the high PWM frequency (1 kHz to 100 kHz) and the slowly evolving cell voltage (the discharge in Figure 5 occurs over multiple hours). Both charging ( $T_s$ ) and discharging ( $T_r$ ) phases (see Figure 7) are then governed by a first-order Ordinary Differential Equation (ODE). Using standard techniques, detailed in, e.g., [Baronti et al. 2013] we obtain the following description for the intra-cycle behavior.

$$i(t, V, i_0, R) = \frac{V}{R} - \frac{V - i_0 R}{R} \exp\left(\frac{-R}{L}t\right) \quad (8)$$

$$T_d(i_d, V, i_0, R) = \frac{-L}{R} \log\left(\frac{V - i_d R}{V - i_0 R}\right) \quad (9)$$

$$q(T, V, i_0, R) = \frac{V}{R}T + \frac{L(V - i_0 R)}{R^2} \left[ \exp\left(\frac{-R}{L}T\right) - 1 \right] \quad (10)$$

Here,  $i$  is the inductor current,  $T_d$  is its inversion, i.e., a time period that yields a desired current  $i_d = i(T_d)$ , and  $q = \int_0^T i$  is the charge transferred during a time period  $T$ . The parameters of these formulas must be adjusted according to the individual phases. With the respective cell voltages  $V_s, V_r$ , the corresponding resistances  $R_s, R_r$  and the peak current  $\hat{J}$  (see Figure 6), this parameterization summarizes to

$$[V \ R \ i_d \ i_0] = \begin{cases} \begin{bmatrix} V_s & R_s & \hat{J} & 0 \end{bmatrix} & \text{for inductor charging } (T_s), \\ \begin{bmatrix} -V_r & R_r & 0 & \hat{J} \end{bmatrix} & \text{for inductor discharging } (T_r). \end{cases} \quad (11)$$

Eq. (10) includes dissipative losses from the circuit components and is sufficient to drive a simulation by calculating the evolution of all participating battery cells.

*Switching losses.* In addition to dissipative losses, we also face switching losses from the transitions of the PWM signals (cf. Figure 3(a)). These losses summarize to

$$E_{sw,s} = \frac{1}{2}(t_{\text{OFF}}\hat{J}V_s + C_{\text{OSS}}V_s^2) \quad E_{sw,r} = \frac{1}{2}(t_{\text{ON}}\hat{J}V_r + C_{\text{OSS}}V_r^2). \quad (12)$$

Here,  $C_{\text{OSS}}$  is the output capacitance of the transistor that must be charged before conduction.  $t_{\text{ON}}$  and  $t_{\text{OFF}}$  summarize turn-on delay plus rise time and turn-off delay plus fall time, respectively, and can be obtained from the transistor data sheets. The corresponding loss terms summarize the overhead incurred during the short signal transition at  $\hat{J}$  from  $\phi_1$  to  $\phi_2$  for  $t_{\text{OFF}}$  and  $\phi_2$  to  $\phi_3$  for  $t_{\text{ON}}$  where the current cannot be utilized. The transition from  $\phi_4$  to  $\phi_1$  does not incur the loss because  $i = 0$  there. Please refer to Chapter 4.3 *Switching Losses* in [Erickson and Maksimovic 2001] for further information.

*4.2.2. Long-term transfer model.* The equations developed in 4.2.1 quantitatively describe a single PWM cycle. Over many cycles, they define two recursions. We formulate these in terms of voltage because that allows for a long-term closed form solution with further speedup. Using the charge formulation from Eq. (10) within one linear part of the



voltage model from Eq. (6), the sender voltage becomes

$$\begin{aligned} V_s[k+1] &= V_s[k] - \zeta q^s(V_s[k]) \\ &= \underbrace{\left[ 1 - \left( \zeta \frac{T_s}{R_s} + \zeta \frac{L}{R_s^2} \left( \exp\left(\frac{-R_s T_s}{L}\right) - 1 \right) \right) \right]}_{=: \alpha} V_s[k] = \alpha^k V_s[0]. \end{aligned} \quad (13)$$

Similarly, the receiver voltage evolves as follows.

$$\begin{aligned} V_r[k+1] &= V_r[k] + \zeta q_r(V_r[k], i_0) + \zeta q_b \\ &= V_r[k] + \zeta \left[ \frac{-V_r[k] T_r}{R_r} + \frac{L(-V_r[k] - i_0 R_r)}{R_r^2} \left( e^{-R_r T_r/L} - 1 \right) + q_b \right] \\ &= \underbrace{\left[ 1 - \zeta \frac{T_r}{R_r} - \zeta \frac{L}{R_r^2} \left( e^{-R_r T_r/L} - 1 \right) \right]}_{=: \beta} V_r[k] - \zeta \frac{L i_0}{R_r} \left( e^{-R_r T_r/L} - 1 \right) + \underbrace{\zeta q_b}_{=: \gamma} \end{aligned} \quad (14)$$

Here,  $q_b$  is a constant correction term for the charge transferred over the safety diode during  $T_b$  (see Fig. 6). We use the estimate  $q_b = i_{b,\min}^2 \frac{L}{V_r[0]}$  with the lowest  $i_b$  and the lowest receiver voltage  $V_r[0]$  of the respective PWM cycle. Since  $q_b$  is extremely small compared to  $q_r$ , better estimates are usually not necessary, even omission may be acceptable in many cases (see also Section 4.2.3).

After defining  $\alpha, \beta \in R_r$ , we now want to substitute  $i_0$ . We thus consider the varying peak current at the end of the charging phase as given by Eq. (8) and introduce another helper constant  $d \in \mathbb{R}$ :

$$i_0 = \frac{1}{R_s} \left( 1 - e^{-R_s T_s/L} \right) V_s[k] =: d V_s[k] \quad (15)$$

Given the current in this form, we can now transform Eq. (14) further:

$$V_r[k+1] = \beta V_r[k] + \gamma + \underbrace{\zeta \frac{-L}{R_r} \left( e^{-R_r T_r/L} - 1 \right) d}_{=: \theta} V_s[k] = \beta V_r[k] + \gamma + \theta \alpha^k V_s[0] \quad (16)$$

Next, we introduce auxiliary voltage  $V_x[k] := V_r[k] + \frac{\alpha^k \theta}{\beta - \alpha} V_s[0]$  with the simpler recursion

$$V_x[k+1] = \beta V_x[k] + \gamma. \quad (17)$$

We can thus calculate  $V_r[k]$  by transforming  $V_x[0] = V_r[0] + \frac{\theta}{\beta - \alpha} V_s[0]$  and then using

$$V_r[k] = \beta^k V_x[0] + \frac{\beta^k - 1}{\beta - 1} \gamma - \frac{\alpha^k \theta}{\beta - \alpha} V_s[0]. \quad (18)$$

Note that this requires  $\beta \neq \alpha$  which follows from  $R_s < R_r$  in most cases. If required, it can even be enforced by slightly altering  $T_r$ , i.e., by transferring slightly more charge over the diode at the end of the discharge phase (cf. Figure 6). Together with Eq. (13), Eq. (18) represents a closed-form long-term transfer model, allowing the efficient simulation of thousands of PWM cycles in the CPCSF with high accuracy.

*Switching losses in long-term transfer model.* Dividing the energy dissipated during switching (Eq. (12)) by its respective voltage yields the corresponding charge that is lost during each PWM cycle. Solving the recursion with this additional data changes

the following parameters, leaving  $\tilde{\gamma} := \gamma$ ,

$$\tilde{\alpha} := \alpha - \frac{\zeta}{2} \left[ C_{oss} + dt_{OFF} \right] \quad \tilde{\theta} := \theta - \frac{\zeta}{2} dt_{ON} \quad \tilde{\beta} := \beta - \frac{\zeta}{2} C_{oss}. \quad (19)$$

With the parameters from Eq. (19), we can then operate Eq. (18) and Eq. (13) as before.

**4.2.3. Long-term transfer model evaluation.** In the following, we evaluate the closed-form model – Eqs. (13) and (18) – that we developed in Section 4.2.2. We consider every combination of the following parameter vectors as an individual scenario.

$$\begin{aligned} \hat{J} &\in \{0.25, 1.0, 2.0\} & (R_s \ R_r) &\in \left\{ (0.01 \ 0.012), (0.5 \ 0.6), (1.3 \ 1.4) \right\} \\ Q_{\max} &\in \{0.1 \text{ A h}, 1.1 \text{ A h}\} & (z_s[0] \ z_r[0]) &\in \left\{ (0.4 \ 0.3), (0.8 \ 0.2), (0.45 \ 0.65) \right\} \end{aligned} \quad (20)$$

For a macro step size of 10 s, driving the simulation in a step-by-step fashion requires around 3 s for each two-cell scenario on an INTEL(R) XEON(R) CPU E5-1620. This happens because 10 s of balancing time can easily correspond to  $10^5$  PWM cycles. By contrast, the closed form expression we developed in Section 4.2.2 requires less than 100  $\mu$ s on average. The speedup factor is around 45000 overall for this duration compared to iterative methods calculating individual steps such as the models from [Kauer et al. 2015a; Kauer et al. 2015b]. At the same time, the closed-form long-term model achieved a worst case relative error given by  $\epsilon_{\text{rel}} := \frac{\|\Delta Q_{\text{ef}} - \Delta Q_{\text{step}}\|}{\|\Delta Q_{\text{step}}\|}$  in the order of  $10^{-5}$  without including  $q_b$  in Eq. (14). Including this correction term reduces the relative error to  $\epsilon_{\text{rel}} \approx 10^{-8}$ . Note that higher capacities as in our final case study (Section 5) lead to an even more benign situation since the voltage changes more slowly.

### 4.3. Battery Management Algorithms Simulation

Corresponding to the system architecture described in Section 2, each smart cell in the CPCSF operates as an individual entity. Consequently, the computational device of each smart cell, which is represented by a microcontroller core, is considered as an independent process in the DES of the CPCSF. On our virtual computational core, we mimic the behavior of a real-time operating system such as Micrium  $\mu$ C/OS-III which was used for the development platform shown in Figure 2. Real-time operating systems allow to run several processes in a logically parallel fashion, using a time-sliced schedule considering process priorities. In the CPCSF, we implement individual processes for monitoring of the SoC of the cell, the operation of the CAN communication interface as well as the request and acknowledge processes of the active balancing strategy as described in Algorithm 1. Furthermore, charging control of the cell, processing and storing information received from other cells as well as safety-related operations are implemented, such as preventing over- and undercharging the cell. All acquisition of performance metrics of the simulation is implemented via an external monitor concept, running in a separated environment. Consequently, no non-functional code is integrated in the actual processes implementing the devices in the simulation, allowing future performance evaluation of different computational platforms.

### 4.4. CAN Communication Simulation

Exchange of information between smart cells is implemented using the CAN bus, a field bus initially designed for automotive use. The technology was first introduced in 1991 by BOSCH [Bosch 1991]. Due to its early public documentation and available implementations in Very High Speed Integrated Circuit Hardware Description Language (VHDL), practically every microcontroller manufacturer offers microprocessors

Table II. Frame duration at different CAN speeds. Exemplary a message length of  $n_{\text{data}} = 4$  byte is assumed.

Data Rate [kbit/s]	33	83	125	250	500	1000
$T_{\text{bit}}$ [ $\mu\text{s}$ ]	30.03	12	8	4	2	1
$T_{\text{frame}}$ [ms]	3.75	1.49	1	0.5	0.25	0.125

with included CAN interfaces. Thus it constitutes an affordable and robust network to be used in the proposed environment.

Every Electronic Control Unit (ECU) connected to the bus is equal in terms of sending and receiving capabilities. The broadcast-oriented topology uses Carrier Sense Multiple Access / Collision Resolution (CSMA/CR) to control access of transmitting units. Every CAN frame consists of a header containing an 11 or 29 bits identifier, a payload section with up to 8 data bytes and a trailer employing a 15 bits Cyclic Redundancy Check (CRC). Widely adopted transmission rates range up to 125 kbit/s (Low Speed) and 1 Mbit/s (High Speed), utilizing a two wire bus and are reduced to 33 kbit/s and 88 kbit/s on single wire CAN respectively. Transmitted logical values are generated high-impedance and thus recessive (1, high) and low-impedance therefore dominant (0, low). Thus, lower Identifier (ID) values are prioritized due to their leading zeros.

In the CPCSFS, parameter information exchange as well as negotiation of active balancing transactions on the cell level is performed via individual messages over CAN. To ensure realistic behavior of the bus simulation, messages are put into a send queue in the respective CMU, realizing a First In First Out (FIFO) buffer. Enqueueing a new message into the CAN buffer creates a new event in the simulation environment for bus access. Taking into account the message length and CAN bus speed, waiting time delays for message transmission on the bus are implemented. Transmission time of a frame  $T_{\text{frame}}$  is given by:

$$T_{\text{frame}} < 1.25 \cdot (67 \text{ bit} + n_{\text{data}} \cdot 8 \text{ bit}) \cdot T_{\text{bit}} \quad (21)$$

Here,  $n_{\text{data}}$  is the length of the message payload given in bits and  $T_{\text{bit}}$  is the transmission time of a single bit. The base length of a message is 67 bit, representing the length of header and trailer in a frame. The provided equation is a conservative approximation, since actual message lengths also depend on bit stuffing. Table II shows frame transmission times for different CAN data-rates at a data length of  $n_{\text{data}} = 4$  byte. This is due to the fact that most of the traffic between cells is generated by cell parameter broadcasts comprising a 32 bit single precision float value.

Lower speed CAN topologies yield a higher resistance to Electromagnetic Interference (EMI) and are thus preferable in safety-critical environments such as automotive, where high frequencies (i.e. PWM generation) exist. Here, 125 kbit/s is commonly used for communication in body electronics. Regarding the simulation framework, faster CAN transmission rates yield the requirement of a finer time granularity of the discrete events. Simulating the bus system from the hardware platform discussed in Section 2 for the proposed CPCSFS, a CAN bus speed of 125 kbit/s, resulting in a transmission delay of approximately 1 ms for each message sent, is chosen.

## 5. EXPERIMENTAL RESULTS

In this section, we discuss the experimental setup and the results obtained with the Cyber-physical Co-Simulation Framework (CPCSFS) introduced in Section 4. A screenshot of the Graphical User Interface (GUI) frontend of our CPCSFS implementation visualizing the SoC evolution over time during an active cell balancing run of a 96 smart cell battery pack, using the *Minimum* strategy is shown in Figure 8. All results were

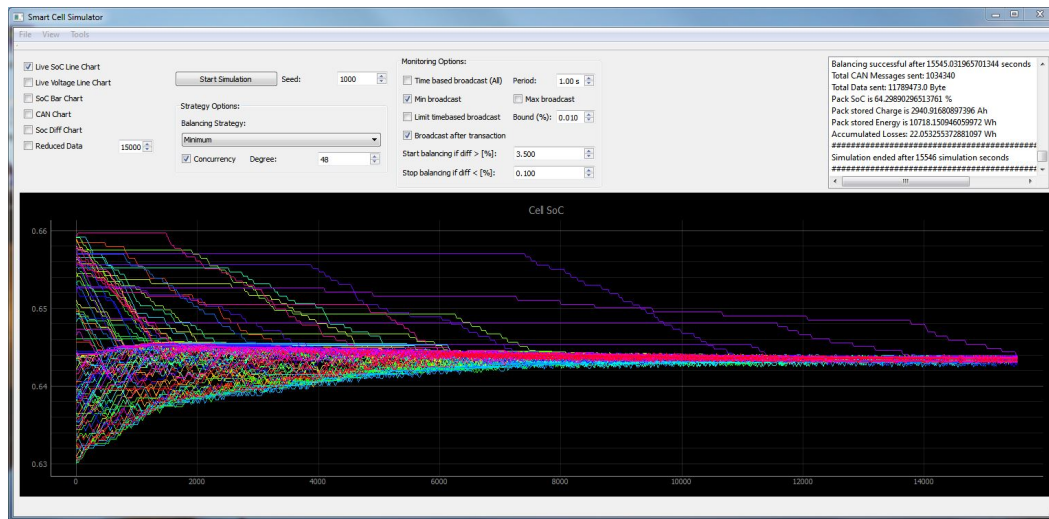


Fig. 8. Co-simulation framework performing simulation of active cell balancing with the *Minimum* strategy for 96 smart cells forming a 21.6 Wh battery pack.

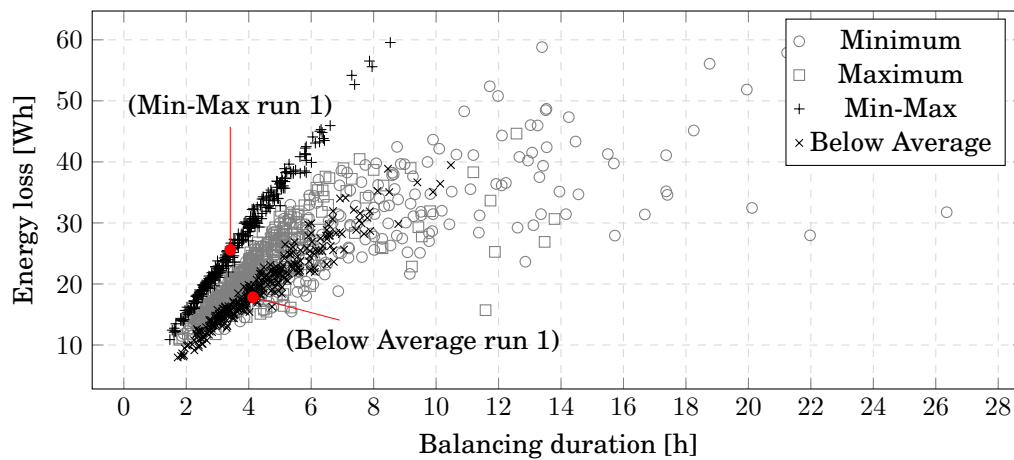


Fig. 9. Scatter plot showing the balancing duration until 0.1% difference in the pack was obtained and the corresponding energy loss encountered by the balancing of 300 SoC variations with a range of 3% for the analyzed battery pack with 96 smart cells. Each balancing strategy was simulated for the 300 SoC variations, resulting in 1200 points in the plot.

computed on a quad-core INTEL(R) XEON(R) CPU E5-1620, running at a maximum clock of 3.60GHz with 32GB of RAM.

To show the capabilities of the system-level simulation and assess the proposed request-driven balancing architectures in a realistic environment, we model a typical EV 21.6 kWh battery pack, using the smart cell architecture described in Section 2. Besides the smart cell architecture, its parameters are similar to the packs found in recent electric vehicles such as the Nissan Leaf or BMW i3. Our pack is arranged in a 96S24P fashion with a nominal voltage of 360 V. This term specifies that it consists of 96 series-connected smart cells with each smart cell comprising 24 parallel-connected SAMSUNG INR18650-25R cells, each with a nominal voltage of 3.75 V and a nominal

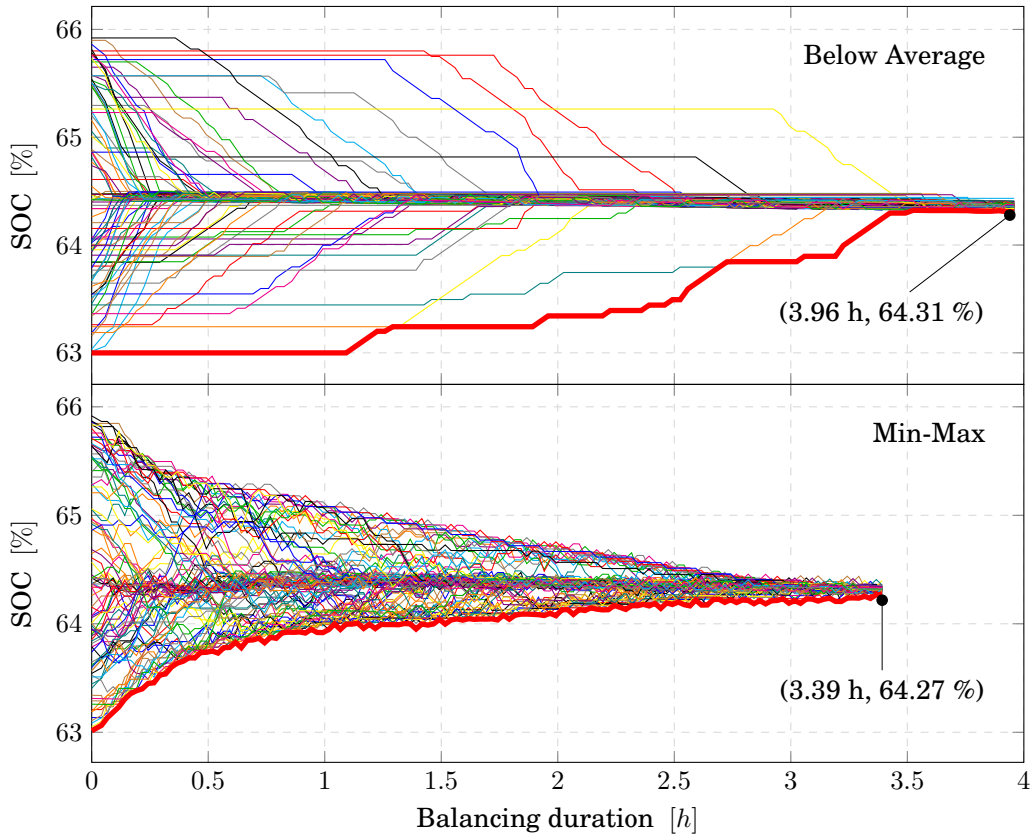


Fig. 10. Exemplary SoC evolutions over time with an initial SoC variation obtained from random seed 1000 (highlighted in Figure 9) of the *Below Average* and the *Min-Max* strategies.

capacity of 2.5 A h. Note that parallel-connected cells are electrically indistinguishable and hence are considered as a single cell from a battery management perspective. We chose these cells as we have been able to characterize them in our laboratory, yielding the discharge curve shown in Figure 5 and exhibiting a typical internal resistance of  $0.02215 \Omega$ .

For the simulations, we have performed a batch run on 300 different SoC variations in the battery pack. While, for comparability, we always used a SoC range of 3% between lowest to highest cell-SoC in the pack, the starting point of the range is varied between 20% and 80% SoC. Each cell currently not involved in a charge transfer checks its requirement to request charge once per second. The pack is considered as equalized when the SoC deviation  $\delta = \max(Z) - \min(Z)$  falls below  $\epsilon = 0.1\%$  and the balancing process is stopped. Note that for a 21.6 kW h battery pack,  $\epsilon = 0.1\%$  results in a maximum deviation of 21.6 W h across the whole pack, which we consider as reasonable. With each of the 300 SoC variations, we performed simulations for the proposed four balancing strategies *Minimum*, *Maximum*, *Min-Max* and *Below Average*, resulting in a total of 1200 simulation runs with an overall simulation runtime of 97 h. Hence, a single simulation run for a complete equalization of the 96-cell battery pack takes less than 5 minutes on average. This enables fast evaluation of changes in algorithms and architectures. The simulation runs were performed with the parameters

of the balancing architecture summarized in the following:

$$\begin{aligned} [R_L \ R_C \ R_M \ L \ Q_{\max}] &= [0.005\Omega \ \frac{1}{24}0.02215\Omega \ 0.0011\Omega \ 12 \ \mu\text{H} \ 24 \times 2.5 \ \text{A h}] \\ [\hat{J} \ J_{\text{avg}} \ t_{\text{on}} \ t_{\text{off}} \ C_{\text{oss}}] &= [12 \ \text{A} \ 3 \ \text{A} \ (6.7 + 6.0) \ \text{ns} \ (34 + 4.4) \ \text{ns} \ 1700 \ \text{pF}] \end{aligned} \quad (22)$$

These values correspond to measurements obtained on a hardware implementation of the circuit modeled in Section 4.2, comprising a BOURNS 1140-120K-RC inductor, INFINEON OPTIMOS BSC010NE2LS transistors and SAMSUNG INR18650-25R battery cells. Note that the peak current  $\hat{J}$  of 12 A in the inductor results in an average balancing current  $J_{\text{avg}}$  of 3 A.

The results illustrated in the scatter plot in Figure 9, comparing balancing duration and energy loss of the individual balancing runs, show that both the *Min-Max* and *Below Average* strategies show very compact results across the different SoC variations. Table III summarizes the obtained minimum, maximum and average values for the balancing time and energy loss. For comparison, results for passive balancing with  $J_{\text{avg}} = 0.5 \ \text{A}$  and full concurrency are also given. Here, all cells are brought to the SoC of the cell representing the pack minimum. Consequently, high energy losses of more than one order of magnitude in the average case compared to active balancing with the *Below Average* strategy are encountered. Note that for passive balancing, the balancing current only influences the balancing time and not the energy efficiency. High current values, however, generate significant heat which has to be dissipated by a cooling system in order not to damage cells or circuitry. A detailed discussion of this trade-off comparing the performance of active and passive balancing techniques for centralized BMSs can be found in [Baronti et al. 2014b].

For active balancing, the results clearly show that the *Min-Max* and *Below Average* strategies dominate the *Minimum* and *Maximum* strategies. For certain SoC variations, both the *Minimum* and *Maximum* strategies exhibit a significant increase in balancing time, resulting from a decreased level of possible concurrent transfers due to the more restrictive request and acknowledge policies compared to the *Below Average* and *Min-Max* strategies. On the other hand, neither of the two strategies *Min-Max* and *Below Average* dominates the other, with the *Min-Max* strategy being slightly faster and the *Below Average* strategy being slightly more energy efficient. For these two strategies, exemplary SoC developments over time with the same initial variation (highlighted in Figure 9) are plotted in Figure 10. The different characteristics of the balancing strategies can clearly be seen with the *Below Average* strategy performing every transfer between cells below the pack-SoC average requesting charge and the cells acknowledging the request if they are above the SoC average. Here, although achieving a relatively fast equalization with a very low energy loss and high final pack-SoC, the effective pack-SoC, determined by the lowest cell SoC within the pack, only increases very late in the balancing run and stays at the starting value for more than one hour. By contrast, the *Min-Max* strategy immediately increases the effective pack-SoC by bringing up the cells with the lowest SoC in the pack, as well as directly bringing the cells with the highest SoC towards the average. This is beneficial in situations where the pack either has to mobilize additional effective capacity, e.g., for increasing the driving range of an EV by balancing when the pack approaches its minimum SoC, or when the pack shall be charged fully, requiring that no cells with significant above-average SoC are present. Furthermore, the *Min-Max* strategy is more than half an hour faster than the *Below Average* strategy in this scenario, however with a slightly higher energy loss. Finally, to illustrate the scalability of the CPCSF, we have performed runs of the *Below Average* strategy with the same parameters as above but with a pack containing 12, 24, 48, 96 and 192 smart cells in series, respectively. The calculated runtime of the simulator for simulating 60 s of balancing behavior in the pack are given in Table IV.

Table III. Results of the simulation runs for the four active balancing strategies and passive balancing with 300 SoC variations each.

Strategy	Balancing Time [h]			Energy Loss [Wh]		
	Min	Max	Avg	Min	Max	Avg
Below Average	1.725	10.475	4.411	7.958	39.477	19.302
Minimum	1.850	26.350	6.884	11.490	58.806	26.657
Maximum	1.775	13.792	4.473	10.850	44.637	22.251
Min-Max	1.475	8.533	3.679	10.886	59.560	26.863
Passive	6.348	6.948	6.804	192.54	298.05	249.48

Table IV. Runtime for simulating 60 s of balancing behavior for different battery pack sizes.

Pack Topology	12S24P	24S24P	48S24P	96S24P	192S24P
Simulation runtime [s] for 60 s balancing	0.13	0.42	1.18	4.4	16.8

While there is a quadratic growth in runtime with increasing number of cells, the overall short simulation times show the suitability of the framework for rapid design evaluation.

## 6. RELATED WORK

In this section, the related work relevant in the context of this paper is discussed.

### 6.1. Design of Battery Management Systems

A comprehensive overview covering the state of the art of Li-Ion BMSs for EVs is given in [Brandl et al. 2012]. In commercial battery packs, centralized BMS architectures are still dominating. Here, a central master controller specifically tailored to the pack acquires all sensor information such as voltage and temperature of individual cells as well as the pack current. Additionally, it processes and creates control signals for cell balancing. Maintaining the Li-Ion cells in their safe operating range such that upper and lower thresholds of SoC and temperature are not crossed is mandatory in order not to damage the cells and the main task of the BMS. In this context, challenges and best practices of BMS design are thoroughly discussed in [Lu et al. 2013].

The recently emerging approaches to decentralize the control of the battery pack are driven by the requirements of higher efficiency, modularity and easier integration in order to cope with the perennial demand for shorter design cycles and time-to-market. Decentralization at the level of sensing and balancing hardware with conventional centralized control is proposed in [Baronti et al. 2012; Otto et al. 2012]. The concept on which this paper builds upon is characterized by fully autonomous smart cells that individually control their parameters and coordinate their actions with other smart cells via communication. It was initially introduced in [Steinhorst et al. 2014].

### 6.2. Active Cell Balancing Architectures and Strategies

With the goal to equalize the charge levels of individual cells in a series-connected battery pack, cell balancing is an important task performed by BMSs. Various cell balancing approaches are reviewed in [Cao et al. 2008; Moore and Schneider 2001; Daowd et al. 2011]. Generally, architectures are classified into *passive* and *active* cell balancing. Passive cell balancing dissipates excess energy of cells above minimum pack SoC via a resistor until they reach the charge level of the weakest cell [Kutkut and Divan 1996; Stuart and Zhu 2009]. While this technique is easy to implement and has low cost in comparison with the active cell balancing techniques, it suffers from low efficiency since all excess energy is dissipated as heat across the balancing resistors.

In contrast to passive cell balancing, active approaches increase the energy efficiency of the balancing process significantly by transferring energy from cells with high SoC to cells with low SoC instead of dissipating it.

While capacitors and transformers can be employed as the energy storage element in active balancing architectures, inductor-based approaches are beneficial regarding energy efficiency and hardware architecture, especially in the context of smart cells where modularity is important and homogeneous modules are required for each SBM. Here, approaches range from simple architectures that can perform charge transfers only between neighboring cells such as the one used in this paper, see [Kutkut 1998], to complex architectures that enable non-neighbor transfers and cell bypassing [Kauer et al. 2013] or enabling many-to-many transfers [Kauer et al. 2015b].

### 6.3. Modeling of Cell Balancing Architectures

In the control domain, it is typically assumed that the effective balancing current can be freely chosen as an input variable. Under this assumption, the complex problem of optimal charge routing can be formulated as a linear program. However, this ignores actuation aspects and the influences of changing cell voltages. The rising voltage of the receiving cell, for instance, affects the switch timing and, therefore, the amount of charge transferred per cycle. Assuming linear dynamics, the worst-case efficiency of various balancing topologies can be calculated [Preindl et al. 2013] and model predictive control with global knowledge and a horizon of 1 becomes an optimal strategy [Danielson et al. 2013]. In [Caspar and Hohmann 2014], the authors suggest to continuously adjust the PWM settings to keep the current high. This approach still has other drawbacks, however, like omitting the energy lost in transistor switching or modeling balancing current and link usage as a single variable. Both of these simplifications may be justifiable in the optimization setting operated in. For simulation approaches on the other hand, the state of the art is determined by intra-cycle models that provide a closed-form equation for individual PWM cycles where they achieve SPICE-level accuracy [Kauer et al. 2013; Kauer et al. 2015a]. While they are hugely faster than SPICE simulations, these approaches are only sufficient for evaluating individual balancing scenarios. We have built upon these in Section 4.2.2 and derived a closed formulation for long-term transfer that can be evaluated instantly.

### 6.4. System-level Balancing and BMS Simulation

While co-simulation approaches are established in other domains such as smart grid [Hua et al. 2011], system-level simulation of BMS and active cell balancing architectures, especially enabling development and investigation of balancing strategies, has not been extensively considered in literature. A simulation framework to compare different active balancing approaches with centralized control including the load behavior of an EV drivetrain is presented in [Caspar et al. 2014]. Approaches to analysis of balancing architectures in BMSs are usually hardware-oriented and cover module-level or intra-module verification aspects [Guerin and Liu 2010; Baronti et al. 2014a; Lukasiwycz et al. 2014] and only implicitly investigate the global balancing strategy suitable for a given architecture. In [Lee and Cheng 2005], a balancing strategy using fuzzy control is presented and evaluated using SPICE simulation for a single pair of cells transferring charge, however not considering pack-level criteria. An approach from the control domain investigates the performance of seven balancing architectures with an optimized centrally controlled strategy in [Preindl et al. 2013], but abstracts certain hardware characteristics that we consider as important in our CPCSF for the required level of simulation accuracy. A first approach to fully distributed algorithms for charge equalization of supercapacitors is presented in [Liu et al. 2015].



## 7. CONCLUDING REMARKS

In this paper, we have introduced a Cyber-physical Co-Simulation Framework (CPCSF) for battery packs equipped with smart cells that allows rapid design and analysis of algorithms, hardware architectures and parameter sets. Based on the concept of smart cells that consist of a cell and a Cell Management Unit (CMU) which locally manages the parameters of the cell, Battery Management System (BMS) functionality on pack-level is achieved by cooperation of the smart cells via communication. Such a completely decentralized architecture requires novel algorithms for BMS functions which are centrally controlled in state-of-the-art architectures. Here, we illustrate the algorithmic possibilities of the smart cell battery pack architecture by introducing a new class of request-driven cell balancing strategies. Cell balancing is one of the most important functions of a BMS and involves both PWM signals in the microsecond range as well as overall pack charge equalization happening in the range of hours. Consequently, accurate analysis of the system-level behavior is difficult without a capable simulation framework comprising detailed optimized models of the system components. The CPCSF that is introduced in this paper combines sophisticated modeling approaches of system components such as the battery cells, balancing hardware, algorithms, computational devices and communication channel using a Discrete-Event Simulation (DES) approach. This enables a fast and accurate multi-level cyber-physical co-simulation of a complete EV battery pack with a capacity similar to the ones in the Nissan Leaf or BMW i3, but equipped with smart cells. As a consequence, this allows, for the first time, to quantitatively and qualitatively analyze the proposed request-driven balancing strategies in a realistic setup.

## REFERENCES

- F. Baronti, G. Fantechi, R. Roncella, and R. Saletti. 2012. Intelligent cell gauge for a hierarchical battery management system. In *IEEE Transportation Electrification Conference and Expo (ITEC)*. 1–5. DOI: <http://dx.doi.org/10.1109/ITEC.2012.6243471>
- F. Baronti, G. Fantechi, R. Roncella, and R. Saletti. 2013. High-Efficiency Digitally Controlled Charge Equalizer for Series-Connected Cells Based on Switching Converter and Super-Capacitor. *IEEE Transactions on Industrial Informatics* 9, 2 (2013), 1139–1147. DOI: <http://dx.doi.org/10.1109/TII.2012.2223479>
- F. Baronti, C. Bernardeschi, L. Cassano, A. Domenici, R. Roncella, and R. Saletti. 2014a. Design and Safety Verification of a Distributed Charge Equalizer for Modular Li-Ion Batteries. *IEEE Transactions on Industrial Informatics* 10, 2 (2014), 1003–1011. DOI: <http://dx.doi.org/10.1109/TII.2014.2299236>
- F. Baronti, R. Roncella, and R. Saletti. 2014b. Performance Comparison of Active Balancing Techniques for Lithium-Ion Batteries. *Journal of Power Sources* 267, 1 (2014), 603–609. DOI: <http://dx.doi.org/10.1016/j.jpowsour.2014.05.007>
- Bosch. 1991. Controller Area Network, Version 2.0b. (1991). <http://www.can.bosch.com/>.
- M. Brandl, H. Gall, M. Wenger, V. Lorentz, M. Giegerich, F. Baronti, G. Fantechi, L. Fanucci, R. Roncella, R. Saletti, S. Saponara, A. Thaler, M. Cifrain, and W. Prochazka. 2012. Batteries and battery management systems for electric vehicles. In *Proc. of Design, Automation Test in Europe Conference Exhibition (DATE)*. 971–976. DOI: <http://dx.doi.org/10.1109/DATE.2012.6176637>
- J. Cao, N. Schofield, and A. Emadi. 2008. Battery balancing methods: A comprehensive review. In *Proc. of Vehicle Power and Propulsion Conference (VPPC)*. 1–6. DOI: <http://dx.doi.org/10.1109/VPPC.2008.4677669>
- M. Caspar, T. Eiler, and S. Hohmann. 2014. Comparison of Active Battery Balancing Systems. In *Proc. of Vehicle Power and Propulsion Conference (VPPC)*. 1–8. DOI: <http://dx.doi.org/10.1109/VPPC.2014.7007027>
- M. Caspar and S. Hohmann. 2014. Optimal Cell Balancing with Model-based Cascade Control by Duty Cycle Adaption. In *IFAC World Congress*, Vol. 19. 10311–10318. DOI: <http://dx.doi.org/10.3182/20140824-6-ZA-1003.01613>
- C. Danielson, F. Borrelli, D. Oliver, D. Anderson, and T. Phillips. 2013. Constrained flow control in storage networks: Capacity maximization and balancing. *Automatica* 49, 9 (2013), 2612–2621. DOI: <http://dx.doi.org/10.1016/j.automatica.2013.05.014>
- M. Daowd, N. Omar, P. Van Den Bossche, and Joeri Van Mierlo. 2011. Passive and active battery balancing comparison based on MATLAB simulation. In *Proc. of Vehicle Power and Propulsion Conference (VPPC)*. 1–7. DOI: <http://dx.doi.org/10.1109/VPPC.2011.6043010>

- R. Erickson and D. Maksimovic. 2001. *Fundamentals of Power Electronics*. Springer.
- J. Guerin and W. Liu. 2010. Cell balancing algorithm verification through a simulation model for lithium ion energy storage systems. *SAE Publication 2010-01-1079* (2010). DOI: <http://dx.doi.org/10.4271/2010-01-1079>
- L. Hua, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili. 2011. Power system and communication network co-simulation for smart grid applications. In *Proc. of Innovative Smart Grid Technologies (ISGT)*. 1–6. DOI: <http://dx.doi.org/10.1109/ISGT.2011.5759166>
- M. Kauer, S. Narayanaswami, S. Steinhorst, M. Lukasiewicz, S. Chakraborty, and L. Hedrich. 2013. Modular System-Level Architecture for Concurrent Cell Balancing. In *Proc. of the Design Automation Conference (DAC)*. 155:1–155:10. DOI: <http://dx.doi.org/10.1145/2463209.2488926>
- M. Kauer, S. Narayanaswamy, M. Lukasiewicz, S. Steinhorst, and S. Chakraborty. 2015a. Inductor Optimization for Active Cell Balancing using Geometric Programming. In *Proc. of the Conference on Design, Automation and Test in Europe (DATE)*. 1–4. DOI: <http://dx.doi.org/10.7873/DATE.2015.0051>
- M. Kauer, S. Narayanaswamy, S. Steinhorst, M. Lukasiewicz, and S. Chakraborty. 2015b. Many-to-Many Active Cell Balancing Strategy Design. In *Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC)*. 267–272. DOI: <http://dx.doi.org/10.1109/ASPAC.2015.7059016>
- N. Kutkut. 1998. A modular nondissipative current diverter for EV battery charge equalization. In *Proc. of Applied Power Electronics Conference and Exposition (APEC)*, Vol. 2. 686–6902. DOI: <http://dx.doi.org/10.1109/APEC.1998.653973>
- N. Kutkut and D. Divan. 1996. Dynamic equalization techniques for series battery stacks. In *Proc. of the International Telecommunications Energy Conference (INTELEC)*. 514–521. DOI: <http://dx.doi.org/10.1109/INTLEEC.1996.573384>
- Y. Lee and M. Cheng. 2005. Intelligent control battery equalization for series connected lithium-ion battery strings. *Industrial Electronics, IEEE Transactions on* 52, 5 (Oct. 2005), 1297–1307. DOI: <http://dx.doi.org/10.1109/TIE.2005.855673>
- J. Liu, Z. Huang, J. Peng, and J. Wang. 2015. Distributed cooperative voltage equalization for series-connected super-capacitors. In *Proc. of the American Control Conference (ACC)*. 4523–4528. DOI: <http://dx.doi.org/10.1109/ACC.2015.7172041>
- L. Lu, X. Han, J. Li, J. Hua, and M. Ouyang. 2013. A review on the key issues for lithium-ion battery management in electric vehicles. *Journal of Power Sources* 226, 0 (2013), 272–288. DOI: <http://dx.doi.org/10.1016/j.jpowsour.2012.10.060>
- M. Lukasiewicz, S. Steinhorst, and S. Narayanaswamy. 2014. Verification of Balancing Architectures for Modular Batteries. In *Proc. of the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 30:1–30:10. DOI: <http://dx.doi.org/10.1145/2656075.2656104>
- S. Moore and P. Schneider. 2001. A review of cell equalization methods for lithium ion and lithium polymer battery systems. *SAE Publication 2001-01-0959* (2001). DOI: <http://dx.doi.org/10.4271/2001-01-0959>
- S. Narayanaswamy, S. Steinhorst, M. Lukasiewicz, M. Kauer, and S. Chakraborty. 2014. Optimal Dimensioning of Active Cell Balancing Architectures. In *Proc. of Design, Automation Test in Europe Conference Exhibition (DATE)*. 140:1–140:6. DOI: <http://dx.doi.org/10.7873/DATE.2014.153>
- A. Otto, S. Rzepka, T. Mager, B. Michel, C. Lanciotti, T. Günther, and O. Kanoun. 2012. Battery Management Network for Fully Electrical Vehicles Featuring Smart Systems at Cell and Pack Level. In *Advanced Microsystems for Automotive Applications 2012*, Gereon Meyer (Ed.). Springer Berlin Heidelberg. DOI: [http://dx.doi.org/10.1007/978-3-642-29673-4\\_1](http://dx.doi.org/10.1007/978-3-642-29673-4_1)
- M. Petricca, D. Shin, A. Bocca, A. Macii, E. Macii, and M. Poncino. 2013. An automated framework for generating variable-accuracy battery models from datasheet information. In *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED)*. 365–370. DOI: <http://dx.doi.org/10.1109/ISLPED.2013.6629324>
- M. Preindl, C. Danielson, and F. Borrelli. 2013. Performance evaluation of battery balancing hardware. In *Proc. of the European Control Conference (ECC)*. 4065–4070.
- S. Steinhorst, M. Lukasiewicz, S. Narayanaswamy, M. Kauer, and S. Chakraborty. 2014. Smart Cells for Embedded Battery Management. In *Proc. of the International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*. 59–64. DOI: <http://dx.doi.org/10.1109/CPSNA.2014.22>
- T. Stuart and W. Zhu. 2009. Fast equalization for large lithium ion batteries. *Aerospace and Electronic Systems Magazine, IEEE* 24, 7 (July 2009), 27–31. DOI: <http://dx.doi.org/10.1109/MAES.2009.5208557>
- Team SimPy. 2015. SimPy Discrete Event Simulation Library for Python. <http://simpy.readthedocs.org/>. (2015). Accessed: 2015-04-20.

Received September 2015; revised December 2015; accepted February 2016