Synthesis of Active Cell Balancing Architectures for Battery Packs

Martin Lukasiewycz, *Member, IEEE,* Matthias Kauer, *Member, IEEE,* and Sebastian Steinhorst, *Member, IEEE*

***Abstract*—Active balancing architectures effectively increase the efficiency of large battery packs by equalizing charge between cells. For this purpose, a balancing circuit and appropriate control scheme have to be designed to enable the charge transfer via energy storage elements such as inductors. Using a manual approach to design balancing architectures can be tedious and error-prone, resulting in potentially suboptimal solutions. As a remedy, this paper presents an automatic synthesis of balancing circuits and their corresponding control, optimizing the number of required MOSFETs and necessary control signals. The proposed synthesis combines a SAT solver to explore the search space with a graph-based verification that iteratively excludes infeasible solutions until the optimal architectures are obtained. The experimental results are carried out for three given template circuits and two signal templates. The synthesis results in architectures that are superior in terms of all design objectives in comparison to solutions from literature that result from a manual design approach.**

***Index Terms*—Circuit synthesis, active cell balancing, battery pack design.**

I. Introduction

LARGE battery packs are equally important for Electric Vehicles (EVs) and smart grid applications. These packs consist of a large number of battery cells which are connected in series to provide a desired output voltage. Since battery cells suffer from capacity variations which further increase with aging, charge balancing is required to utilize the full energy of a battery pack. Note that cells which are connected in parallel to increase the effective capacity of packs form an electrical unit and can be considered as one large cell that is inherently balanced. Consequently, only series-connected cells are relevant for the balancing process.

While passive balancing strategies equalize charge by dissipating energy via a resistor, active balancing architectures promise a more efficient way of charge equalization. In active balancing, charge is transfered via storage elements, such as inductors, between the cells. For this purpose, a circuitry consisting of Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFETs) and storage elements as well as a sophisticated control scheme are employed. There exist numerous active balancing architectures with different capabilities and properties which are the result of a manual design process.

While a manual design process is viable for simple architectures, upcoming requirements mandate more complex designs.

M. Lukasiewycz, M. Kauer, and S. Steinhorst are with TUM CREATE Limited, Singapore, 138602 Singapore e-mail: martin.lukasiewycz@tum-create.edu.sg.

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

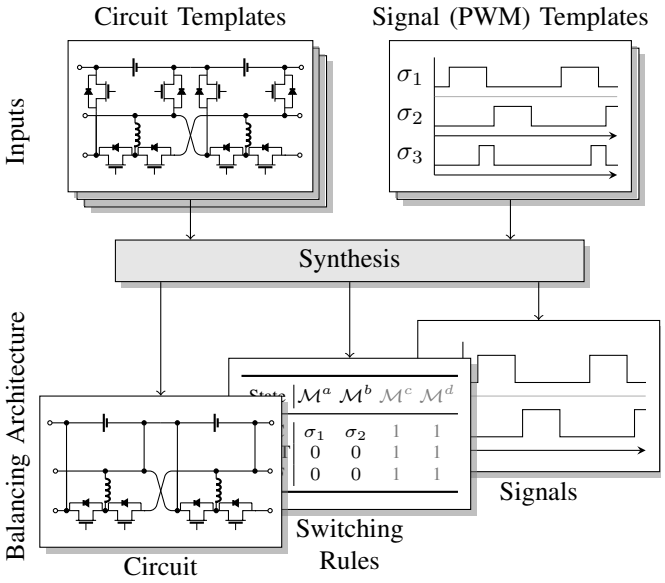


Fig. 1: Overview of the proposed synthesis approach. Given a set of circuit and signal templates, the synthesis determines an optimal balancing architecture that consists of the circuit, switching rules, and signals.

One example are scalable battery packs that consist of modular *smart cells* [1], [2], where balancing architectures are composable such that each cell has a dedicated balancing circuit and controller. In such a battery pack, a charge transfer between non-neighboring cells demands non-trivial architectures and control.

Contributions of the paper. This paper, for the first time, proposes an automatic synthesis approach for the design of active balancing architectures, including their control scheme. As illustrated in Figure 1, the proposed synthesis is using circuit and signal templates to determine an optimal circuit with the corresponding control that consists of the switching rules and signals. While the paper focuses on highly modular architectures for scalable battery packs, the proposed methodology can be adapted as well to conventional architectures with centralized control.

Section II introduces the proposed synthesis approach. First, the functional principles of active balancing are explained, using a simple circuit for charge transfer. Second, a model for active balancing is defined. Finally, the proposed synthesis flow and iterative approach are outlined. The iterative approach consists of two major steps per iteration:

- Section III specifies the satisfiability (SAT) based search engine. For this purpose, a set of constraints is defined

such that determined solutions satisfy the charge transfer requirements via the inductor as storage element. Since not all solutions are feasible, a verification step becomes necessary.
- Section IV introduces the verification step which is based on the previous work in [3]. Within the verification, potential faults, such as short circuits, that cannot be initially covered by the search engine, are detected. These faults are incrementally excluded from the search engine such that the synthesis obtains optimal and correct results.

Section V presents experimental results, showing a synthesis with three template circuits and two signal templates. Within an overall runtime of less than two hours, an optimal active balancing architecture is obtained. Compared to a manually designed architecture from literature like in [3], the automatically synthesized architecture is superior in terms of lower complexity regarding components and control.

Finally, Section VI presents related work before the paper is concluded in Section VII.

Architecture synthesis integration. The developed methodology is part of a design flow to obtain fully functional balancing architectures with specific components and control. The design flow is defined as follows:

1) Requirements definition
2) Architecture synthesis
3) Component selection and sizing
4) Control design

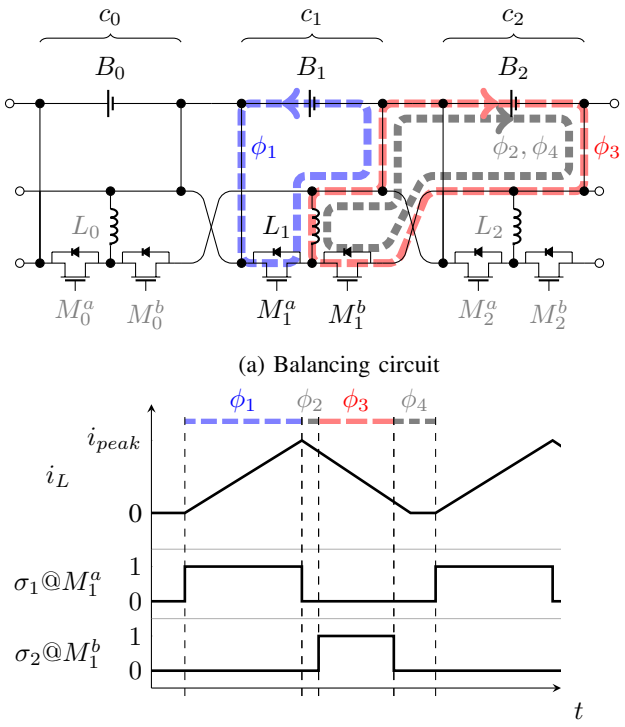
In the first stage, requirements are defined depending on the field of application of the balancing architecture. Here, appropriate scenarios have to be defined that are used within the search engine. The set of scenarios has to cover all possible charge transfer situations such as concurrent transfers, non-neighbor transfers via different distances, etc. On the other hand, the set should not be too large in order not to overstrain the search engine. Such a set of scenarios is given and discussed in the experimental results, see (6). Additionally, the requirements might impose problem-specific optimization objectives for the proposed synthesis. In this paper, we provide general optimization Objectives (O1) to (O3) that reduce the amount of hardware and control.

The second stage in the design flow is the proposed methodology and contribution of this paper. From the requirements, the methodology obtains a system-level architecture that is independent of the underlying components and operating control.

Components and detailed circuit controls are determined in stage three and four. Computer-aided component selection and sizing approaches have been explored in literature, see [29], [30], for MOSFETs and inductors in active cell balancing. The specific components also mandate certain operating constraints, e.g., the maximum distance of charge transfers. Depending on the components, the control that includes the exact duration of Pulse Width Modulation (PWM) signals has to be determined. A guideline for an optimal control design can be found in [28].

II. Synthesis Approach

In the following, the functionality of active cell balancing architectures is introduced using Figure 2. To formalize the problem, a model is presented that describes the circuit and control. Finally, the basic functionality of the proposed synthesis flow and approach are outlined.



(a) Balancing circuit

(b) Control scheme and resulting inductor current

Fig. 2: Example of a balancing architecture that illustrates a charge transfer between neighboring cells via an inductor. The transfer is carried out in four phases ϕ_1 to ϕ_4 where appropriate PWM signals are applied to the MOSFETs.

A. Balancing Architectures

Active cell balancing for battery packs relies on architectures that are capable of transferring charge between cells. Such an architecture, which is a combination of a balancing circuit and control scheme, is illustrated in Figure 2. The circuit consists of battery cells B, a set of MOSFETs M, and inductors L. Each cell with the additional balancing circuitry forms a module c. A MOSFET is controlled like a switch and it has a diode between the drain and source that blocks current flow in one direction.

By controlling the MOSFETs, charge can be transferred between cells via the inductors that function as temporary energy storage elements. For this purpose, MOSFETs can be either closed (1), open (0), or controlled by PWM signals. For the circuit in Figure 2a, charge is transferred if all MOSFETs are open except two that are controlled by non-overlapping PWM signals as illustrated in Figure 2b. The PWM signals result in four phases ϕ_1 to ϕ_4 that are occurring periodically and ensure the charge transfer.

In phase ϕ_1, the inductor L_1 is charged from B_1 via M_1^a, which is temporarily closed. The duration of this *charging phase* has to be chosen in consideration of the inductance and cell voltage to ensure an optimal utilization of the inductor.

Note that the current flow does not pass any diode since this would dissipate energy and unnecessarily reduce the efficiency of the transfer.

In phase ϕ_2, M_1^a is opened and the charging of the inductor is discontinued. This *freewheeling phase* is necessary to account for the time that MOSFETs require to change between their open and closed state and vice versa. Without phase ϕ_2, it could happen that both MOSFETs M_1^a and M_1^b were closed for a short amount of time and a short circuit over B_1 and B_2 via these two switches occurred. At the same time, it has to be ensured that each charged inductor in the freewheeling phase can discharge via a diode, otherwise the voltage of the inductor would immediately increase and all energy is dissipated via the inductor itself, damaging components.

In phase ϕ_3, M_1^b is temporarily closed and the inductor discharges its energy into cell B_2. In this *discharge phase*, the current flow does not pass any diode to ensure an energy-optimal transfer.

In the final phase ϕ_4, M_1^b is opened and the discharging of the inductor is carried out via the diode. This *blocking phase* ensures that the inductor is fully discharged, but at the same time B_2 cannot start charging the inductor in the opposite direction once the current flow stops.

While this example shows a single transfer, the architecture in Figure 2 allows concurrent charge transfers. That means in a battery pack with more than three cells in series there might be multiple charge transfers at the same time, see for instance [28, Figure 3] as an example of concurrent transfers. Concurrent charge transfers significantly reduce the balancing time, particularly for large battery packs with dozens of cells in series. Note that concurrent charge transfers are also required for the architectures in our experimental results as can be seen in the defined scenarios in (6) where at least two transfers per scenario are considered.

B. Modeling

In the following, a model for balancing architectures is defined. This model comprises the circuit, switching rules, and signals.

Circuit. A battery pack consists of a set of $N=\{0,1,..,|N|-1\}$ series-connected modules where each module c_n with $n\in N$ consists of the battery cell B_n, the MOSFETs M_n^x, the diodes D_n^x (one per MOSFET), and one inductor L_n. The set of elements might be extended by capacitors, transformers, and other components.

In the following, the set of MOSFETs is defined as $\mathcal{M}$. All MOSFETs of a certain type are defined as $\mathcal{M}^x$ while all MOSFETs of a certain module with index n are defined as $\mathcal{M}_n$.

Signals. To enable charge transfers, certain MOSFETs have to be in a predefined static state (open or closed) or controlled by predefined PWM signals, see Figure 2b. The set of all PWM signals is defined by Σ and for each $\sigma\in\Sigma$ the function

$$\delta:\Sigma\rightarrow\mathbb{R}\times\mathbb{R} \tag{1}$$

determines the start and end time when the signal is active, i.e., the respective MOSFETs are temporarily closed. From the signals, phases Φ are determined such that a single phase $\phi\in\Phi$ is defined by a subset of signals from Σ. Corresponding to the signals, each phase has a start and end time determined by

$$\delta:\Phi\rightarrow\mathbb{R}\times\mathbb{R}. \tag{2}$$

The set of phases has a known order where two phases $\phi,\tilde{\phi}\in\Phi$ are consecutive if there exists a $t\in\mathbb{R}$ such that $\delta(\phi)=(t',t)$ and $\delta(\tilde{\phi})=(t,t'')$.

Switching Rules. A transfer scenario is defined by $\mathcal{T}$ where a single transfer $T=(n,\tilde{n})\in\mathcal{T}$ is defined by the source cell index $n\in N$ and destination cell index $\tilde{n}\in N$. Additionally, each transfer $T=(n,\tilde{n})$ defines a domain by

$$d(T)=\{\min(\{n,\tilde{n}\}),...,\max(\{n,\tilde{n}\})\} \tag{3}$$

and it holds that domains are disjoint:

$$\forall T,\tilde{T}\in\mathcal{T}:d(T)\cap d(\tilde{T})=\{\} \tag{4}$$

This requirement is necessary to avoid competing transfers by separating them.

For a certain transfer scenario $\mathcal{T}$, the configuration of each MOSFET M has to be determined accordingly. For this purpose, the function

$$c(\mathcal{T},n)\in S \tag{5}$$

defines a state for each module c_n for a certain scenario $\mathcal{T}$. A state $s\in S$ of a module defines whether charge is sent (SRC), received (DEST), the module is unused (OFF), or it is located between a sender and its corresponding receiver (BRIDGE). The states that are considered in this paper are listed in Table I, including the formal definition of function $c(\mathcal{T},n)$. Note that the direction of the charge transfer is explicitly considered. The number of states and their definition can be adapted correspondingly to the considered problem and charge transfer requirements. Finally, with a given state s and specific MOSFET M_n^x for the respective module, the signal that shall be applied is determined. These switching rules can be represented in a table as for instance illustrated in Table II.

C. Synthesis Flow

The basic synthesis flow is illustrated in Figure 1 and the corresponding developed design tool is shown in Figure 3. The design tool is implemented using the ECLIPSE framework. Thus, it is highly extensible in terms of functionality and layout.

The proposed design method is using templates for the circuit and signals which are in the considered cases a result

TABLE I: Definition of module states $s\in S$ for the configuration function $c(\mathcal{T},n)$.

State s	is the result of $c(\mathcal{T},n)$ iff
SRC_RIGHT	$\exists T=(n,\tilde{n})\in\mathcal{T}:n<\tilde{n}$
SRC_LEFT	$\exists T=(n,\tilde{n})\in\mathcal{T}:n>\tilde{n}$
DEST_RIGHT	$\exists T=(\tilde{n},n)\in\mathcal{T}:\tilde{n}<n$
DEST_LEFT	$\exists T=(\tilde{n},n)\in\mathcal{T}:\tilde{n}>n$
BRIDGE	$\exists T=(\tilde{n},\hat{n})\in\mathcal{T}:n\in d(T),n\neq\tilde{n},n\neq\hat{n}$
OFF	$\forall T\in\mathcal{T}:n\notin d(T)$

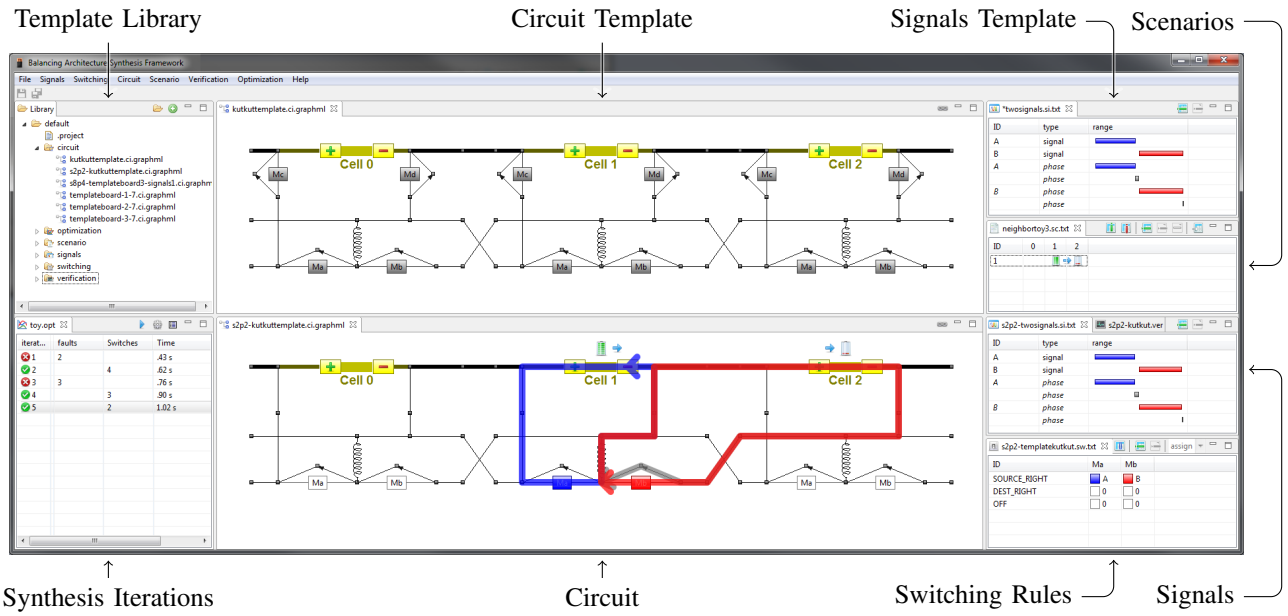


Fig. 3: Illustration of the design tool that enables a synthesis of active cell balancing architectures for battery packs.

of a manual process. The experimental results show some possible templates which can be easily derived from existing circuits and signal patterns, using domain knowledge. Here, the circuit templates represent general switching networks while the signal templates rely on tried and tested patterns. The templates might also be automatically generated which, however, is not in the scope of this paper.

Given the circuit and signal templates, the optimal architectures are determined within an iterative process. The resulting circuit is based on its template where specific MOSFETs might be removed to reduce their overall number. Correspondingly, the resulting signals are derived from the corresponding template by considering only those signals that are used by the switching rules. The resulting switching rules are the key element as they define which signals and MOSFETs are used, implicitly encoding the optimization objectives such as the number of required MOSFETs.

D. Synthesis Approach

The proposed iterative synthesis approach for active balancing architectures is illustrated in Figure 4. One iteration consists of two steps: In the first step, a *search engine* determines a potential solution. In the second step, the proposed solution is verified. The synthesis is carried out for exactly one circuit template and signal template. That means, in case of multiple templates, all combinations of circuits and signals can be synthesized and compared as it is done in the experimental results in Section IV.

In the first step, the search engine uses a SAT solver to determine potential active balancing architectures. For this purpose, a set of binary variables as well as linear constraints is given in Section III to define the search space. Linear constraints are used instead of pure Conjunctive Normal Form (CNF) to allow a higher expressiveness. It should be mentioned

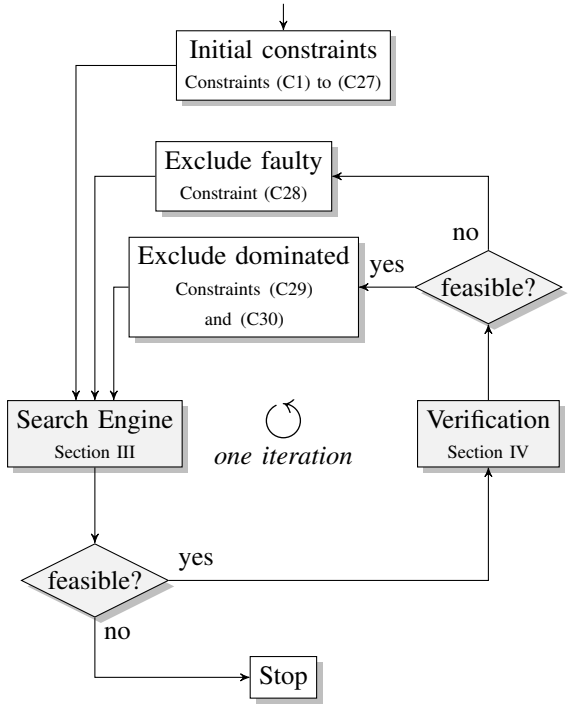


Fig. 4: Illustration of the iterative synthesis approach for active cell balancing architectures, consisting of two major steps: the search engine and verification.

that technically SAT solvers that are capable of handling these constraints are also known as Pseudo-Boolean (PB) solvers. In practice, SAT solvers can be extended to handle these linear constraints directly or via a translation into CNF [4].

The search space is constrained in a way such that each feasible active balancing architecture equals a variable assignment that satisfies all constraints. On the other hand,

not each feasible solution of the constrained problem equals a feasible active balancing architecture. This is due to the fact that prohibitive rules such as the exclusion of short circuits cannot be efficiently defined as SAT problem. Note that prohibitive rules might be formalized using Quantified Boolean Formula (QBF) [5] which are extending SAT with universal and existential quantifiers. However, known solvers for QBF can only address problems about two orders of magnitude smaller than SAT, see [5], making it impossible to solve the discussed problem in a reasonable amount of time.

Since the original SAT problem does not exclude each potentially infeasible balancing architecture, a verification step becomes necessary which is detailed in Section IV. Using a graph-based verification approach from [3], infeasible balancing architectures can be identified and the reasons for the infeasibility are subsequently excluded from the search space. This is done by extending the SAT problem with additional constraints that prohibit specific undesired flows such as short circuits. On the other hand, if the balancing architecture is determined to be feasible by the verification, the objectives are obtained and all dominated solutions are excluded from the search space. Since the considered objectives are linear, this is simply achieved by extending the problem with a set of additional constraints.

With this iterative approach, the synthesis is carried out until the remaining search space is empty, i.e., the search engine does not find any feasible solution anymore, and all optimal balancing architectures are determined.

Note that the approach shares similarities with Satisfiability Modulo Theories (SMT) [6] where a SAT solver is extended with background theories to address first-order logic problems. For the discussed synthesis of balancing architectures, the problem is not fully described by first-order logic and, thus, the approach has to rely on a loose coupling of the proposed search engine and verification. Nevertheless, the proposed synthesis follows the *eager* SMT approach, see [6], where the SAT solver proposes a solution that is either accepted by the background theory or rejected with an additional constraint that is added to the set of constraints.

III. Search Engine

The search engine is defined by a set of linear constraints which are detailed in this section. Given a circuit template and signal template, a set of binary variables is introduced that encodes the synthesis results, i.e, the resulting circuit, switching rules, and used signals. In the following, Constraints (C1) to (C27) define *necessary but not sufficient* rules such that each feasible balancing architecture is also a feasible solution to the search problem. On the other hand, not each feasible solution to the search problem is also a feasible balancing architecture, making the verification step in Section IV necessary.

The efficient definition of the search engine requires a transformation of the circuit template to a graph structure as illustrated in Figure 5. The graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ comprises the set of vertices $\mathcal{V}$ and the set of directed and undirected edges $\mathcal{E}$, respectively. In order not to lose information regarding the original circuit, different types of edges and vertices have to be introduced. The vertices are defined as follows:

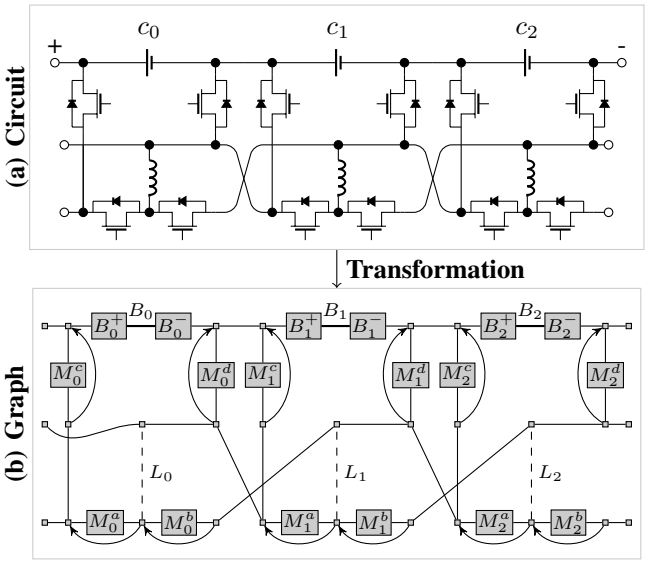


Fig. 5: Illustration of the transformation of the balancing circuit template to a graph representation.

- Each MOSFET of the original circuit is transformed into a corresponding vertex. The set of MOSFET vertices is given as $\mathcal{V}_M \subset \mathcal{V}$ where each MOSFET M_n^x corresponds to a vertex in $\mathcal{V}_M$.
- Each battery cell is transformed into two vertices (and one edge). The vertices are given as $\mathcal{V}_B \subset \mathcal{V}$ with $\mathcal{V}_B = \mathcal{V}_{B^+} \cup \mathcal{V}_{B^-}$ where $\mathcal{V}_{B^+}$ represents the set of positive terminals and $\mathcal{V}_{B^-}$ the set of negative terminals, respectively.
- All remaining vertices $\mathcal{V} \backslash (\mathcal{V}_B \cup \mathcal{V}_M)$ are used for linking elements and correspond to connecting terminals in the circuit.

The edges are defined as follows:

- Each inductor of the original circuit is transformed into a corresponding edge. The set of inductor edges is $\mathcal{E}_L \subset \mathcal{E}$ where each inductor L_n corresponds to an undirected edge in $\mathcal{E}_L$.
- Each battery cell is transformed into an undirected egde that is connected by the respective positive and negative vertices. The set of battery edges is defined as $\mathcal{E}_B \subset \mathcal{E}$.
- Each diode is transformed into a directed edge. The set of diodes is given as $\mathcal{E}_D \subset \mathcal{E}$.
- All remaining edges $\mathcal{E} \backslash (\mathcal{E}_B \cup \mathcal{E}_L \cup \mathcal{E}_D)$ represent electrical connections that correspond to the original circuit.

Given a circuit in a suitable netlist format such as Simulation Program with Integrated Circuit Emphasis (SPICE) [7], the corresponding graph can be obtained with an appropriate parser. It is also possible to model the circuit manually with the transformation rules given above.

A. Switching and Rules

The first set of variables and constraints defines the state and switching rules for MOSFETs. The introduced variables use the following scheme:

- $\mathbf{x}_{\mathcal{T},M,\sigma} \in \{0,1\}$ is 1 if MOSFET M in scenario $\mathcal{T}$ is controlled by signal $\sigma \in \Sigma \cup \{0,1\}$, 0 otherwise.

- $\mathbf{y}_{s,\mathcal{M}^x,\sigma} \in \{0,1\}$ is 1 if the MOSFETs $\mathcal{M}^x$ in state s are controlled by signal $\sigma \in \Sigma \cup \{0,1\}$, 0 otherwise.
- $\mathbf{z}_{\mathcal{T},M,\phi} \in \{0,1\}$ is 1 if MOSFET M in scenario $\mathcal{T}$ is closed in phase $\phi \in \Phi$, 0 otherwise.

Here, the $\mathbf{x}$ variables represent the actual state of a MOSFET for a certain module and scenario while the $\mathbf{y}$ variables encode the switching rules. The variables $\mathbf{z}$ encode the relation between MOSFETs and phases. The constraints that define the boundaries for these variables are as follows: $\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, M \in \mathcal{M}$:

$$\sum_{\sigma \in \Sigma \cup \{0,1\}} \mathbf{x}_{\mathcal{T},M,\sigma} = 1 \tag{C1}$$

$\forall s \in S, \mathcal{M}^x \subseteq \mathcal{M}$:

$$\sum_{\sigma \in \Sigma \cup \{0,1\}} \mathbf{y}_{s,\mathcal{M}^x,\sigma} = 1 \tag{C2}$$

$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, n \in N, s = c(\mathcal{T},n), \mathcal{M}^x \subseteq \mathcal{M}, M_n^x \subseteq \mathcal{M}^x, \sigma \in \Sigma \cup \{0,1\}$:

$$\mathbf{x}_{\mathcal{T},M_n^x,\sigma} = \mathbf{y}_{s,\mathcal{M}^x,\sigma} \tag{C3}$$

$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, M \in \mathcal{M}, \phi \in \Phi$:

$$-\mathbf{z}_{\mathcal{T},M,\phi} + \sum_{\sigma \in \phi \cup \{1\}} \mathbf{x}_{\mathcal{T},M,\sigma} \geq 0 \tag{C4}$$

$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, M \in \mathcal{M}, \phi \in \Phi, \sigma \in \phi \cup \{1\}$:

$$-\mathbf{x}_{\mathcal{T},M,\sigma} + \mathbf{z}_{\mathcal{T},M,\phi} \geq 0 \tag{C5}$$

Constraint (C1) defines that each MOSFET in a certain scenario is controlled by exactly one signal or it is closed or open. Constraint (C2) states that for each MOSFET type and state of a module, exactly one signal is applied or the respective MOSFET is statically closed or open. Thus, this constraint defines the unambiguity of the resulting switching rules table. Constraint (C3) ensures that the switching rules are applied to all MOSFETs in all scenarios. Constraints (C4) and (C5) ensure that a MOSFET is considered to be closed in a certain phase if and only if it is controlled by a signal that is active in the phase, or it is statically closed. In other words, these two constraints define in which phases the MOSFETs are closed or open, respectively, by setting the appropriate $\mathbf{z}$ variable values.

B. Flow

Charge flow is encoded via the edges $\mathcal{E}$ of the graph $\mathcal{G}$. For this purpose, additional variables are introduced that model charge flow for a certain transfer and phase:

- $\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} \in \{0,1\}$ is 1 if there is a flow in a certain scenario, transfer, and phase on the edge $e \in \mathcal{E}$ in the direction from v to $\tilde{v}$, 0 otherwise.

The flow in the circuit is modeled with the following constraints:
$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, v \in \mathcal{V}$:

$$\sum_{e=(v,\tilde{v}) \in \mathcal{E}} \mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} = \sum_{e=(\tilde{v},v) \in \mathcal{E}} \mathbf{f}_{\mathcal{T},T,\phi,e=(\tilde{v},v)} \tag{C6}$$

$$\sum_{e=(v,\tilde{v}) \in \mathcal{E}} \mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} \leq 1 \tag{C7}$$

$$\sum_{e=(\tilde{v},v) \in \mathcal{E}} \mathbf{f}_{\mathcal{T},T,\phi,e=(\tilde{v},v)} \leq 1 \tag{C8}$$

$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, e = (v,\tilde{v}) \in \mathcal{E}, e = (\tilde{v},v) \in \mathcal{E}$:

$$\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + \mathbf{f}_{\mathcal{T},T,\phi,e=(\tilde{v},v)} \leq 1 \tag{C9}$$

Constraint (C6) is the flow conservation constraint that ensures that the sum of incoming flows equals the sum of outgoing flows. Constraints (C7) and (C8) ensure that all flows are disjunct and not merging and splitting with other flows. Constraint (C9) forbids trivial circular flows between two vertices that are connected by an undirected edge e.

Additionally, a charge flow is only possible via MOSFETs that are closed in a certain phase as modeled with the following constraints:
$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, v \in \mathcal{V}_M, v \to M, e = (v,\tilde{v}) \in \mathcal{E}$:

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + \mathbf{z}_{\mathcal{T},M,\phi} \geq 0 \tag{C10}$$

$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, e = (v,\tilde{v}) \in \mathcal{E}_D, e \to M$:

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + (1 - \mathbf{z}_{\mathcal{T},M,\phi}) \geq 0 \tag{C11}$$

Constraint (C10) states that a flow is only possible via a MOSFET vertex in a certain phase if the MOSFET is closed in the respective phase. The notation $v \to M$ obtains the corresponding MOSFET for the considered vertex. Analogously, Constraint (C11) states that a flow via a diode is only existent if the corresponding MOSFET is open in the considered phase. The notation $e \to M$ obtains the corresponding MOSFET for the considered diode edge.

C. Predicates

The problem with the encoding of flows is that cycles are not prohibited, which might exist in parallel to the desired flows. In fact, cycles satisfy the requirement that the incoming and outgoing flows are equal. If it is required that a certain flow passes an edge that represents a battery, diode, or inductor (see the next subsection), it is not enough that the respective binary variable is 1. This is because the edge might be part of a cycle that is not part of the actual flow. Therefore, predicate variables are introduced for each vertex to ensure that flows pass certain edges.

Predicates are used to identify if a flow passed a certain element like a battery, diode, or inductor. For this purpose, the following additional variables are introduced:

- $\mathbf{p}_{\mathcal{T},T,\phi,v,X} \in \{0,1\}$ is 1 if the predicate $X \in \{D,B,L\}$ (diode, battery, or inductor) is active in vertex v for a given scenario $\mathcal{T}$, transfer T, and phase ϕ, 0 otherwise.

The constraints that determine if a predicate is active, are defined as follows:
$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, X \in \{D,B,L\}, v \in \mathcal{V}$:

$$-\mathbf{p}_{\mathcal{T},T,\phi,v,X} + \sum_{e=(\tilde{v},v) \in \mathcal{E}_X} \mathbf{f}_{\mathcal{T},T,\phi,e=(\tilde{v},v)} + \sum_{e=(\tilde{v},v) \in \mathcal{E} \setminus \mathcal{E}_X} \mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} \cdot \mathbf{p}_{\mathcal{T},T,\phi,\tilde{v},X} \geq 0 \tag{C12}$$

Constraint (C12) ensures that a predicate is only active if the edge of the incoming flow represents the predicate or the incoming flow already carries the predicate. The first

sum defines whether the specific edge was passed while the second sum defines if the predicate is already *carried* by the flow. Note that the product of the two binary variables in Constraint (C12) has to be linearized. This is done by substitution and additional variables and constraints with $a \cdot b = c$ and constraints $a+b-2c \geq 0$ and $c-a-b+1 \geq 0$.

D. Transfer

Each phase has different requirements regarding the charge flow which has to be represented by the constraints. Since for more complex transfer schemes there might be more than four phases, subsets of all phases are defined as follows:

- $\Phi_{\mathrm{ch}} \subseteq \Phi$ - Charging phases
- $\Phi_{\mathrm{fr}} \subseteq \Phi$ - Freewheeling phases
- $\Phi_{\mathrm{di}} \subseteq \Phi$ - Discharging phases
- $\Phi_{\mathrm{bl}} \subseteq \Phi$ - Blocking phases

The order of these phases is fixed while particularly the cardinality of the freewheeling phases might be higher than one, as can be seen in the experimental results.

Depending on the considered phase, certain elements of the graph $\mathcal{G}$ might be removed when defining the flow variables $\mathbf{f}$ and predicate variables $\mathbf{p}$. This filtering is summarized in the following:

- Φ_{ch} - Remove diode edges $\mathcal{E}_D$ and remove all battery edges $\mathcal{E}_B$ except for the source cell n for each transfer $T = (n, \tilde{n})$ since charging is not using diodes and only involves the source cell.
- Φ_{fr} - Remove battery edge in $\mathcal{E}_B$ for sender n for each transfer $T = (n, \tilde{n})$ since freewheeling never involves the source cell.
- Φ_{di} - Remove diode edges $\mathcal{E}_D$ and remove all battery edges $\mathcal{E}_B$ except for the destination cell $\tilde{n}$ for each transfer $T = (n, \tilde{n})$ since discharging is not performed via diodes and only to the destination cell.
- Φ_{bl} - Remove all battery edges $\mathcal{E}_B$ except for the receiver cell $\tilde{n}$ for each transfer $T = (n, \tilde{n})$ since the blocking phase is discharging to the destination cell only.

Additionally, all vertices and edges outside the domain $d(T)$ of the considered transfer are removed. This is due to the fact that the flows of a transfer shall never be routed via a circuit that is outside the domain in order not to conflict with other transfers.

The constraints for the charging phases are defined as follows:

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T = (i,j) \in \mathcal{T}, \phi \in \Phi_{\mathrm{ch}}, e = (v^-, v^+) \in \mathcal{E}_B$:

$$\mathbf{f}_{\mathcal{T},T,\phi,e=(v^-,v^+)} = 1 \tag{C13}$$

$$\mathbf{p}_{\mathcal{T},T,\phi,v^+,L} = 0 \tag{C14}$$

$$\mathbf{p}_{\mathcal{T},T,\phi,v^-,L} = 1 \tag{C15}$$

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T \in \mathcal{T}, \phi \in \Phi_{\mathrm{ch}}$:

$$\sum_{e=(v,\tilde{v}) \in \mathcal{E}_L} \mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} = 1 \tag{C16}$$

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T \in \mathcal{T}, e = (v,\tilde{v}) \in \mathcal{E}_L, \phi \in \Phi_{\mathrm{ch}}, \tilde{\phi} \in \Phi \setminus \Phi_{\mathrm{ch}}$:

$$\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} = \mathbf{f}_{\mathcal{T},T,\tilde{\phi},e=(v,\tilde{v})} \tag{C17}$$

During charging, Constraint (C13) ensures a charge flow from the positive terminal vertex to the negative terminal vertex outside of the battery cell. Constraints (C14) and (C15) ensure that this flow passes one inductor, i.e, the predicate is not active in the positive terminal node and it is active once the flow reached the negative terminal node. Constraint (C16) mandates that exactly one inductor is used for the charge transfer. This is an optional constraint that might be relaxed, depending on the balancing architecture requirements. Constraint (C17) states that all inductors, that were charged in the charging phase, have to be discharged during all remaining phases. Correspondingly, inductors that were not charged are not discharged.

The constraints for the freewheeling and blocking phases are defined as follows:

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T \in \mathcal{T}, e = (v,\tilde{v}) \in \mathcal{E}_L, \phi \in \Phi_{\mathrm{fr}} \cup \Phi_{\mathrm{bl}}$:

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + (1 - \mathbf{p}_{\mathcal{T},T,\phi,\tilde{v},D}) \geq 0 \tag{C18}$$

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + \mathbf{p}_{\mathcal{T},T,\phi,v,D} \geq 0 \tag{C19}$$

Constraints (C18) and (C19) ensure that the discharging of the inductor during the freewheeling phase Φ_{fr} and discharging with blocking phase Φ_{bl} is performed via a diode.

The constraints for the discharging and blocking phases are defined as follows:

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T \in \mathcal{T}, e = (v,\tilde{v}) \in \mathcal{E}_L, \phi \in \Phi_{\mathrm{di}} \cup \Phi_{\mathrm{bl}}$:

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + (1 - \mathbf{p}_{\mathcal{T},T,\phi,\tilde{v},B}) \geq 0 \tag{C20}$$

$$-\mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})} + \mathbf{p}_{\mathcal{T},T,\phi,v,B} \geq 0 \tag{C21}$$

Constraints (C20) and (C21) ensure that the discharging in phase Φ_{di} and Φ_{bl} is done via the receiving battery cell.

E. Synchronization-free Transfer

One additional requirement for the considered architecture is that a transfer is controlled by a single module. That means for a certain transfer, all PWM signals are controlled by exactly one module. This has the advantage that no synchronization with very high precision between two modules becomes necessary in a distributedly controlled architecture. For this purpose, the introduction of the following variables becomes necessary:

- $\mathbf{w}_{\mathcal{T},n} \in \{0,1\}$ is 1 if module $n \in N$ contains at least one MOSFET in scenario $\mathcal{T}$ that is controlled by a signal $\sigma \in \Sigma$, 0 otherwise.

The corresponding constraints are defined as follows:

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, n \in N$:

$$-|\mathcal{M}^x \subseteq \mathcal{M}| \cdot (1 - \mathbf{w}_{\mathcal{T},n}) + \sum_{M_n \in \mathcal{M}} (\mathbf{x}_{\mathcal{T},M_n,0} + \mathbf{x}_{\mathcal{T},M_n,1}) \geq 0 \tag{C22}$$

$\forall \mathcal{T} \in \mathcal{T}_{\mathrm{all}}, T \in \mathcal{T}$:

$$\sum_{n \in d(T)} \mathbf{w}_{\mathcal{T},n} \leq 1 \tag{C23}$$

Constraint (C22) states that a module n needs to control its MOSFETs with a PWM signal if not all its MOSFETs are either in the closed or open state. Constraint (C23) ensures that for each transfer only a single cell is using PWM controlled MOSFETs such that no synchronization between modules is necessary for the respective PWM signals.

F. Objectives

Additional variables are introduced that are later used for the objective functions:

- $\mathbf{u}_{\mathcal{M}^x,\sigma} \in \{0,1\}$ is 1 if a certain MOSFET type is statically controlled by exactly one signal $\sigma \in \{0,1\}$ in all states, 0 otherwise. If this is the case, the respective MOSFET can be removed from the circuit.
- $\mathbf{v}_{\mathcal{M}^x} \in \{0,1\}$ is 1 if a certain MOSFET type is statically controlled by either 0 or 1 in all states, 0 otherwise. If this is the case, the respective gate-driver does not have to be capable of applying PWM signals.

The constraints that set these variables accordingly are defined as follows:
$\forall \mathcal{M}^x \subseteq \mathcal{M}, \sigma \in \{0,1\}$:

$$-|S| \cdot \mathbf{u}_{\mathcal{M}^x,\sigma} + \sum_{s \in S} \mathbf{y}_{s,\mathcal{M}^x,\sigma} \geq 0 \tag{C24}$$

$$\sum_{s \in S} \mathbf{y}_{s,\mathcal{M}^x,\sigma} - \mathbf{u}_{\mathcal{M}^x,\sigma} \leq |S| - 1 \tag{C25}$$

$\forall \mathcal{M}^x \subseteq \mathcal{M}$:

$$-|S| \cdot \mathbf{v}_{\mathcal{M}^x} + \sum_{s \in S} (\mathbf{y}_{s,\mathcal{M}^x,0} + \mathbf{y}_{s,\mathcal{M}^x,1}) \geq 0 \tag{C26}$$

$$\sum_{s \in S} (\mathbf{y}_{s,\mathcal{M}^x,0} + \mathbf{y}_{s,\mathcal{M}^x,1}) - \mathbf{v}_{\mathcal{M}^x} \leq |S| - 1 \tag{C27}$$

Constraints (C24) and (C25) determine if a MOSFET is statically open or closed in all states. If this is the case, the respective MOSFET can be removed from the circuit. In case of $\sigma \in 0$, the MOSFET is removed breaking the connection. In case of $\sigma \in 1$, a permanent connection replaces the MOSFET. Constraints (C26) and (C27) determine if a MOSFET is only switched between open or closed in all states, i.e., it is not controlled by a PWM signal in any state. If this is the case, it is beneficial in terms of cost, volume, and energy efficiency.

IV. Verification

In the following, our verification step is discussed which relies on a graph-based methodology outlined in Figure 6. Given the circuit as a graph, the graph is modified for each charge transfer and phase, depending on the switching rules. Here, vertices are removed that represent MOSFETs which are open during the respective phase. In the following, possible charge flows on the graphs are identified by a graph search algorithm. Finally, the flows are evaluated using a predefined rule set that ensures the correctness of the designed circuit. This graph-based verification approach is proposed in [3].

Using the verification in combination with the proposed search engine ensures the synthesis of optimal balancing architectures. This is done by extending the search engine with additional constraints, depending on the results of the verification. In case there exist faults in the balancing architecture, the verification provides counter examples. Learning from these counter examples, faults can be excluded from the search engine. In case the verification confirms that the current balancing architecture is feasible, all dominated solutions in terms of their objectives can be excluded from the search engine by adding a set of corresponding constraints.

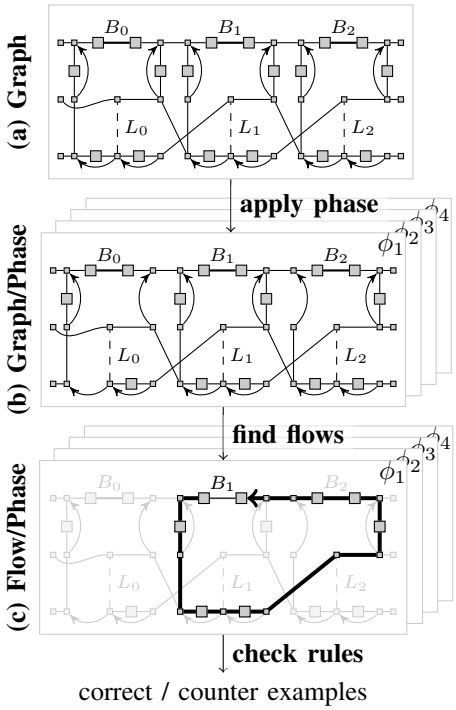


Fig. 6: Illustration of the verification methodology as proposed in [3].

An example for this process is illustrated in Figure 7. Here, a balancing circuit with four MOSFETs is optimized until the solution with two MOSFETs from Figure 2 is obtained. Note that this synthesis example solely considers a single scenario where charge is transfered from c_1 to c_2. The resulting switching rules are given in Table II where c_0 is in state OFF, c_1 in state SRC_RIGHT, and c_2 in state DEST_RIGHT.

The constraints that exclude faulty flows as well as the constraints that exclude dominated solutions are introduced in the following.

A. Exclude Faulty

In case an obtained balancing architecture from the search engine is not feasible, the verification determines counter examples for each respective phase ϕ in form of one or more faulty flows Π_{all}. Each flow is a cycle in the circuit graph, consisting of the respective edges. Note that due to

TABLE II: Resulting switching rules for the example in Figure 7 with the signals from Figure 2b for single scenario where charge is transfered from c_1 to c_2.

State	$\mathcal{M}^a$	$\mathcal{M}^b$	$\mathcal{M}^c$	$\mathcal{M}^d$
SRC_RIGHT	σ_1	σ_2	1	1
DEST_RIGHT	0	0	1	1
OFF	0	0	1	1

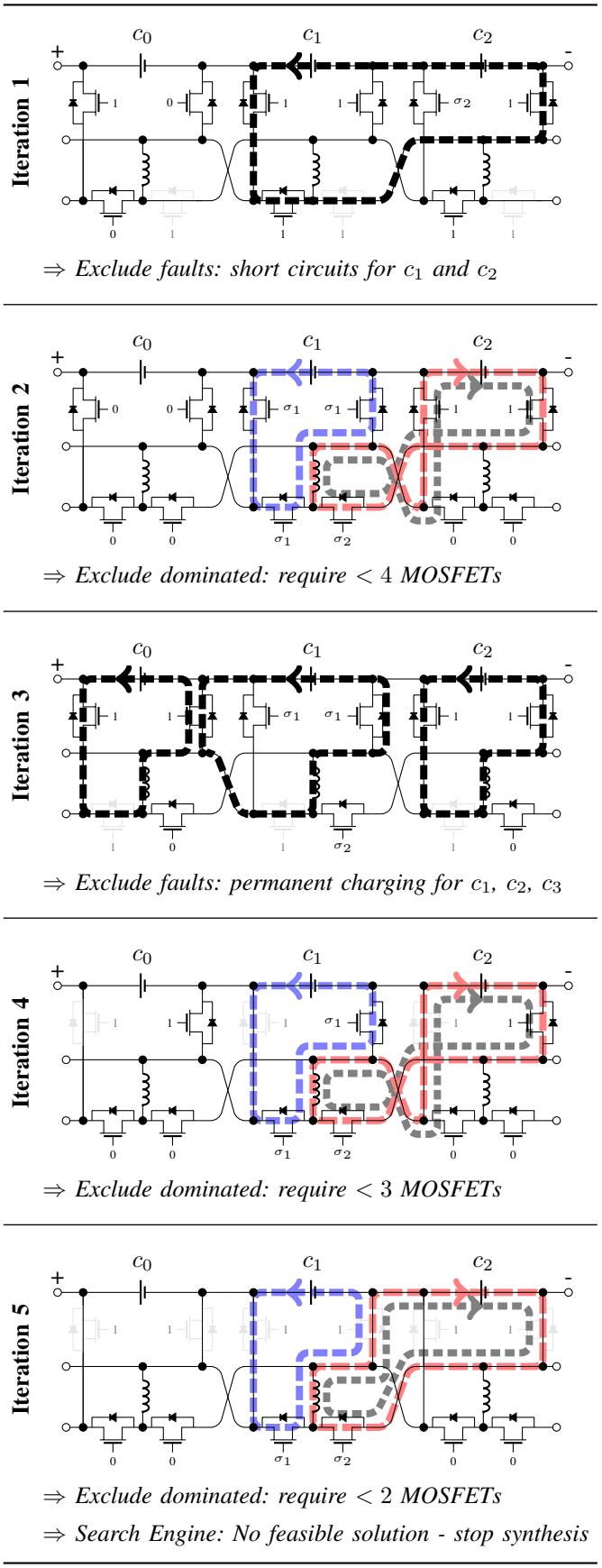


Fig. 7: Illustration of an example circuit template and its iterative synthesis, excluding faulty solutions and dominated solutions. This example uses the signals from Figure 2b.

the definition of constraints in the search engine, there has to exist a flow for each transfer and phase. Therefore, faults that result in non-existent flows are prohibited.

The constraint that excludes faulty flows is defined as follows:
$\forall \mathcal{T} \in \mathcal{T}_{\text{all}}, T \in \mathcal{T}, \phi \in \Phi, (\mathcal{T}, T, \phi) \to \Pi_{\text{all}}, \Pi \in \Pi_{\text{all}}$:

$$\sum_{M \in \mathcal{M}_\Pi} (1 - \sum_{\sigma \in \phi \cup \{1\}} \mathbf{x}_{\mathcal{T},M,\sigma}) + \sum_{e=(v,\tilde{v}) \in \mathcal{E}_L \cap \mathcal{E}_\Pi} (1 - \mathbf{f}_{\mathcal{T},T,\phi,e=(v,\tilde{v})}) \geq 0 \qquad \text{(C28)}$$

Constraint (C28) excludes the flow Π in the considered scenario, transfer, and phase from the search engine. This is achieved by enforcing that at least one of the MOSFETs is not closed in phase ϕ or the flow does not use the inductor. Here, $\mathcal{M}_\Pi$ represents all MOSFETs that are used by the flow while $\mathcal{E}_\Pi$ represents all used edges of the flow. To increase the effectiveness of this approach, the flows should contain the minimal amount of MOSFETs that are necessary. That means, if a flow is also possible via a diode, the respective MOSFET should be removed from the flow as it does not have any influence on the infeasibility of the solution. An example for this is the flow in Iteration 1 of Figure 7 where M_2^d (the MOSFET on the most right side) can be excluded and instead the flow is possible via the corresponding diode.

B. Exclude Dominated / Objectives

In case a balancing architecture is feasible, dominated solutions should be excluded from the further search. Here, we consider the following three objective functions:

$$\min\colon \quad o_1(\mathbf{y}) = \sum_{\mathcal{M}^x \subseteq \mathcal{M}} (1 - \mathbf{u}_{\mathcal{M}^x,0} + \mathbf{u}_{\mathcal{M}^x,1}) \qquad \text{(O1)}$$

$$\min\colon \quad o_2(\mathbf{y}) = \sum_{\mathcal{M}^x \subseteq \mathcal{M}} (1 - \mathbf{v}_{\mathcal{M}^x}) \qquad \text{(O2)}$$

$$\min\colon \quad o_3(\mathbf{y}) = \sum_{s \in S} \sum_{\mathcal{M}^x \subseteq \mathcal{M}} \sum_{\sigma \subseteq \Sigma} \mathbf{y}_{s,\mathcal{M}^x,\sigma} \qquad \text{(O3)}$$

Objective (O1) minimizes the number of required MOSFETs in the balancing circuit. This objective optimizes the cost, volume, and energy consumption of the resulting circuit. Objective (O2) minimizes the number of MOSFETs that have to be controlled by a PWM signal in any state. This objective optimizes the cost and energy consumption as the gate-drivers for PWM control are larger and require more power. Objective (O3) minimizes the number of PWM signals in all states, i.e., the absolute number of PWM signals in the resulting switching rules table. This objective has an influence on the energy consumption as controlling a MOSFET in PWM mode requires more energy than having it statically closed or open.

The exclusion of dominated solutions is achieved by the following additional constraints:
$\forall i \in \{1,...,3\}, j \in \mathbb{N}(\text{iteration})$:

$$o_i(\mathbf{y}) - Z \cdot \mathbf{d}_{i,j} < o_{i,j} \qquad \text{(C29)}$$

$\forall j \in \mathbb{N}(\text{iteration})$:

$$\sum_{i=1}^{3} \mathbf{d}_{i,j} < 3 \qquad \text{(C30)}$$

In Constraint (C29), Z is a large number and $o_{i,j}$ is the determined objective value for objective i in iteration j. If the variable $\mathbf{d}_{i,j}$ becomes 0, the constraint has to satisfy $o_i(\mathbf{y}) < o_{i,j}$ such that this specific objective is improved. Constraint (C30) ensures that at least one $\mathbf{d}_{i,j}$ becomes 0 such that at least one objective is improved. As a result, all dominated solutions are excluded. Note that this approach allows an arbitrary number of linear objectives while for the example in Figure 7 only Objective (O1) was used.

V. Experimental Results

In the following, experimental results are presented that give evidence of the applicability of the proposed approach. For this purpose, three circuit templates and two signal templates have been used to obtain optimal results. All experiments have been carried out on an Intel Core i5 (4300U) CPU with 1.90 GHz and 8 GB RAM.

For the search engine, the SAT4J solver [8] has been used. Note that for the proposed problem, SAT4J has been always faster than a commercial mathematical solver like GUROBI [9] or CPLEX [10] which might be also used for implementing the search engine. This is due to the fact that solvers based on backtracking like SAT4J perform better than classical Integer Linear Programming (ILP) solvers in a scenario where constrained problems are solved incrementally. For the verification step, our implementation from [3] has been applied that relies on the graph library JUNG [11].

A. Scenarios and Templates

The design goal of our experimental results is a balancing architecture that supports the direct charge transfer between cells that are not adjacent. Here, we focus on architectures that use inductors as energy storage element, promising a high efficiency. As reference implementation, we use the manually designed architecture from [3] which comes closest to the defined requirements (it additionally requires synchronization for the PWM signals which is a major obstacle for efficient large-scale implementation). The considered scenarios look as follows:

$$\begin{aligned}\mathcal{T}_{\text{all}} = \{&\{(1,3),(4,7)\},\\ &\{(3,1),(7,4)\},\\ &\{(1,2),(4,3),(6,7)\},\\ &\{(2,1),(3,4),(7,6)\}\}\end{aligned} \tag{6}$$

These scenarios are sufficient as they contain all relevant transfer distances and combinations. Note that each determined optimal balancing architecture has been further verified with a full set of 239 possible scenarios for a number of 7 cells.

The circuit and signal templates are illustrated in Figure 8. The circuit templates are the result of an extension of existing circuits from literature, applying additional domain knowledge. Note that for each circuit template, exactly one module is depicted and by attaching seven of them in series, the full circuit for the synthesis is obtained. The first template circuit module that consists of 20 MOSFETs is inspired by the architecture in [12] which is illustrated in Figure 2a. Here, the original circuit with 2 MOSFETs is extended with 18

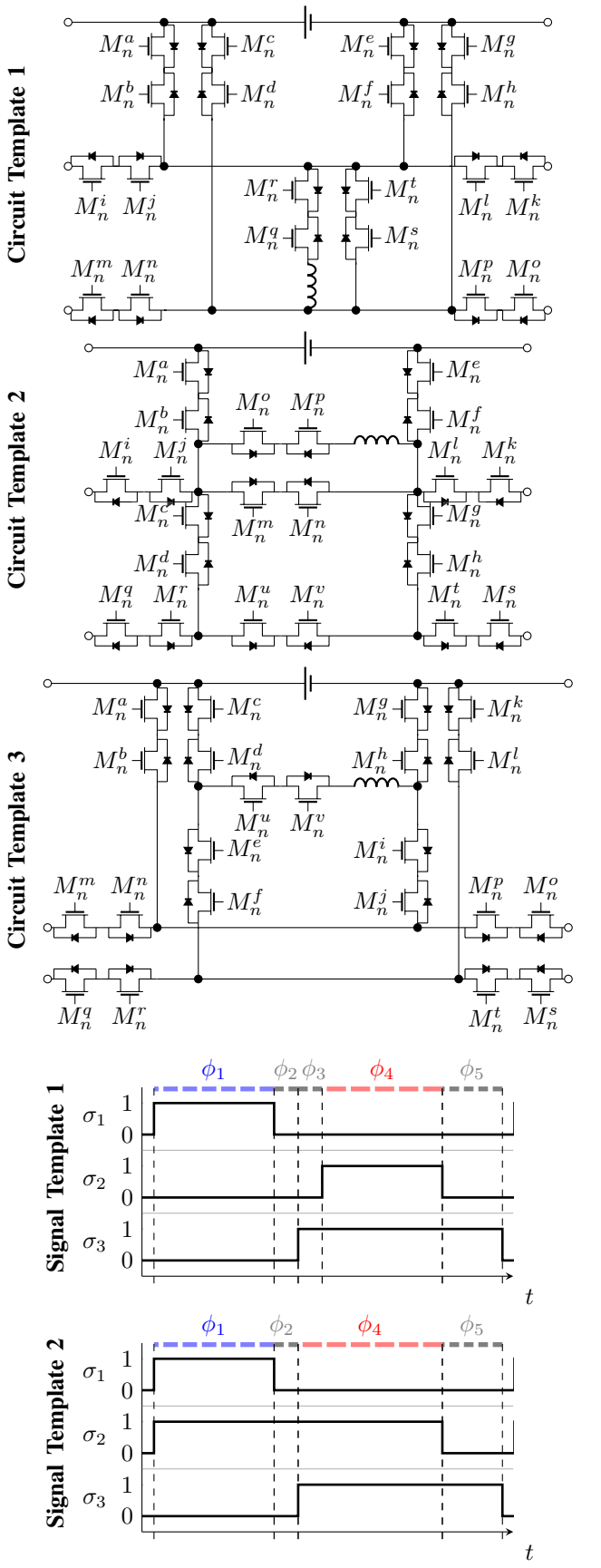


Fig. 8: Circuit and signal templates that are used for the experimental results.

additional MOSFETs. The second template is motivated by the development board in [3]. While the original circuit consists of 12 MOSFETs, the extended circuit template has 22 MOSFETs. The third and last template is inspired by the combination of the first two template modules and comprises 22 MOSFETs. It should be noted that designing the template circuits currently requires a certain domain knowledge while an automatic design of template circuits is part of future work.

The first signal template in Figure 8 is a simplified scheme that is deduced from the manually determined architecture in [3] by reducing the number of signals to three. Correspondingly, the second signal template is an adaptation of the first template. Note that the signals in Figure 2b are implicitly contained in the first signal template and can be obtained by not using signal σ_3. Corresponding to the circuit templates, the generation of signal templates requires domain knowledge and automatic approaches might be applied. However, during various tests, the two depicted signal templates showed to be effective and simple at the same time. Complex signal templates should anyway be avoided in order to prohibit overly complex PWM patterns and switching rules.

B. Synthesis Results

The synthesis has been performed for all combinations of circuit templates and signal templates. This results in six different combinations as outlined in Table III. Despite the fact that three objectives are considered, each circuit and signals pair resulted in exactly one optimal solution instead of multiple Pareto-optimal solutions. This can be explained by the high correlation of the considered objectives where a low value of Objective (O1) results in small values for Objectives (O2) and (O3). Even among all combinations there exists only a single Pareto-optimal solution which is highlighted in Table III.

This optimal balancing architecture is determined for the third template circuit and first signal template. The iterative synthesis process for this specific combination is exemplary illustrated in Figure 9. It can be seen that until iteration 34, all balancing architectures obtained by the search engine are infeasible as determined by the verification. This is a common case for all combinations of circuits and signals where in the beginning the iterative synthesis approach first has to exclude several faulty solutions before the objectives are effectively optimized. From the 2272 s runtime for this specific optimal circuit and signals combination, 599 s are spent in the verification step. This means, the verification requires approximately one quarter of the overall runtime while the search engine requires three quarters. Such an approximate ratio between the runtime of the search and verification is observed for all combinations.

TABLE III: Results of all combinations of template circuits and signals. The optimal result is highlighted and better in all objectives compared to a state-of-the-art manual design from [3].

Inputs		Constraints		Time	Iter-	Objectives		
Circuit	Signals	#cons	#vars	(s)	ations	(O1)	(O2)	(O3)
1	1	86545	58980	1200	189	9	4	8
1	2	70059	47520	731	138	9	6	10
2	1	88783	59282	678	110	10	4	8
2	2	71757	48128	364	98	10	6	10
3	**1**	**93239**	**63102**	**2272**	**203**	**8**	**4**	**8**
3	2	75527	50858	1813	173	8	6	10
Manual design [3]						12	6	20

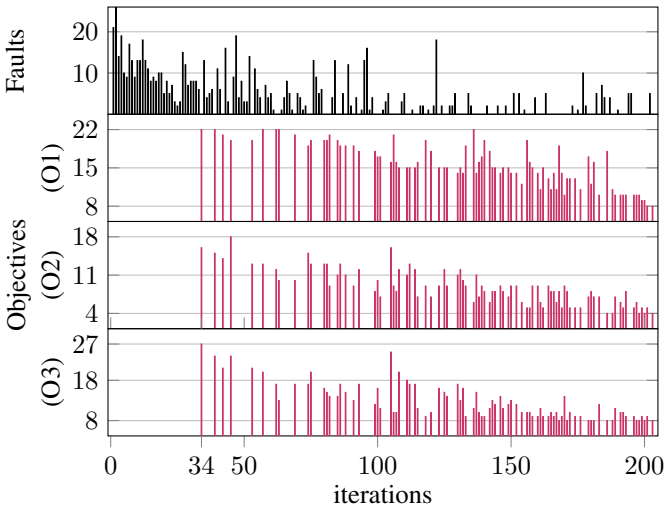


Fig. 9: Illustration of the iterative synthesis for the third template circuit and first signals template, see Table III. For each infeasible architecture, the bars (Faults) show the faults and the objectives are not evaluated. For each feasible architecture, no faults exist and the bars indicate the values of Objectives (O1) to (O3).

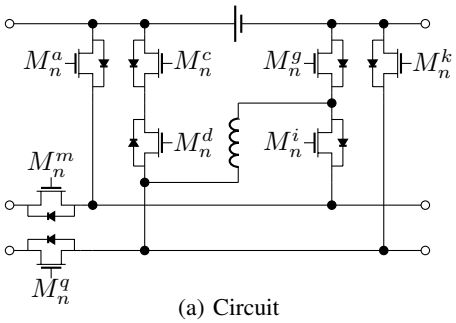
Overall, the synthesis for all six combinations requires 7058 s and a subsequent verification of the optimal solution for all 239 possible scenarios requires 91.3 s. Thus, the synthesis is performed within two hours, which is a completely acceptable time frame considering the complexity of the task as well as the value of the results.

The optimal balancing architecture is illustrated in Figure 10. Compared to a manually designed solution that is listed in Table III, the synthesized solution is better in all considered objectives. In fact, the synthesized solution contains certain asymmetries, considering MOSFETs M_n^c, M_n^d, M_n^g, and M_n^i and the connection of the inductor. These asymmetries would be very untypical for a manual design and in result could hardly be found without a fully automated design process as proposed in this paper.

Exemplary, a charge transfer for the synthesized solution is illustrated in Figure 11. Here, three modules are considered and a transfer from the first to the third cell is shown. This figure illustrates that a single charge transfer is in fact non-trivial and a manual design not easily possible.

C. Performance and Scalability

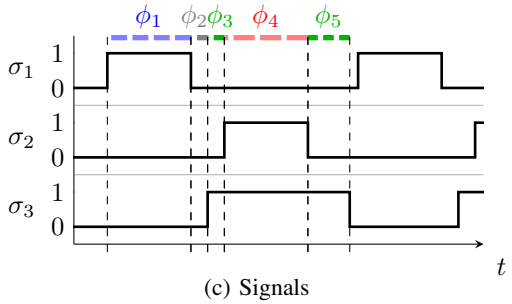
The approach shows a sufficient performance where the considered templates are not trivial and an optimal balancing architecture could be obtained within less than two hours, taking all combinations of templates into account. As outlined in the following, there are several measures that can be taken to further improve the runtime of the proposed approach.



(a) Circuit

State	$\mathcal{M}^a$	$\mathcal{M}^c$	$\mathcal{M}^d$	$\mathcal{M}^g$	$\mathcal{M}^i$	$\mathcal{M}^k$	$\mathcal{M}^m$	$\mathcal{M}^q$
SRC_RIGHT	1	1	0	0	0	1	0	0
SRC_LEFT	0	σ_1	1	σ_1	σ_2	0	1	σ_3
DEST_RIGHT	0	σ_3	1	σ_2	σ_1	0	1	σ_1
DEST_LEFT	1	1	0	0	0	1	0	0
OFF	0	0	0	0	0	0	0	0
BRIDGE	0	0	0	0	0	0	1	1

(b) Switching rules



(c) Signals

Fig. 10: Optimal balancing architecture, consisting of the circuit, switching rules, and signals.

First, the used SAT solver could be replaced by a faster implementation. The currently used solver was chosen due to its easy integration into the existing framework as it is already part of the ECLIPSE framework. Other SAT solvers even make use of multiple processing cores to further speed up the solving time. Additionally, preprocessing methods might be applied to reduce the number of variables and constraints.

Second, the coupling between search engine and verification can be improved corresponding to the *lazy* approach in SMT solving [13]. That means, the verification could check the feasibility of non-complete solutions of the search engine in order to give feedback earlier on faulty flows. This, however, would require several changes for the search engine and verification steps which would have negative influence on the maintainability of the synthesis design tool.

Nevertheless, it is questionable whether using larger templates is a reasonable approach in general. Instead, an automated variation of templates with a manageable amount of MOSFETs (around 20 as in the used templates) should be considered as preferred method as it respects the exponential nature of the used SAT solver.

VI. Related Work

In the following, related work is discussed. First, previous work from the domain of active cell balancing is presented, followed by relevant work in logic and system synthesis.

A. Active Cell Balancing

Cell balancing is an important task performed by Battery Management Systems (BMSs) with the goal to equalize the charge levels of individual cells in a series-connected battery pack. A review of the various cell balancing approaches is done in [14], [15], [16]. Architectures are broadly classified into *passive* (dissipative) and *active* (redistributive) cell balancing.

Passive cell balancing involves dissipation of excess energy of a cell across a resistor until the charge levels are equalized. The approach might be implemented with a fixed resistor as in [17] or as a controlled shunting where the balancing is performed with a switch or relay [18]. The advantage of this technique is its ease of implementation and low cost in comparison with the active cell balancing techniques. However, the major disadvantage of passive cell balancing is the low efficiency since energy is dissipated as heat across the resistors.

In contrast to passive cell balancing, active approaches increase the energy efficiency of battery packs significantly by transferring excess charge from cells with high State-of-Charge (SoC) to cells with low SoC. This is done via temporary energy storage elements such as capacitors, transformers, or inductors.

Capacitor-based cell balancing techniques use capacitors to transfer charge between cells, see [19], [20], [21], [22]. This method is characterized by simple control requirements and low installation space. However, the major disadvantage of this method is the inherent energy loss associated with charging capacitors from cells.

Transformers might be used for transferring energy between cells as proposed in [23], [24], [25], [26]. Fast equalization speed and non-neighbor charge transfers are main advantages of transformer-based active cell balancing techniques. However, transformers can be expensive, heavy, and might require large installation space as well as complex control.

Inductor-based balancing is energy efficient compared to capacitor-based approaches since there is practically no energy loss while charging. Compared to the transformer approach, inductors also require less installation space and are lighter. Thus, inductor-based architectures are good candidates for sustainable active cell balancing. Single-inductor balancing architectures use only one inductor in order to equalize the entire battery pack, see [27]. Cell pairs for balancing are selected by the control algorithm and corresponding switches associated with the cells are controlled in order to exchange charge. This method lacks modularity and does not allow concurrent charge transfers. By contrast, architectures using more than one inductor are a promising approach, see [28], [12]. The main advantage of this approach is its energy efficiency and modular nature of the architecture. Approaches range from simple architectures that can perform charge transfers only between neighboring cells, see [12], to complex architectures that enable non-neighbor transfers and cell bypassing [28]. Due to their

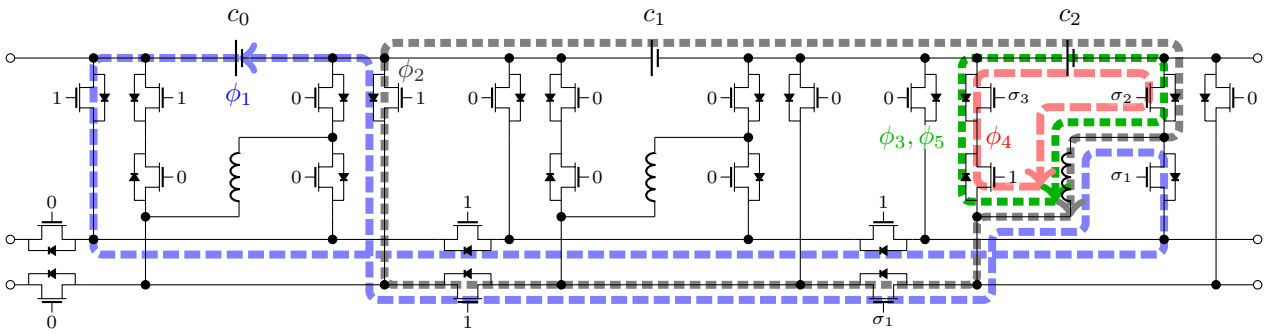


Fig. 11: Example of a charge transfer from c_0 to c_2 with the determined optimal circuit and switching rules. Charge is transfered from the first module to the third module. That means, the state for c_0 is SRC_RIGHT, for c_1 it is BRIDGE, and for c_2 it is DEST_RIGHT.

modularity, inductor-based approaches are furthermore well-suited for application in emerging smart cell architectures [2].

To the best of our knowledge, the paper at hand is the first work that provides a synthesis approach that optimizes balancing architectures at a structural level.

B. Logic and System Synthesis

Logic synthesis is a major field of research in electronic design automation. In digital logic synthesis [31], the goal is the design of components or systems from building blocks that are characterized by their digital behavior. Often, logic synthesis problems can be reduced to a Boolean representation such that SAT solvers [32] or Binary Decision Diagrams (BDDs) [33] can be applied to solve complex design problems. By contrast, analog circuit synthesis has to consider the electrical aspects of components that are not purely digital while improving specific performance metrics. Problems in this domain are more complex and often addressed by solutions that involve stochastic methods such as evolutionary computation [34], [35]. Other approaches such as [36] are capable of constructing analog circuits, following a set of predefined rules.

The considered active cell balancing problem is a mix of digital and analog logic synthesis. While the problem is formulated in a discrete domain like it is done for digital synthesis, the circuits contain energy storage elements, e.g, inductors, which are components from the analog domain. At the same time, active cell balancing not only requires the design of a circuit but also the definition of switching rules, demanding a synthesis approach to be capable of capturing these characteristics. Thus, to the best of our knowledge, existing approaches from logic synthesis cannot be directly applied. Instead, as presented, a tailored approach becomes necessary which is using optimization techniques from the system synthesis domain as discussed in the following.

In system synthesis, architectural exploration in combination with scheduling poses a similar problem as the discussed active cell balancing synthesis. In [37], [38], the problem is solved by applying a SAT solver such that each infeasible solution in terms of scheduling results is pruned. However, this pruning approach might result in poor scalability as discussed in [39]. Instead, it is beneficial to further analyze infeasible solutions to determine the reason for their infeasibility. This methodology has been successfully applied in [40] where a SAT solver suggests solutions and in case of infeasibility, the search space is further constrained by excluding the reason of the infeasibility. In fact, the proposed approach in the work at hand shares the main optimization principles with the synthesis approach in [40] and applies these to the domain of active cell balancing. In a wider sense, the proposed methodology also follows the SMT approaches [41] where SAT solvers are combined with complex background theories to extend the expressiveness of SAT solvers beyond CNF and linear expressions with binary variables.

VII. Concluding Remarks

This paper presents a synthesis approach for active cell balancing architectures, using design automation for the first time for the discussed problem. In contrast to the state of the art that relies on manual design, the proposed approach is capable of delivering solutions that are better in all considered objectives. The experimental results show that an optimal architecture is determined within less than two hours, exploring a large search space and multiple templates for circuits and signals. It remains an open research question how the methodology might be efficiently combined with generative methods to reduce the dependability on predefined templates.

References

[1] A. Otto, S. Rzepka, T. Mager, B. Michel, C. Lanciotti, T. Günther, and O. Kanoun, "Battery management network for fully electrical vehicles featuring smart systems at cell and pack level," in *Advanced Microsystems for Automotive Applications 2012*. Springer, 2012, pp. 3–14.

[2] S. Steinhorst, M. Lukasiewycz, S. Narayanaswamy, M. Kauer, and S. Chakraborty, "Smart cells for embedded battery management," in *Proceedings of the 2nd International Conference on Cyber-Physical Systems, Networks, and Applications*, 2014, pp. 59–64.

[3] M. Lukasiewycz, S. Steinhorst, and S. Narayanaswamy, "Verification of balancing architectures for modular batteries," in *Proceedings of the 12th International Conference on Hardware/Software Codesign and System Synthesis*, 2014, pp. 30:1–30:10.

[4] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into SAT," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp. 1–26, 2006.

[5] H. Samulowitz and R. Memisevic, "Learning to solve QBF," in *Proceedings of the 22nd National Conference on Artificial Intelligence*, vol. 7, 2007, pp. 255–260.

[6] L. De Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Formal Methods: Foundations and Applications*. Springer, 2009, pp. 23–36.

[7] L. W. Nagel and D. O. Pederson, *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California, 1973.

[8] D. Le Berre and A. Parrain, "The Sat4j library, release 2.2," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59–64, 2010.

[9] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[10] Cplex, IBM ILOG, "12.2 users manual," 2010.

[11] J. O'Madadhain, D. Fisher, S. White, and Y. Boey, "The JUNG (java universal network/graph) framework," *University of California, Irvine, California*, 2003.

[12] N. Kutkut, "A modular nondissipative current diverter for EV battery charge equalization," in *Proceeding of the Thirteenth Annual Applied Power Electronics Conference*, 1998, pp. 686–690.

[13] R. Sebastiani, "Lazy satisfiability modulo theories," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 3, pp. 141–224, 2007.

[14] J. Cao, N. Schofield, and A. Emadi, "Battery balancing methods: A comprehensive review," in *Proceeding of Vehicle Power and Propulsion Conference*, 2008, pp. 1–6.

[15] S. W. Moore and P. J. Schneider, "A review of cell equalization methods for lithium ion and lithium polymer battery systems," *SAE Publication 2001-01-0959*, 2001.

[16] M. Daowd, N. Omar, P. Van den Bossche, and J. Van Mierlo, "Passive and active battery balancing comparison based on MATLAB simulation," in *Proceedings of Vehicle Power and Propulsion Conference*, 2011, pp. 1–7.

[17] N. Kutkut and D. Divan, "Dynamic equalization techniques for series battery stacks," in *Proceedings of the 18th International Telecommunications Energy Conference*, 1996, pp. 514–521.

[18] T. Stuart and W. Zhu, "Fast equalization for large lithium ion batteries," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 7, pp. 27–31, July 2009.

[19] C. Pascual and P. Krein, "Switched capacitor system for automatic series battery equalization," in *Proceeding of the Twelfth Annual Applied Power Electronics Conference*, 1997, pp. 848–854.

[20] R. Fukui and H. Koizumi, "Double-tiered switched capacitor battery charge equalizer with chain structure," in *Proceeding of the 39th Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 6715–6720.

[21] A. Baughman and M. Ferdowsi, "Double-tiered switched-capacitor battery charge equalization technique," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 6, pp. 2277–2285, June 2008.

[22] C. Speltino, A. Stefanopoulou, and G. Fiengo, "Cell equalization in battery stacks through state of charge estimation polling," in *Proceedings of the American Control Conference*, 2010, pp. 5050–5055.

[23] J.-W. Shin, G.-S. Seo, C.-Y. Chun, and B.-H. Cho, "Selective flyback balancing circuit with improved balancing speed for series connected lithium-ion batteries," in *Proceedings of the International Power Electronics Conference*, 2010, pp. 1180–1184.

[24] A. Imtiaz and F. Khan, "Time shared flyback converter based regenerative cell balancing technique for series connected li-ion battery strings," *IEEE Transactions on Power Electronics*, vol. 28, no. 12, pp. 5960–5975, Dec 2013.

[25] M. Einhorn, W. Roessler, and J. Fleig, "Improved performance of serially connected li-ion batteries with active cell balancing in electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 6, pp. 2448–2457, July 2011.

[26] D. V. Cadar, D. M. Petreus, and T. M. Patarau, "An energy converter method for battery cell balancing," in *Proceedings of the 33rd International Spring Seminar on Electronics Technology*, 2010, pp. 290–293.

[27] S.-H. Park, T.-S. Kim, J.-S. Park, G.-W. Moon, and M.-J. Yoon, "A new buck-boost type battery equalizer," in *Proceedings of the Twenty-Fourth Annual Applied Power Electronics Conference*, 2009, pp. 1246–1250.

[28] M. Kauer, S. Naranayaswami, S. Steinhorst, M. Lukasiewycz, S. Chakraborty, and L. Hedrich, "Modular system-level architecture for concurrent cell balancing," in *Proceedings of the 50th Design Automation Conference*, 2013, pp. 155:1–155:10.

[29] S. Narayanaswamy, S. Steinhorst, M. Lukasiewycz, M. Kauer, and S. Chakraborty, "Optimal dimensioning of active cell balancing architectures," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2014, pp. 140:1–140:6.

[30] M. Kauer, S. Narayanaswamy, M. Lukasiewycz, S. Steinhorst, and S. Chakraborty, "Inductor optimization for active cell balancing using geometric programming," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2015, pp. 281–284.

[31] S. Hassoun and T. Sasao, *Logic synthesis and verification*. Springer Science & Business Media, 2012, vol. 654.

[32] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[33] R. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 293–318, 1992.

[34] T. McConaghy, P. Palmers, M. Steyaert, and G. G. Gielen, "Variation-aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 9, pp. 1281–1294, 2009.

[35] J. D. Lohn and S. P. Colombano, "Automated analog circuit synthesis using a linear representation," in *Evolvable Systems: From Biology to Hardware*. Springer, 1998, pp. 125–133.

[36] M. Meissner and L. Hedrich, "FEATS: framework for explorative analog topology synthesis," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 213–226, 2015.

[37] F. Reimann, M. Glaß, C. Haubelt, M. Eberl, and J. Teich, "Improving platform-based system synthesis by satisfiability modulo theories solving," in *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis*, 2010, pp. 135–144.

[38] F. Reimann, M. Lukasiewycz, M. Glass, C. Haubelt, and J. Teich, "Symbolic system synthesis in the presence of stringent real-time constraints," in *Proceedings of the 48th Design Automation Conference*, 2011, pp. 393–398.

[39] M. Lukasiewycz, S. Steinhorst, and S. Chakraborty, "Priority assignment for event-triggered systems using mathematical programming," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 982–987.

[40] M. Lukasiewycz and S. Chakraborty, "Concurrent architecture and schedule optimization of time-triggered automotive systems," in *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*, 2012, pp. 383–392.

[41] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to DPLL(T)," *Journal of ACM*, vol. 53, no. 6, pp. 937–977, 2006.

Martin Lukasiewycz Dr. Martin Lukasiewycz is a Principal Investigator in the TUM CREATE Centre for Electromobility in Singapore since 2011. Since 2014 he is Adjunct Assistant Professor at Nanyang Technological University in Singapore. Before, he worked at AUDI AG in Germany in the E/E architecture and FlexRay division, the chair Hardware/Software Co-Design at the University of Erlangen-Nuremberg in Germany, and the Institute for Real-Time Computer Systems at the Technical University of Munich.

Matthias Kauer Matthias Kauer is working as a research associate at TUM CREATE Centre for Electromobility in Singapore since 2012. He has studied Applied Mathematics at the Technical University of Munich from 2006 to 2011.

Sebastian Steinhorst Dr. Sebastian Steinhorst is a senior researcher with TUM CREATE Centre for Electromobility in Singapore. His research focuses on architectures and design automation for networked embedded systems with emphasis on electric vehicles. He is also leading a research team on embedded battery management. Before joining TUM CREATE in 2011, Sebastian was with the Electronic Design Methodology Group of Goethe-University Frankfurt/Main, Germany.