VDA Verband der Automobilindustrie

FAT Forschungsvereinigung Automobiltechnik

FAT-Schriftenreihe 364

Methodische Ansätze zur Auswahl von Bordnetzstrukturen mit erhöhten Zuverlässigkeitsanforderungen

Methodische Ansätze zur Auswahl von Bordnetzstrukturen mit erhöhten Zuverlässigkeitsanforderungen

Forschungsstellen

Fraunhofer-Institut für Integrierte Schaltungen IIS,
Institutsteil Entwicklung Adaptiver Systeme EAS
Dr. Peter Schneider

Universität Kassel
Fachbereich Elektrotechnik/Informatik
Fachgebiet Fahrzeugsysteme und Grundlagen der Elektrotechnik
Prof. Dr. rer. nat. Ludwig Brabetz

TU Dortmund
Arbeitsgebiet Bordsysteme
Fakultät für Elektrotechnik und Informationstechnik
Prof. Dr.-Ing. Stephan Frei

Das Forschungsprojekt wurde mit Mitteln der Forschungsvereinigung Automobiltechnik e.V. (FAT) gefördert.

Überblick

Im Rahmen der Automatisierung von Fahrfunktionen und der Elektrifizierung von Antriebssträngen inkl. elektrischer Lenkung, elektrische Bremsen sind die Anforderungen an die Betriebssicherheit von Fahrzeugen in den letzten Jahren enorm gestiegen und steigen stetig weiter. Dabei sollen einerseits die Fahrzeuge ein Maximum an Funktionaler Sicherheit bieten, andererseits soll die Verfügbarkeit des Fahrzeugs nicht eingeschränkt sein.
Die Bewertung und Auslegung der Fahrzeugsicherheit orientiert sich an den Anforderungen und der Vorgehensweise des internationalen Standard ISO 26262 - Funktionale Sicherheit [1]. Durch Anwendung der ISO 26262 werden den einzelnen Bordnetzkomponenten ASIL-Level (Automotive Integrity Safety Level) zugeordnet. Dies betrifft beispielweise die Anforderungen an Ausfallwahrscheinlichkeiten, Ausfallraten, usw. Durch die Verknüpfung einzelner Komponentenanforderungen wird schließlich ein gefordertes Sicherheitsniveau des Fahrzeugs bzw. des gesamten elektrischen Bordnetzes erreicht. Durch die Möglichkeit des Herunterbrechens der genannten Anforderungen von beispielsweise einer Komponente auf zwei oder drei Komponenten, sogenannte „Dekomposition“, ergibt sich eine Vielzahl von Kombinationsmöglichkeiten um die Gesamtheit der Anforderungen an das Bordnetz zu erfüllen. In der Praxis erfordert das händische „Durchspielen“ der Vielzahl an Kombinations-Möglichkeiten einen hohen zeitlichen Aufwand und viel Ingenieursleistung.
Die Komplexität wird noch dadurch gesteigert, dass durch die erhöhten Sicherheitsanforderungen an die automatisierten Fahrfunktionen, sowie die Einführung mehrerer Spannungslagen im Bordnetz (z.B. 48V/12V; 48V/12V/12V; HV/48V/12V) eine Vielzahl potentieller Lösungsmöglichkeiten für eine passende E/E Architektur existiert.
Rein rechnerisch ist eine Vielzahl an Topologien der E/E-Architekturen möglich. Sie unterscheiden sich in der Wahrscheinlichkeit von möglichen Ausfällen und damit im erreichbaren Sicherheitslevel. Bei der Sicherstellung der Energieversorgung sicherheitskritischer Systeme spielen neben statischen Spannungsniveaus, die zu erreichen sind, auch dynamische Effekte im Bordnetz eine große Rolle, wie zum Beispiel:

- wie viel Zeit wird benötigt, um eine Unterspannung im Bordnetz zu erkennen?
- wie viel Zeit wird benötigt, um auf eine redundante Energieversorgung umzuschalten?
- wie schnell schalten kritische Komponenten auf einen sicheren Betrieb um, wenn die Versorgungsspannung einbricht?

Solche Fragen sind mit einfachen analytischen Betrachtungen nicht zu beantworten. Im Rahmen des Projekts wurden daher Methoden untersucht, um eine Vorauswahl geeigneter anforderungsspezifischer Bordnetzstrukturen treffen zu können, ohne dass jede einzelne Kombinationsmöglichkeit vorab detailliert betrachtet werden muss. Wichtig für eine effiziente und hilfreiche Reduzierung der Architekturkandidaten sind dabei die folgenden drei Punkte:

- das gesamte Bordnetz und seine Komponenten werden systematisch simuliert
- Architekturkandidaten müssen automatisiert aufgestellt und bewertet werden
- typische Nutzungsprofile (Mission Profiles) sind in die Bewertung einzubeziehen

Dabei bildet das Vorprojekt „Simulationsgestützte Analyse und Bewertung der Fehlertoleranz von Kfz-Bordnetzen“ [2] die Grundlage, um eine Methodik zur Auswahl von Bordnetzarchitekturen für erhöhte Zuverlässigkeitsanforderungen zu schaffen. Die notwendigen Modelle für elektrische Komponenten (Verbraucher, elektrische Maschine, Spannungswandler, Batterie etc.) mit den entsprechenden Fehlermodi wurden hier geschaffen. Als Eingangsgrößen sollten Bewertungsszenarien aus der Betrachtung automatisierter Fahrfunktionen (z.B. Nothaltestrategie) verwendet werden.

Ausgehend von den Vorarbeiten im AK 30 wurde ein generischer Simulations-/Skript-basierter Ansatz für die Vorauswahl von Bordnetzstrukturen mit Einbeziehung der Sicherheits- und Zuverlässigkeitsanforderungen erarbeitet und an einfachen Beispielen demonstriert. Das erlaubt später in der Praxis gezielt nur die Bordnetze detailliert untersuchen zu müssen, die einen Mehrwert für die Erfüllung der Kriterien Funktionale Sicherheit und Zuverlässigkeit bieten. Zusätzlich wurde untersucht, inwieweit standardisierte Daten des physischen Bordnetzes (wie z.B. Kabelbaumdaten, wie Länge und Querschnitte sowie Architekturkonzepte) in den automatisierten Ablauf eingebunden werden können.

In **Arbeitspaket 1** wurden zunächst die Anforderungen an die Bewertung der Bordnetztopologien erarbeitet. Es ging darum, Bewertungskriterien bzw. Metriken aufzustellen sowie Bewertungsszenarien aufzustellen und ein konkretes Beispielsystem für die Bewertung auszuwählen. Daran waren neben den Forschungspartnern auch die Mitglieder des Arbeitskreises mit ihrem Anforderungen aus der täglichen Arbeit intensiv beteiligt. Im Ergebnis wurde das komplexe Szenario eines Nothalte-Vorgangs erarbeitet.

Arbeitspaket 2 beschäftigte sich in erster Linie mit der Generierung von Bordnetz-Topologien. Die Aufgabe, die sich aus der Motivation des Gesamtprojekts ergibt, bestand darin, anhand der freien Designparameter den Raum der möglichen Topologien mit einem hohen Automatisierungsgrad zu erzeugen. Dieser zunächst sehr große Lösungsraum wurde nun durch regelbasierte Einschränkungen reduziert. Diese Regeln umfassen logische, technische und wirtschaftliche Randbedingungen ebenso wie Annahmen und projektbedingte Vorgaben. Mit Hilfe eines generischen Beschreibungsformats wurden die ermittelten Topologien an die anderen Arbeitspakete weitergegeben. Weiterhin ging es darum, Nutzungsszenarien und Fehlerfällen für das resultierende Bordnetz zu beschreiben.

Im Rahmen von **Arbeitspaket 3** spielten vor allen Formate zur Notation der Topologien und deren Übersetzung in simulationsfähige Beschreibungen eine wichtige Rolle. Zunächst wurden vorhandene XML-basierte Formate für die Beschreibung von Bordnetztopologien, wie KBL und das neuere VEC-Format auf ihre Eignung für die hier betrachteten Zwecke untersucht. Im Ergebnis wurden diese recht mächtigen Formate nicht verwendet sondern ein einfaches JSON-basiertes Zwischenformat erstellt. Dieses erlaubt es, die generischen topologischen Informationen von Arbeitspaket 1 aus Matlab einzulesen und in verschiedene Hardware-Beschreibungssprachen zu exportieren. Exemplarisch wurden ein Modelica- und ein Spice-Export implementiert, mit denen die weitere Bewertung der verbliebenen Bordnetz-Topologien mit Hilfe von automatisierten Simulationen durchführbar ist.

In **Arbeitspaket 4** wurden schließlich die Verfahren für eine schnelle und automatisierte Bewertung der Bordnetztopologien erarbeitet. Grundlage dafür waren die simulationsfähigen Beschreibungen aus Arbeitspaket 2 zu den generierten Topologien. Es wurde eine Simulationsmethode erarbeitet und prototypisch implementiert, mit der die generierten Netzlisten effizient in großer Zahl analysiert werden können. Dadurch lassen sich in kurzer Zeit eine sehr große Vielzahl an Netztopologien simulieren und hinsichtlich der Reaktion auf angenommene Fehlerbilder bewerten. Durch die entwickelte Methode kann eine hohe Simulationsgeschwindigkeit erhalten werden und dennoch kapazitive oder induktive Effekte der angeschlossenen Lasten berücksichtigt werden.

Im Ergebnis des Projekts wurde die erarbeitete Bewertungsmethodik für exemplarische Bordnetz-Topologien demonstriert. Zu beispielhaft angenommenen Randbedingungen wurden zunächst eine Vielzahl an möglichen Topologien generiert, zu diesen automatisiert eine entsprechende Netzliste erzeugt und diese schließlich mit dem erarbeiteten Verfahren simuliert und bewertet. Die Analyse komplexerer Topologien und Bewertungsszenarien, wie sie im tatsächlichen Fahrbetrieb entstehen, lässt sich in künftigen Projekten darauf aufbauen.

[1] Road vehicles – Functional safety – Part 5: Product development at the hardware level. ISO 26262-5. Verfügbar: https://www.iso.org/standard/51360.html

[2] FAT 334: Simulationsgestützte Analyse und Bewertung der Fehlertoleranz von Kfz-Bordnetzen, Berlin: Forschungsvereinigung Automobiltechnik e.V., 2020

Teil 1

Anforderungen an die Bewertung

Dr. Roland Jancke

Fraunhofer Institut für Integrierte Schaltungen IIS
Institutsteil Entwicklung Adaptiver Systeme EAS

1 Anforderungen an die Bewertung

Im Rahmen des Projektes wurden Ansätze zur effizienten Bewertung von Bordnetztopologien untersucht. Ziel war es, mit Hilfe von Simulationen sehr schnell aus einer großen Anzahl von theoretisch möglichen Bordnetztopologien diejenigen herauszufiltern, für die sich eine detaillierte Analyse lohnt.
Grundsätzlich sind Fahrzeug-Bordnetze Teil der Sicherheitsarchitektur, da sie die Energie der vorhandenen Quellen zu den Verbrauchern bringen müssen. Sicherheitsrelevante Verbraucher müssen auch im Fehlerfall mit hoher Sicherheit noch mit Energie versorgt werden, um ein verlässliches und ungefährliches Verhalten des Fahrzeugs bis zum Stillstand und Abschalten zu ermöglichen. Somit ging es darum, Bordnetztopologien mit einer hohen Fehlertoleranz zu finden. Aber welche Größen sind es, die sich zur Bewertung und Optimierung von Bordnetz-Topologien eignen?
Im Projekt wird ein simulationsbasierter Ansatz verfolgt, um die Fehlertoleranz von Bordnetz zu untersuchen. Demzufolge müssen die Topologien in einer simulierbaren Form vorliegen und angenommene Fehler darin injiziert werden können. Hierzu gab es im Vorgänger-Projekt bereits umfangreiche Vorarbeiten auf denen das vorliegende Projekt aufbauen kann. Allerdings war im Vorgängerprojekt nicht die Frage untersucht worden, nach welchen Metriken aus einer Vielzahl von Fehlersimulationen am Ende die Robustheit oder Fehlertoleranz für ein vorliegendes Nutzungsszenario abzuleiten ist.
Das erste Arbeitspaket diente daher dazu, solche Bewertungskriterien für die Beurteilung verschiedener Bordnetz-Topologien aufzusetzen und Metriken aufzustellen, die einen quantitativen Vergleich ermöglichen (siehe Abschnitt 1.1). Außerdem wurde vor Beginn der Arbeiten im ersten Arbeitspaket eine Referenzarchitektur definiert. Diese sollte im weiteren Verlauf des Projekts für alle Arbeitspakete als beispielhafte Architektur eines Fahrzeug-Bordnetzes dienen, anhand derer die Untersuchungen der Projektpartner durchgeführt werden (siehe Abschnitt 1.2).
Das Arbeitspaket 1 war von Anfang an übergreifend angelegt, um die Grundlagen für alle weiteren Arbeitspakete zu schaffen. Daher waren in diesem Arbeitspaket alle drei Forschungspartner involviert. Darüber hinaus wurden auch die Mitglieder des AK 30 in diese grundlegenden Diskussionen einbezogen. Ziel war es, über die Beteiligung der industriellen Partner sinnvolle Bewertungskriterien und realistische Beispielssysteme aufzustellen.
Die Diskussionen zur Festlegung der Grundlagen für das Projekt wurden im Rahmen von Online-Workshops durchgeführt.

1.1 Bewertungsszenarien, Bewertungskriterien und Metriken

In einem Workshop wurden die Voraussetzungen geschaffen, um die Bordnetze bezüglich eines kritischen Nutzungsszenarios gegenüberstellen zu können.
Im Ergebnis wurde ein realitätsnahes Szenario für ein sicherheitskritisches Fahrmanöver definiert, welches in Abbildung 1 dargestellt ist. Dabei wird aus einer Fahrt mit konstanter Fahrzeuggeschwindigkeit und festgelegter stationärer Bordnetzlast ein Lenkmanöver (VDA doppelter Spurwechsel) durchgeführt. Dadurch wird das Bordnetz stark durch transiente Stromanstiege durch den Lenkaktuator, den Bremsaktuator und die Stabilitätskontrolle (ESP) belastet. Das Fahrzeug soll anschließend mit konstanter Geschwindigkeit weiterfahren.
Während des ersten Teils des Zyklus wird nun ein Fehler (z.B. Kurzschluss) implementiert.

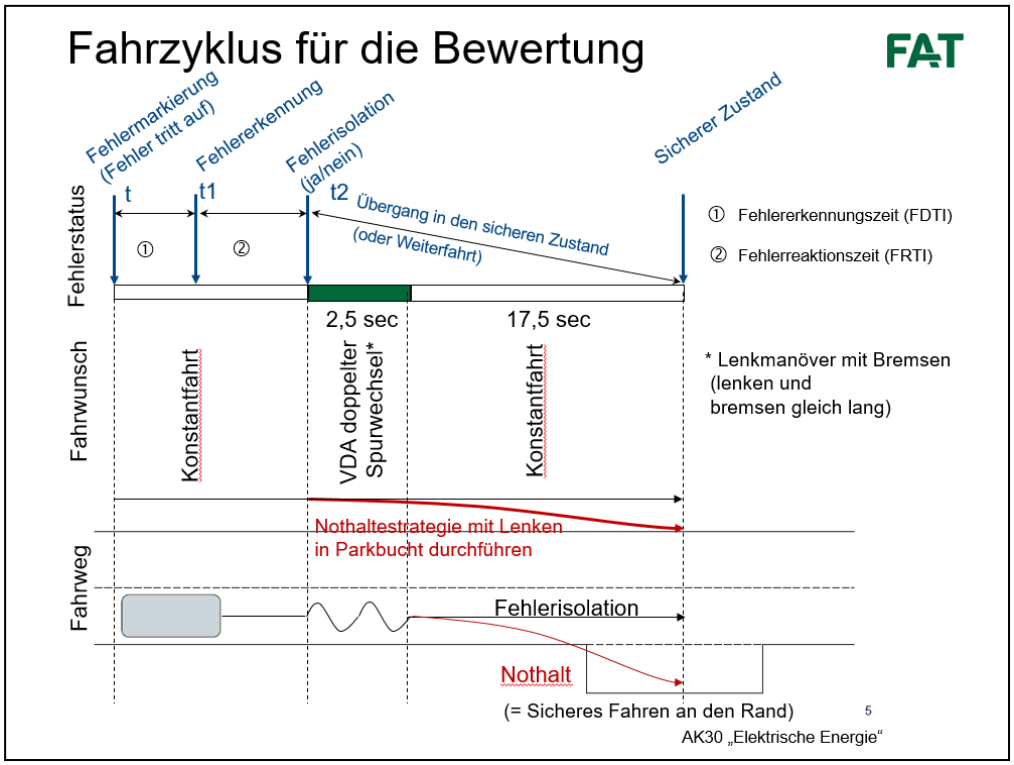


Abbildung 1: Bewertungsszenario Fahrweg mit doppeltem Spurwechsel und Fehlereinspeisung zum Zeitpunkt t zur Beurteilung von Bordnetzen bzgl. ihrer Fähigkeit zur Fehlerisolation und Stabilisierung des Bordnetzes

Bewertet wird dabei, in wie weit die jeweilige Bordnetz-Architektur in der Lage ist, den Fehler durch interne Maßnahmen (z.B. redundante Stromversorgung, Rerouting) zu isolieren und gleichzeitig die sicherheitsrelevanten Verbraucher stabil zu versorgen. Beobachtungsgrößen sind dabei Unterspannung, Überspannung, Bordnetzwelligkeit usw.
Ist das Bordnetz nicht rechtzeitig zur Fehlerisolation in der Lage wird eine Nothaltestrategie durchgeführt, um das Fahrzeug in einen sicheren Zustand zu versetzen. In diesem Fall ist der Spurwechsel bereits Teil der Nothaltestrategie, die insgesamt 20 Sekunden dauert.
Auch hier wird bewertet, ob das Bordnetz befähigt ist, geeignete Energiereserven verfügbar zu machen und stabil zu bleiben.

Mit den oben beschriebenen Grundüberlegungen wurden die Untersuchung der Erkennung und Behandlung von Fehlern stark intensiviert. Gleichzeitig wurden Untersuchungen zur Methodik der Einbindung von Daten-Formaten aus den Kabelbaumdatenbanken und Automatisierung von Simulationsabläufen begonnen. Dazu sind unterschiedliche Ansätze zur Simulation des dargestellten Fahrzyklus verglichen und diskutiert worden. Es kamen sowohl Varianten zur Sprache, bei denen das gesamte Bordnetz in einem Simulator modelliert und ausgeführt wird, als auch Ansätze zur Kopplung verschiedener Simulatoren über standardisierte Schnittstellen. Auch als Austausch-Format zur Beschreibung der Bordnetz-Topologie soll auf industrieweit akzeptierte Daten-Formate gesetzt werden. Für den Austausch von Stress-Signalen wurden neue Ansätze zur Standardisierung eines Mission-Profile-Formats betrachtet.
Im Laufe des Projekts wurden die Bewertungsmetriken schrittweise verfeinert.

1.2 Auswahl eines Beispielsystems für die Durchführung der Bewertung

Abbildung 2 zeigt die Festlegung einer „einhüllenden Bordnetztopologie“, welche als Ausgangsbasis für das Projekt diente. Aus diesem Konstrukt lassen sich die entsprechenden Bordnetztopologien, die später in der Bewertung gegenübergestellt wurden, durch Variation der physischen Verknüpfung (d.h. Leitungsführung und Anordnung der Bordnetzkomponenten) generieren. So konnten daraus beispielsweise Baumstrukturen, Ringstrukturen oder Backbone-Strukturen konzipiert werden und deren Vor- und Nachteile im Hinblick auf Zuverlässigkeit geprüft werden.
Eine zentrale Rolle im Bordnetz spielen neben den Energiespeichern und Quellen (Generator) und den (sicherheitsrelevanten) Verbrauchern dabei die Energieverteiler. Diese können als herkömmliche Stromverteiler (Power Distribution Units) bestückt mit Schmelzsicherungen (ggf. Relais) oder als elektronische Stromverteiler (ePDUs) ausgeprägt sein.
Ziel des Projekts war es, auch diese unterschiedlichen Arten der Absicherung und Verteilung in die Bewertung der Verfügbarkeit und der Fehlertoleranz der Bordnetze mit einzubeziehen.

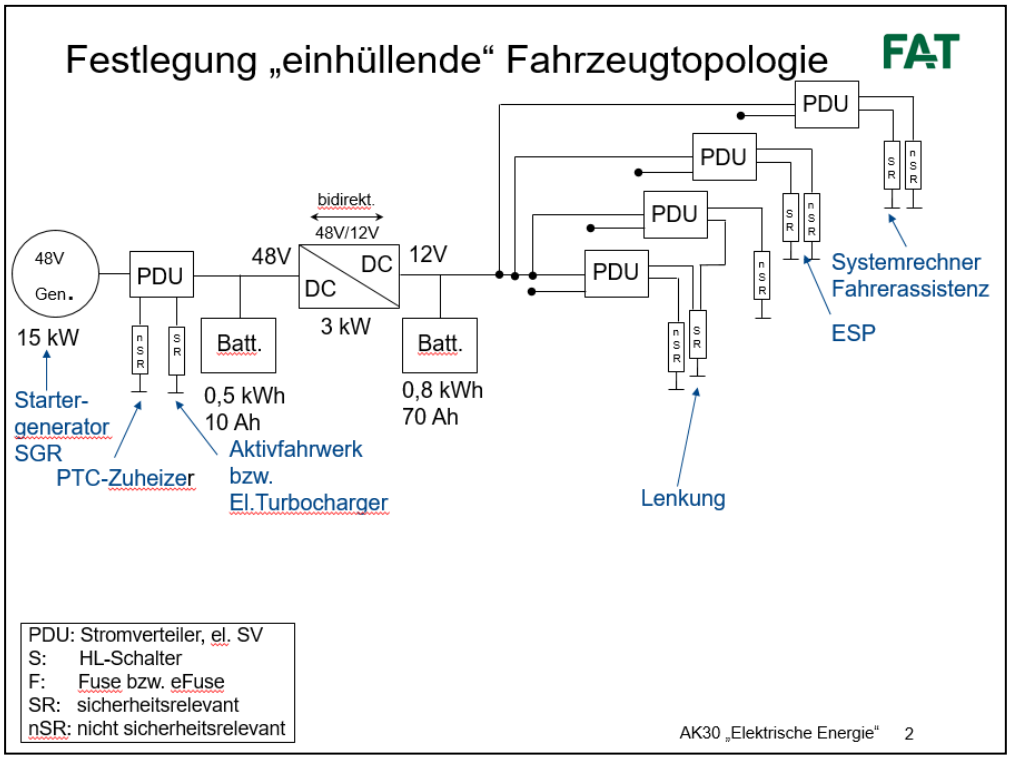


Abbildung 2: Festlegung 48V/12V Referenz-Bordnetzarchitektur. Aus dieser sind durch Variation der elektrischen Verbindungen alternative Topologien konfigurierbar, die im Projekt einer Bewertung zugeführt wurden.

Es wurde eine Architektur mit 4 Stromverteilern auf der Lastseite angenommen, an denen sicherheitsrelevante (S R) und nicht sicherheitsrelevante (n S R) Verbraucher hängen. Die Aufgabe besteht darin, geeignete Topologien zur Verknüpfung dieser Stromverteiler zu finden, so dass mit möglichst wenig redundanten Leitungen eine möglichst große Ausfallsicherheit des gesamten dargestellten Bordnetzes gegeben ist.

Teil 2

Generierung von Bordnetz-Topologien und Nutzungsszenarien

Prof. Dr. rer. nat. Ludwig Brabetz

Prof. Dr.-Ing. Mohamed Ayeb

M.Sc. Christian Koppe

Universität Kassel

Fachgebiet Fahrzeugsysteme und Grundlagen der Elektrotechnik

Inhaltsverzeichnis

AP 2-1 Automatisierte Erstellung von Bordnetz-Topologien

Das Energiebordnetzsystem im Kraftfahrzeug hat die Aufgabe die elektrische Energie bereitzustellen und zu verteilen, damit die elektrischen Verbraucher ihre zugewiesenen Funktionen voll erfüllen können. Da im Kontext der Fahrerassistenz und des autonomen Fahrens ein Teil dieser Funktionen immer sicherheitskritischer wird, ist das Bordnetz zu einer sicherheitskritischen Komponente geworden. Die Auslegung des Bordnetzes muss daher hohen Anforderungen an Zuverlässigkeit in einem kostensensitiven Marktumfeld genügen. Die gängige Herangehensweise basiert auf Erfahrungswerten unter der Berücksichtigung bekannter Randbedingungen, um mehr oder weniger evolutionär zur Auswahl einer Architektur für eine Fahrzeugplattform zu gelangen. Dabei besteht die Gefahr, dass der Lösungsraum zu eng gewählt wird und vielversprechende Bordnetzarchitekturen nicht betrachtet werden, die eventuell zur Lösung der Aufgabe besser geeignet sind. Dies ist vor allem der Fall, wenn wie aktuell der Fall ist, vollkommen neue Ansätze wie beispielsweise die Einführung zentraler Rechnereinheiten, zonaler Architekturen oder redundanter Versorgungswege. Dabei entsteht eine hohe Anzahl an Kombinationsmöglichkeiten, die nicht einzeln und bei jedem Fehlerfall detailliert simuliert werden können, um eine abschließende Bewertung für den gesamten Lösungsraum vorzunehmen. Daher wurde im Rahmen des Projekts eine Methodik entwickelt, die es erlaubt, alle denkbaren Topologien einer bestimmten Versorgungstruktur, definiert durch die Anzahl der Quellen, die Anzahl der Verteilerboxen und den maximalen Grad der Redundanz, automatisch zu generieren und vorab auf logischer Ebene unter Berücksichtigung aller Fehlerfälle zu bewerten. Dies erlaubt eine schnelle Bewertung der Versorgungssicherheit unter Berücksichtigung der relativen Kosten jeder generischen Topologie, so dass in einen zweiten Schritt eine detaillierte Bewertung einer kleinen Anzahl vielversprechender Lösungen zielgerichteter erfolgen kann.

Die Strukturen der Bordnetztopologien unterscheiden sich im Wesentlichen darin, wie die verfügbaren Energiequellen und -Speicher und die einzelnen Verteilerboxen, Power Distribution Units (PDU), miteinander verbunden sind. Eine Verteilerbox enthält i.d.R. Sicherungs-, Schalt- und Messinstrumente. Von der Verteilerbox aus können weitere Verteilerboxen mit elektrischer Energie versorgt werden. Die elektrischen Verbraucher sind ebenso an eine solche Verteilerbox angeschlossen.

So können die Verbindung der einzelnen PDUs beispielsweise linear, ringförmig, maschenartig oder baumstrukturmäßig verlegt sein. Kombinationen aus den genannten Strukturen können ebenso gebildet werden.

Die Vielzahl der möglichen Verbindungen wird dadurch erhöht, dass die PDUs redundant angeschlossen werden können. Das bedeutet, dass mehrere Verbindungen zu den einzelnen PDUs bestehen können. Die redundante Anschlussweise eignet sich zur Verringerung der Ausfallwahrscheinlichkeit der Versorgung der PDUs und trägt somit zur Sicherstellung der Versorgung der daran angeschlossenen Verbraucher und zur Erhöhung der funktionalen Sicherheit.

In diesem Arbeitspaket wurde ein Programm erstellt, welches die unterschiedlichen Topologien generisch erzeugt und auf logischer Ebene bewertet.

Prämissen

Aufgrund der großen Vielzahl von unterschiedlichen und damit möglichen Topologien wurden in dem Arbeitskreis einige Prämissen getroffen.

So wurde anhand der einhüllenden Bordnetztopologie (vgl. Abbildung 1), die von Herrn Dr. Thomas Lang (Bosch) vorgeschlagen wurde, die Anzahl der PDUs im Bordnetz mit vier festgelegt, an denen die sicherheitsrelevanten und nicht sicherheitsrelevanten Verbraucher angeschlossen sind.

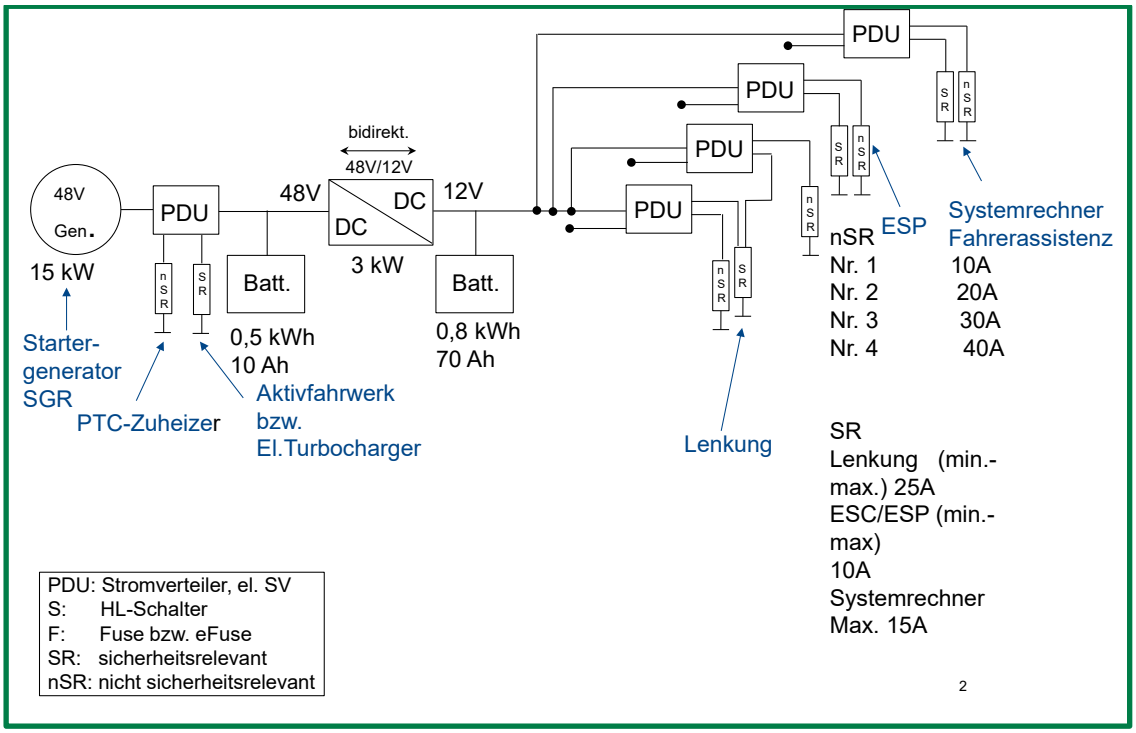


Abbildung 1: Einhüllende Bordnetztopologie.

Die Versorgungspunkte sind selber in sich abgesichert. Das bedeutet, dass die Quellen, die das Bordnetz mit elektrischer Energie versorgen, selber über Sicherungselemente und Überwachungselemente verfügen, sodass diese in einem möglichen Fehlerfall reagieren können. Es bedarf daher keine zusätzliche Unterstützung des Bordnetzsystems.

Es werden bei der Erstellung der möglichen Topologien nur maximal zwei Redundanzen betrachtet. Das bedeutet, dass maximal zwei Versorgungswege von einer PDU zu einer anderen PDU verlegt werden können.

Bei der Fehlerfallbetrachtung wird von einem einfachen Fehlerfall ausgegangen. Folgefehler oder das zeitgleiche Auftreten mehrerer Fehler werden nicht betrachtet.

Die Energie des Bordnetzes wird immer von zwei Quellen bereitgestellt. Als Quellen dienen ein Generator bzw, ein DC/DC-Wandler und eine Batterie.

Die Quellen sind außerdem direkt an den PDUs angeschlossen. Der Versorgungsweg und mögliche Fehlerarten zwischen Quellen und PDUs werden nicht betrachtet.

Beide Quellen können an einer einzelnen PDU angeschlossen sein oder an zwei unterschiedlichen PDUs.

Der Anschluss der Quellen in der Box sei durch eine galvanische Verteilung hergestellt. Eine übergeordnete intelligente Leistungsverteilung und Überwachung wird nicht vorausgesetzt.

In der PDU selber befinden sich intelligente Schalter, welche Leitungen hinzu- oder abschalten und somit den Stromfluss beeinflussen können. Die Schalter seien ansteuerbar.

Zusammenfassung der einzelnen Prämissen.

- Anzahl der PDUs auf vier festgelegt
- Versorgungspunkte sind in sich abgesichert
- nur max. zwei Redundanzen
- nur einfacher Fehlerfall
- Immer zwei Quellen (12V Batterie, Generator / DCDC)
- jede Quelle ist einer Box zugewiesen
- zwei Quellen können einer oder zwei Boxen zugewiesen werden
- ein galvanischer Verteiler als Anschlusspunkt Quelle
- Intelligenz in Box für Selektivität

Grundgerüst der Topologien

Das Grundgerüst für das Erstellen der Vielzahl an Topologien besteht aus vier PDUs, zwei Quellen sowie den schaltbaren und den nichtschaltbaren Verbindungen. Die Abbildung 2 zeigt das Grundgerüst der Topologien. Die blauen Kästchen stellen die PDUs dar. Die PDU sind stets fest platziert. Die erste PDU ist dabei mittig links angeordnet. Die Nummerierung der PDUs erfolgt im Uhrzeigersinn. So ist oben die PDU zwei, mittig rechts die PDU drei und unten die PDU vier angeordnet. Die zwei roten Kreise stellen die beiden Quellen dar. Der Abbildung ist zu entnehmen, dass die beiden Quellen entweder an der PDU eins, linkes Schema in der Abbildung 2, oder an der PDU eins und an der PDU 2, rechtes Schema in der Abbildung 2, angeschlossen sein können. Die blau gestrichelten Linien stellen die schaltbaren Verbindungen dar. Die rot gestrichelten Linien geben dagegen die Backbone-Verbindungen grafisch wieder. Mit Hilfe dieses Grundgerüst lassen sich die Vielzahl an unterschiedlichen Topologien grafisch wiedergeben, vergleichen und interpretieren.

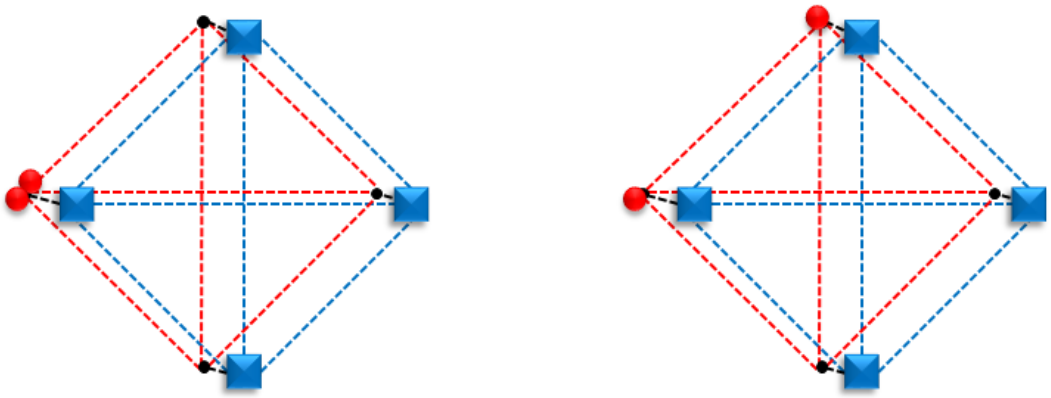


Abbildung 2: Grundgerüst der Topologien.

Schalter in den PDUs

Damit die Leitungen, welche die Verteilerboxen verbinden, in einem möglichen Fehlerfall getrennt werden können, werden intelligente Schalter in den PDUs verwendet. Die intelligenten Schalter haben folgende Eigenschaften:

- Die Schalter sind ansteuerbar
- Die Schalter können Leitungen galvanisch trennen
- Die Schalter können Überwachungs- bzw. Messaufgaben übernehmen

In der Abbildung 3 sind eine Beispieltopologie mit dem Schalterprinzip aus der PDU drei abgebildet. Der rote Schalter in der PDU kann die PDU 3 von der roten Backbone-Verbindung trennen. Der blaue Schalter kann die blaue, schaltbare Verbindung zu der PDU 2 schalten. Beide Schalter sind in der PDU über eine Potentialschiene miteinander verbunden. Von dieser Potentialschiene werden die Verbraucher, die an der PDU angeschlossen sind, mit Energie versorgt. Die Verbraucher selber sind über Sicherungselemente bzw. Schalter mit der Potentialschiene verbunden.

Die Abbildung 4 zeigt die maximale Ausstattung einer PDU. Sie verfügt über drei blaue Schalter für die drei schaltbaren Verbindungen zu den benachbarten 3 PDUs, einen roten Schalter zum Backbone, eine Verbindung zur einer Versorgungsquelle (rot gestrichelt), eine Potentialschiene und abgesicherte Abgänge zu den sicherheitsrelevanten und nicht sicherheitsrelevanten Verbrauchern (schwarz).

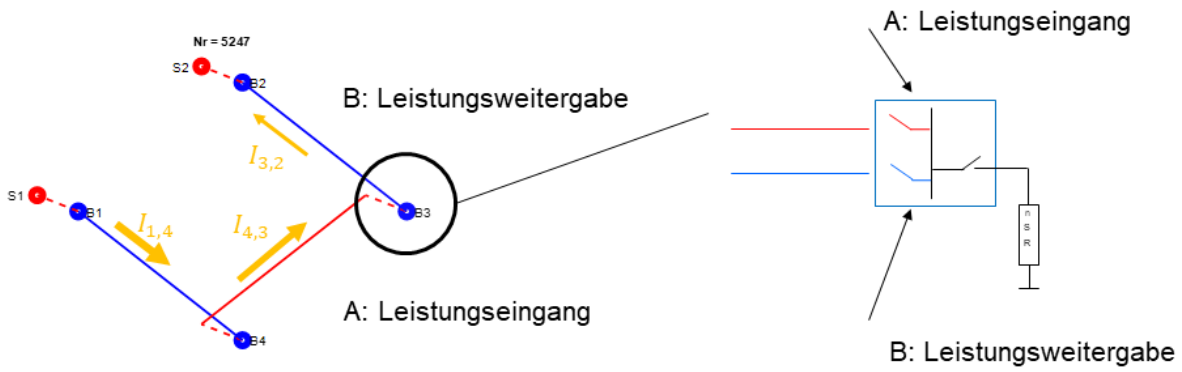


Abbildung 3 Schalter in der PDU.

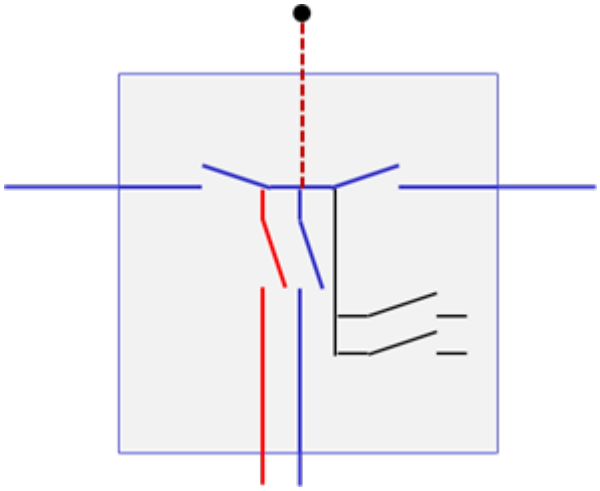


Abbildung 4: maximale Anzahl an Schaltern in der PDU.

Automatische Generierung der Topologien

Die automatische Generierung der Topologien geschieht über ein Matlab-Skript. In diesem Skript werden die Topologien generisch erzeugt. Die Anzahl der Varianten ergibt sich durch das Kombinieren aller möglichen Verbindungen zwischen den PDUs.

Die Möglichen Verbindungen bestehen aus den Backbone- und aus den schaltbaren Verbindungen.

Für die Quellen, welche die Versorgung der PDUs bei den jeweiligen Topologien sicherstellen, sind zwei Kombinationen vorgesehen. Beide Quellen sind der gleichen PDU zugeordnet oder die beiden Quellen sind an zwei unterschiedlichen PDUs verteilt angeschlossen. In dem Skript geschieht der Anschluss der beiden Quellen entweder zusammen an der PDU eins oder getrennt an der PDU eins und an der PDU zwei. Die PDUs drei und vier brauchen dabei nicht berücksichtigt zu werden, da die sich daraus ergebenden Topologien gleichwertig sind mit den zuvor erwähnten zwei Varianten.

Beim automatischen Durchspielen aller Kombinationen entstehen auch Topologien, bei denen, auch im fehlerfreien Fall, nicht alle PDUs versorgt werden. Diese Topologien werden als nicht sinnvoll betrachtet, vorab sortiert und nicht mehr bei der Bewertung berücksichtigt.

Die Abbildung 5 zeigt das Prinzip der Topologien bei unterschiedlichem Anschluss der beiden Quellen an die PDUs. Bei beiden Anschlussprinzipien existieren eine Anzahl von 4096 möglichen Kombinationen.

Für den Fall, dass beide Quellen an der PDU eins angeschlossen sind, verbleiben noch 3834 Kombinationen, welche zu untersuchen sind.

Für den Fall, dass beide Quellen an unterschiedlichen PDUs angeschlossen sind, verbleiben noch 3960 Kombinationen, welche zu untersuchen sind.

Insgesamt bleiben so noch 7794 unterschiedliche Topologien übrig, welche über das Matlab-Skript erzeugt wurden.

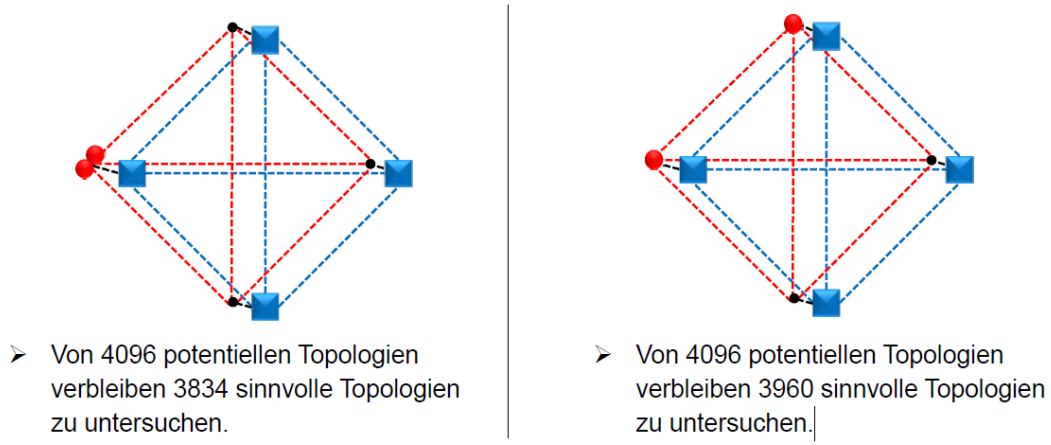


Abbildung 5: Anzahl der möglichen Topologien, bei unterschiedlichen Anschluss der beiden Quellen.

Fehlerarten

Im Bordnetz können eine Vielzahl von Fehlern auftreten.

So kann eine Quelle defekt sein. Eine defekte Quelle kann das Bordnetz nicht weiter mit elektrischer Energie versorgen.

Eine schaltbare Verbindung kann gebrochen sein. Über die Leitung kann die elektrische Energie nicht weitergegeben werden. Das elektrische Potential wäre auf der Leitung quellseitig vorhanden. Jedoch wäre das Potential nach der Bruchstelle (senkenseitig) nicht mehr vorhanden.

Eine schaltbare Verbindung kann einen Kurzschluss gegen Masse haben. Eine Auswirkung bei diesem Fehler ist, dass ein Kurzschlussstrom fließt und dass kein Spannungspotential an der Verbraucherseite anliegen wird.

Eine Backbone -Verbindung kann gebrochen sein. Über die Leitung kann die elektrische Energie nicht weitergegeben werden. Das Potential wäre auf der Leitung quellseitig vorhanden. Jedoch wäre Das Potential nach der Bruchstelle (senkenseitig) nicht mehr vorhanden.

Eine Backbone-Verbindung kann einen Kurzschluss gegen Masse haben. Eine Auswirkung bei diesem Fehler ist, dass ein Kurzschlussstrom fließt und dass kein Spannungspotential an der Verbraucherseite anliegen wird.

Des Weiteren kann ein Fehler auch in der PDU selber auftreten. Dabei sei der Fehler so definiert, dass die Potentialschiene gegen Masse kurzgeschlossen sei. Die PDU kann somit die elektrische Energie an nachfolgende PDUs nicht mehr weitergeben. Die Verbraucher, die an der PDU angeschlossen sind, können nicht weiter mit elektrische Energie versorgt werden.

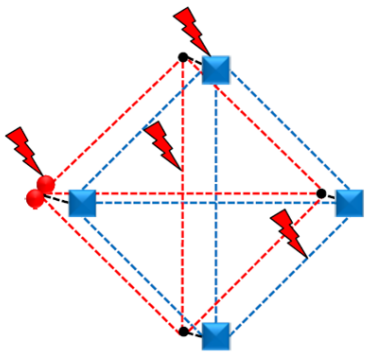


Abbildung 6: Fehlerorte in den Bordnetz.

Die Abbildung 6 stellt die Lokalitäten der einzelnen Fehlerorte dar. Die Fehler sind mit einem roten Blitz gekennzeichnet. In der Abbildung ist zu erkennen, dass die Fehler an den Quellen (rote Kreise), an einer schaltbaren Verbindung (blau gestrichelter Linie), an einer Backbone-Verbindung (rot gestrichelter Linie) sowie in der PDU (blaues Kästchen) auftreten können.

<u>Eine Übersicht über die möglichen Fehlerarten:</u>

- eine Quelle ist defekt
- Leitungsbruch bei einer schaltbaren Verbindung zwischen den PDUs
- eine schaltbare Verbindung zwischen den PDUs ist kurzgeschlossen (gegen Masse)
- eine Backbone-Verbindung zwischen den PDUs ist offen (gebrochen)

- eine feste Backbone-Verbindung zwischen den PDUs ist kurzgeschlossen (gegen Masse)
- eine Box ist defekt: Potentialverteilungsschiene ist kurzgeschlossen (gegen Masse)

Auswertung der Versorgungssicherheit und den Kosten

Damit die unterschiedlichen Topologien miteinander verglichen werden können, findet in dem Matlab Skript eine Auswertung statt. Die Auswertung erfolgt zum einen über eine Fehleranalyse und zum anderen über frei definierte Kosten. Des Weiteren wird die Auswertung mit Hilfe von Fehlerwahrscheinlichkeiten präzisiert.

Die Fehlerbewertung erfolgt über die Anzahl der Möglichkeiten, dass ein bestimmter Fehler (s. Fehlerarten) in einer Topologie auftritt. Die Anzahl, der durch die Fehlerart nicht mehr mit elektrischer Energie versorgten PDUs, wird ausgewertet. Je mehr PDUs durch einen Fehler nicht mehr mit Energie versorgt werden können, desto schlechter fällt die Bewertung aus. In der Abbildung 7 sind anhand einer Beispieltopologie 4 mögliche Fehler und deren Auswirkung beschrieben.

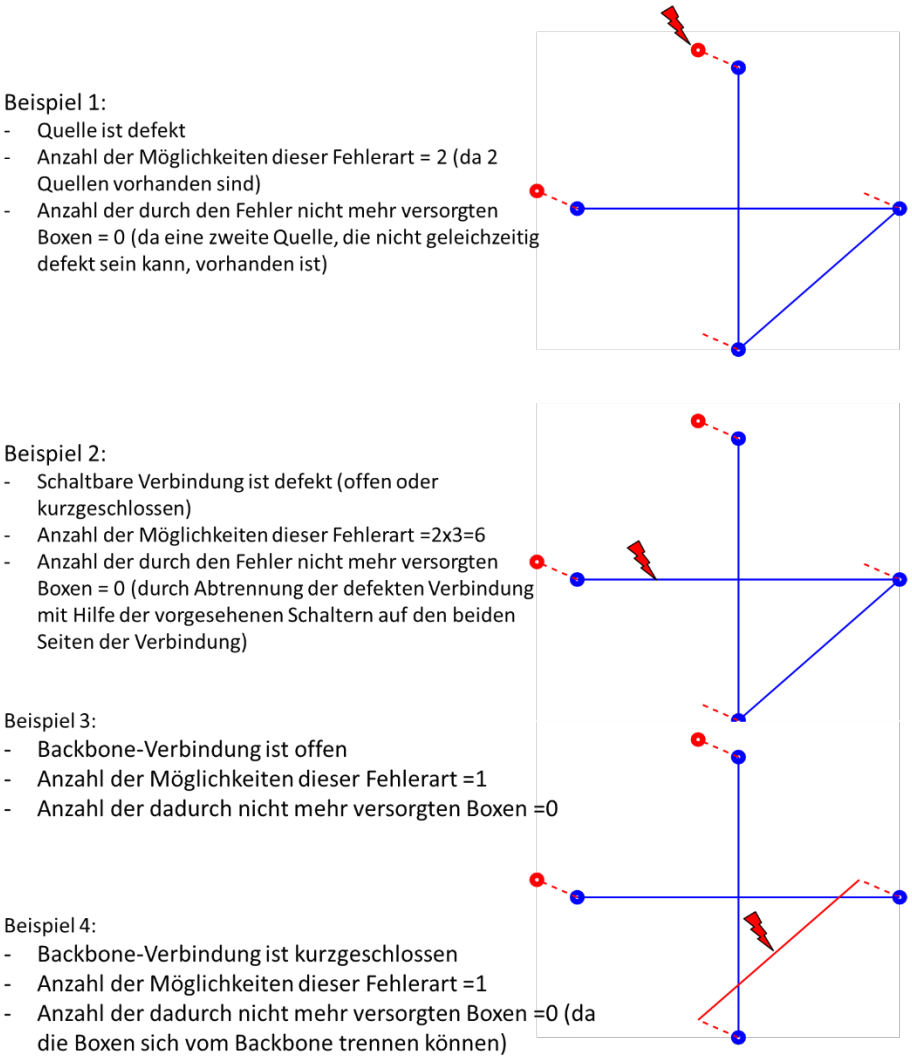


Abbildung 7: Beispiel für zwei Auftretende Fehler anhand einer Beispieltopologie.

Die Topologie in der Abbildung 7 besteht aus zwei schaltbaren und einer Backbone Verbindung. Die Quellen sind an zwei unterschiedlichen PDUs angeschlossen. Die Versorgung aller PDUs mit elektrischer Energie ist im fehlerfreien Fall gewährleistet. Durch die vorhandene Redundanz ist die Versorgung der PDUs bei jedem der hier betrachteten einfachen Fehlern sichergestellt.

In dem Beispiel 3 in der Abbildung 7 ist beschrieben, dass die Backbone-Verbindung offen sei. Das bedeutet, dass die Leitung gebrochen ist und somit kein Stromfluss über diese Verbindung stattfindet. Die Anzahl der durch den nichtmehr versorgten PDUs ist null. Jede PDU wird trotz des Fehlers weiterhin mit elektrischer Energie versorgt. Die PDU eins wird über die Quelle eins, welche direkt an der PDU angeschlossen ist mit Energie versorgt. Die PDU eins gibt die elektrische Energie an die PDU drei weiter. Die PDU zwei wird von der Quelle zwei mit elektrischer Energie durch die direkte Anbindung versorgt. Die PDU zwei versorgt die PDU vier durch die schaltbare Verbindung mit Energie.

In dem Beispiel 4 in der Abbildung ist angenommen, dass die Backbone-Verbindung einen Kurzschluss zur Masse habe. Das bedeutet, dass kein Stromfluss über diese Verbindung stattfinden kann. Auch in diesem Fehlerfall bleiben wie Beispiel drei die vier PDUs versorgt dank der vorgesehenen Möglichkeit der Trennung der PDUs von der Backbone-Verbindung im Kurzschlussfall.

Bei der Abschätzung der Kosten der jeweiligen Topologie werden nur die relativen Kosten in einer einfachen Form betrachtet. Die Kosten für die PDUs und die beiden Quellen werden nicht berücksichtigt, da diese Kosten bei allen 7794 Topologien gleichbleibend sind.

Daher geschieht eine Kostenabschätzung, welche in dem Skript variabel gestaltet werden kann für folgende Bauteile:

- Kosten für die Backbone-Verbindung
- Kosten für die schaltbare Verbindung
- Kosten für die benötigten Schalter in der PDU

Bei den Kosten für eine schaltbare Verbindung werden zwei Möglichkeiten unterschieden. Werden zwei Schalter an den beiden Enden einer schaltbaren Verbindung benötigt so werden die Kosten für zwei Schalter berücksichtigt. Wird nur ein Schalter benötigt, so werden nur die Kosten für einen Schalter berücksichtigt. Für jede Verbindung wird algorithmisch festgesellt, ob die Verbindung in einer Mache liegt und zwei Schalter benötigt oder nicht.

Relative Kosten für:
- eine Backbone-Verbindung = 1
- eine schaltbare Verbindung
  - einfach = 6
  - doppelt = 11
- **Schalter zwischen Box und Backbone = 5**

Fehlerwahrscheinlichkeiten für einzelne Ereignisse:
- Leitungsfehler = 1e-6
- Quellenfehler = 1e-8
- Backbone Verbindung gebrochen = 1e-9
- Backbone Verbindung kurzgeschlossen = 1e-9
- Box-Fehler = 1e-9

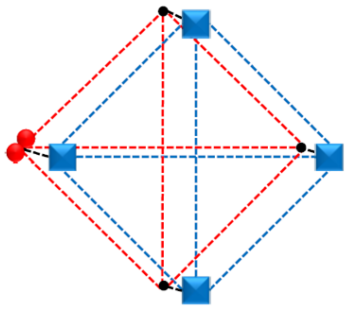


Abbildung 8: Beispielhafte Kosteneingabe für die relativen Kosten.

In der Abbildung 8 sind die angenommen relativen Kosten zu sehen. Diese Parameter können verändert werden. Des Weiteren sind die Fehlerwahrscheinlichkeiten abgebildet. Auch diese Wahrscheinlichkeiten sind veränderbar. So können die Ausfallwahrscheinlichkeiten für Leitungsverbindungen, Quellen und PDUs aus anderen Informationsquellen hier eingegeben werden.

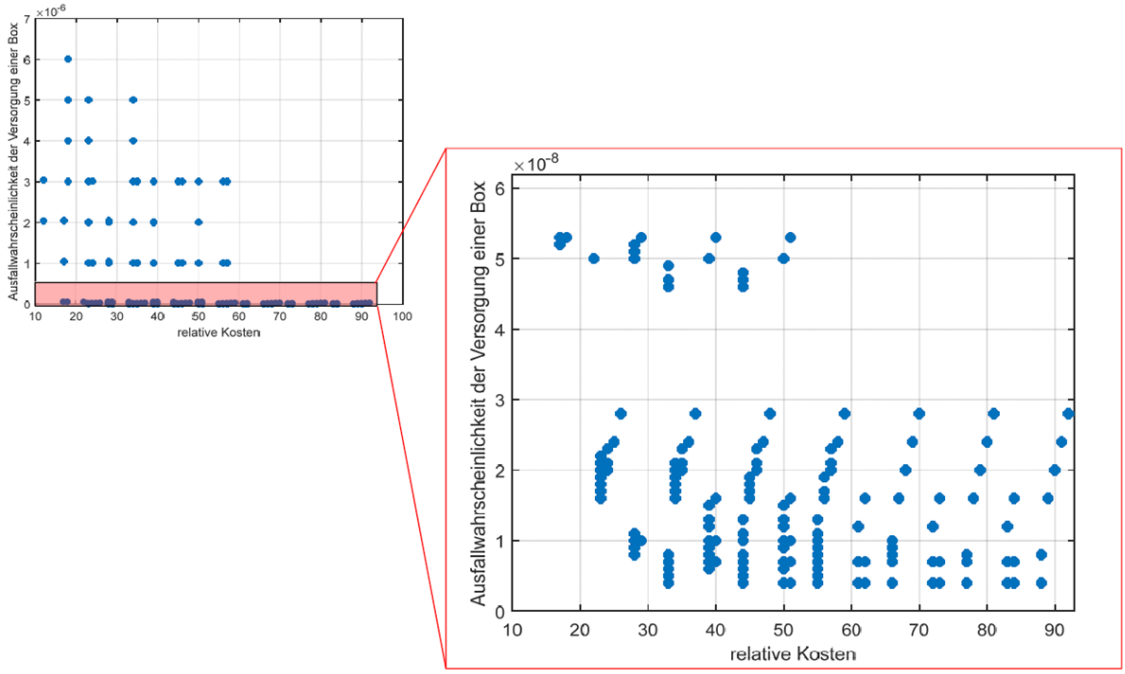


Abbildung 9: Grafische Ausgabe über die relativen Kosten und der Ausfallwahrscheinlichkeiten der PDUs.

In der Abbildung 9 ist eine Grafische Ausgabe der Bewertung der 7794 Topologien, blaue Kreise, zu sehen. Auf der x- Achse sind die relativen Kosten aufgetragen. Auf der y-Achse sind die Ausfallwahrscheinlichkeiten der PDUs aufgetragen. Zu erkennen ist eine Pareto-Front. So dass ein Kompromiss durch Gewichtung zwischen Kosten und Ausfallwahrscheinlichkeit je nach Bedarf definiert werden kann und entsprechend die dazugehörigen Toplogielösungen abgelesen werden können.

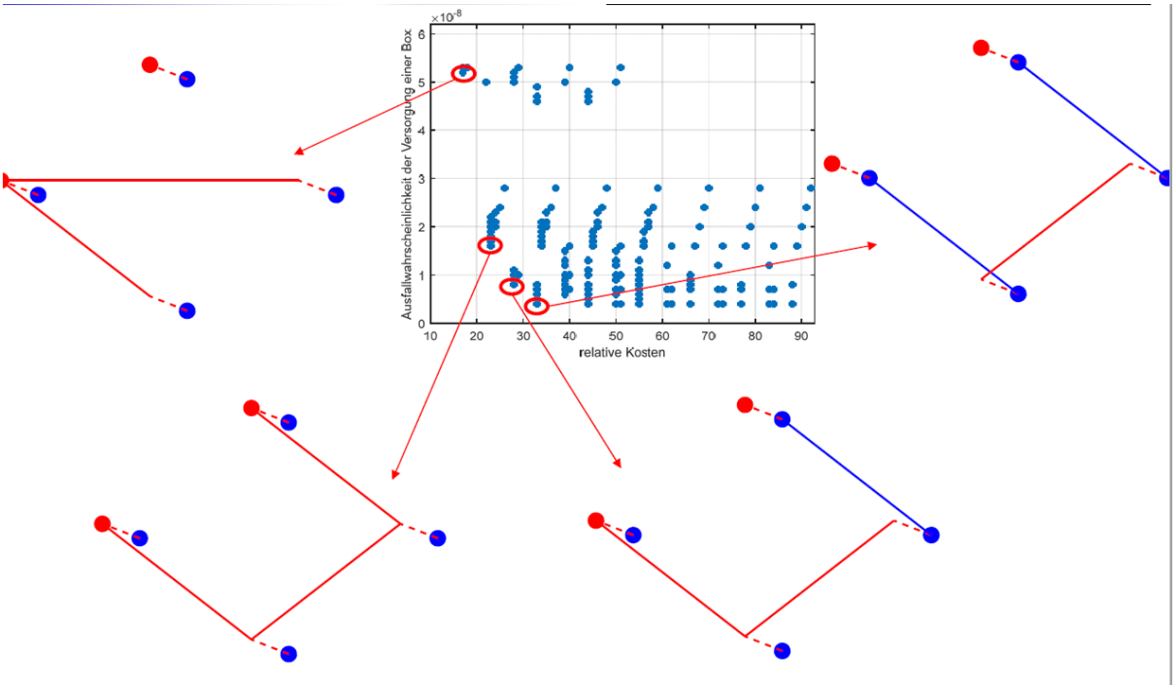


Abbildung 10: Auswahl der Topologien anhand der Pareto-Front.

Dadurch, dass sowohl die relativen Kosten, als auch die Parameter für die Ausfallwahrscheinlichkeiten verändert werden können, ergeben sich neue grafische Ausgaben und somit neue Pareto-Fronten. Die Abbildung 10 zeigt beispielhaft eine Auswahl von unterschiedlichen Topologien anhand der Pareto-Front.

Die Grafische Ausgabe erleichtert somit die Auswahl eine Gewichtung zwischen den relativen Kosten und der Ausfallwahrscheinlichkeit der PDUs.

Datenstruktur

In diesem Abschnitt wird die Datenstruktur, welche in dem Matlab-Skript verwendet wurde, erläutert.

- Unter Matlab entstanden, da günstig für Algorithmen-Entwicklung zur Topologie-Bewertung.
- Einsatz des Datentyps „structure array“, da zum einen strukturiert anlegbar und zum anderen flexibel bezüglich Erweiterbarkeit durch Hinzufügen von zusätzlichen Feldern „fields“ mit beliebigen Datentypen wie Zahlen, Zeichen, Zeichenketten etc...

```
>> Top_sB_nF(3503)
ans =
  struct with fields:
              arch: [1×1 struct]
                 G: 1
       BoK_Anz_Fehler: [1 1 1 1]
     BoK_Anz_VersBoxen: [2 3 3 3]
        BoK_VersBoxen: [4×4 double]
       BBK_Anz_Fehler: [2 1]
     BBK_Anz_VersBoxen: [4 4]
        BBK_VersBoxen: [4×2 double]
       BBO_Anz_Fehler: [1 1 1]
     BBO_Anz_VersBoxen: [4 4 4]
        BBO_VersBoxen: [4×3 double]
       LF_Anz_Fehler: [1 1 1 1]
     LF_Anz_VersBoxen: [3 4 4 4]
        LF_VersBoxen: [4×4 double]
       QF_Anz_Fehler: [1 1]
     QF_Anz_VersBoxen: [4 4]
        QF_VersBoxen: [4×2 double]
            Anz_esL: 1
            Anz_dsL: 3
             Anz_BB: 3
             Anz_sB: 3
             Kosten: 57
             Fehler: 1.0050e-06
    Fehler_per_Box: [1.0000e-09 1.0020e-06 1.0000e-09 1.0000e-09]
```

Abbildung 11: structure array unter Matlab.

In der Abbildung 11 ist das structure, welches in Matlab aufgebaut wurde, dargestellt. Das Array hat den Vorteil, dass strukturiert angelegt werden und bei Bedarf erweitert werden kann.

In diesem Array befinden sich u.a. die Informationen über die Art der Verbindungen, die Fehlerwahrscheinlichkeiten und die Möglichen Fehler.

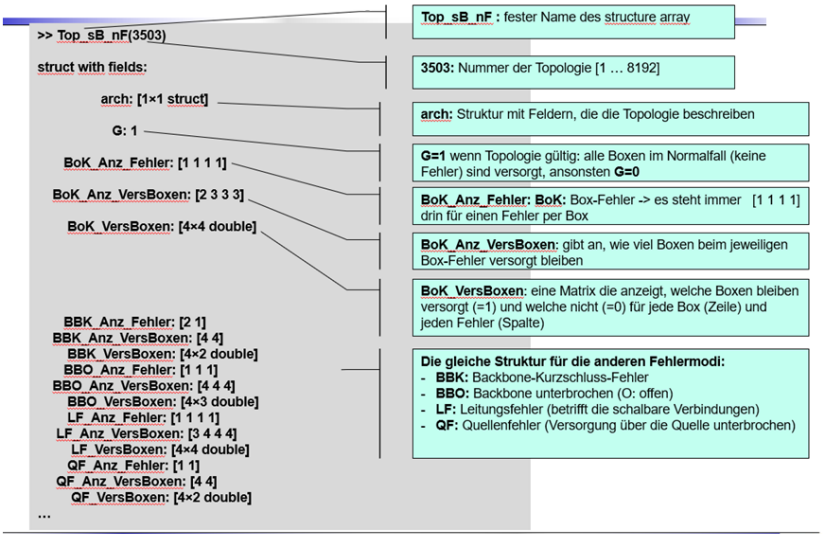


Abbildung 12: Übersicht über die Informationen der PDU.

Die Informationen bzgl. der PDUs in dem Matlab-Skript ist in der Abbildung 12 dargestellt. Dort finden sich u.a. die Informationen über die Anzahl der PDUs, welche im Fehlerfall noch versorgt werden, wieder.

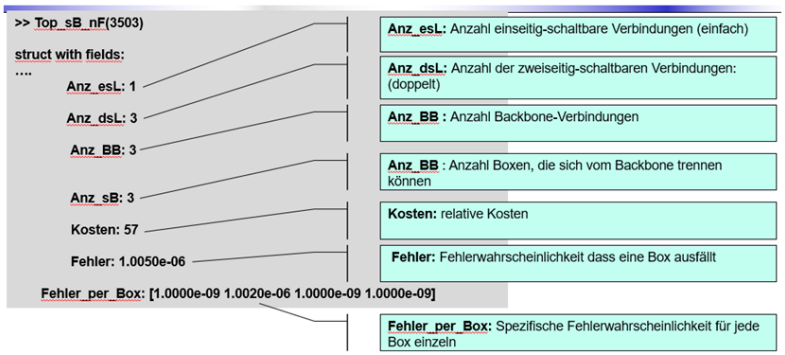


Abbildung 13 Informationen bzgl. der Leitungen, der Kosten und der Fehlerwahrschenlichkeit.

In der Abbildung 13 finden sich die Informationen über die Anzahl der schaltbaren-Verbindungen sowie der Backbone-Verbindungen der jeweiligen Topologie wieder. Des Weiteren bildet die Abbildung 13 die Informationen über die relativen Kosten und der Ausfallwahrscheinlichkeit wieder. Diese Informationen befinden sich in einem struct.

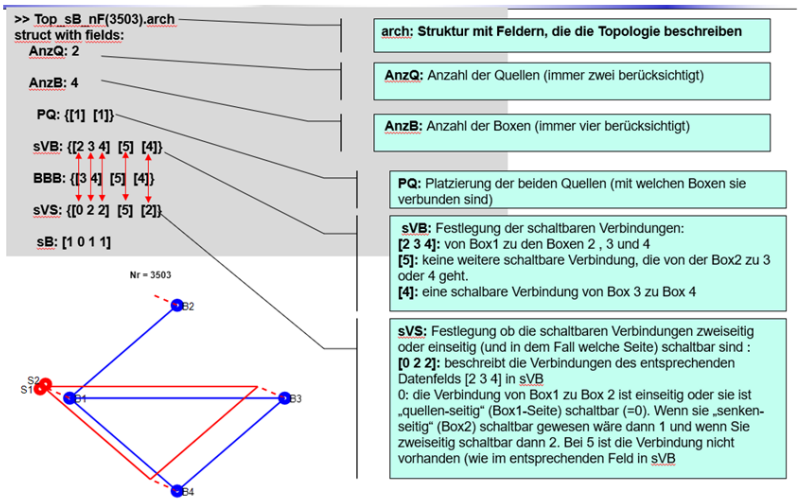


Abbildung 14: Informationen über die Leitungsführung einer Topologie.

Aus der Abbildung 14 geht die Informationen über die Leitungsführung einer Topologie hervor. Es ist anhand eines Beispiels erklärt, wie die Matrizen die Leitungsverbindungen zwischen den jeweiligen PDUs wiederspiegeln. In dieser Abbildung sind die Informationen über die schaltbaren Verbindungen aufgeführt. In diesem Teil der Datenstruktur wird außerdem beschrieben, ob die schaltbare Verbindung einseitig oder zweiseitig geschaltet wird. Die Information spielt bei der Berechnung der relativen Kosten der Topologien eine Rolle.

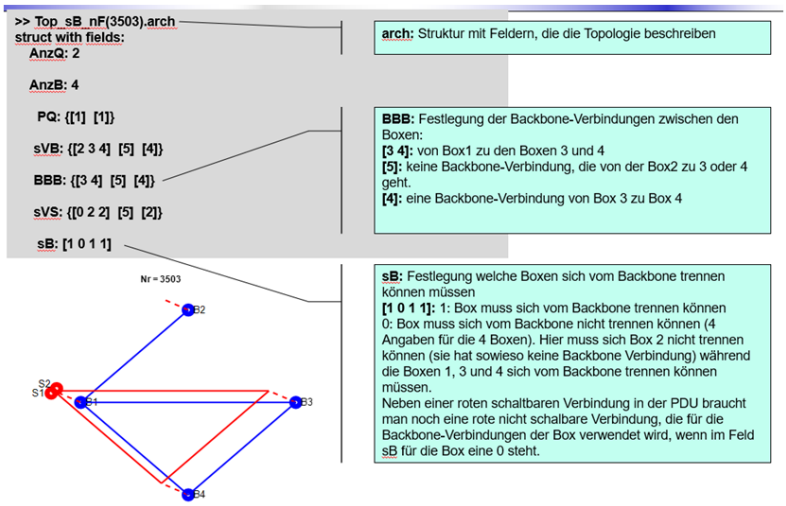


Abbildung 15 Informationen über die Backboneleitung einer Topologie.

In der Abbildung 15 geht die Informationen über die Backbone-Leitungsführung einer Topologie hervor. Es ist anhand eines Beispiels erklärt, wie die Matrixen in dem Matlab-Skript die Backbone-Verbindungen zwischen den jeweiligen PDUs wiederspiegeln. Des Weiteren sind die Punkte beschrieben, wo die Backbone-Verbindung anliegt und ob die PDU sich von diesem Punkt trennen kann.

AP 2-2 Beschreibung von Nutzungs- und Ausfallszenarien

Anhand der einhüllenden Bordnetztopologie wird gezeigt, wie ein Topologie-Beispiel im Fahrzeug positioniert werden kann.

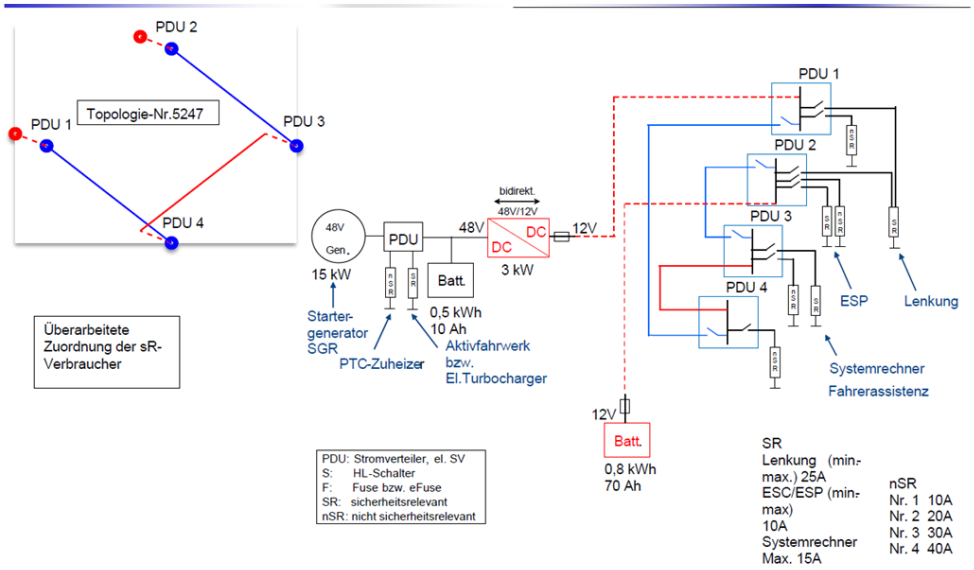


Abbildung 16: Topologie Nr.5247 angewendet auf die einhüllende Bordnetztopologie.

Die Abbildung 16 zeigt die Topologie Nr.5247 auf die einhüllende Bordnetztopologie angewendet. Zu erkennen sind die Anschlusspunkte der beiden Quellen an die PDU eins bzw. an die PDU 2. Die schaltbaren Verbindungen sowie die Backbone-Verbindung, sind sowohl in der einhüllenden Topologie als auch in dem Grundgerüst, welches in Matlab erzeugt wird, wiedergegeben.

Die Abbildung 17 zeigt hingegen, die gleiche Topologie auf ein Fahrzeug projiziert. In der Abbildung ist das Fahrzeug in verschiedene Rasterelemente unterteilt. Die einzelnen PDUs sind im Fahrzeug platziert und nach der Topologie Nr. 5247 miteinander verknüpft. Die Einteilung des Fahrzeuges in ein Raster ermöglicht so eine Abschätzung über die Länge der jeweiligen Verbindungen. Auch hierbei existiert wieder eine Vielzahl von unterschiedlichen Möglichkeiten die Verbindungen im Fahrzeug auszulegen.

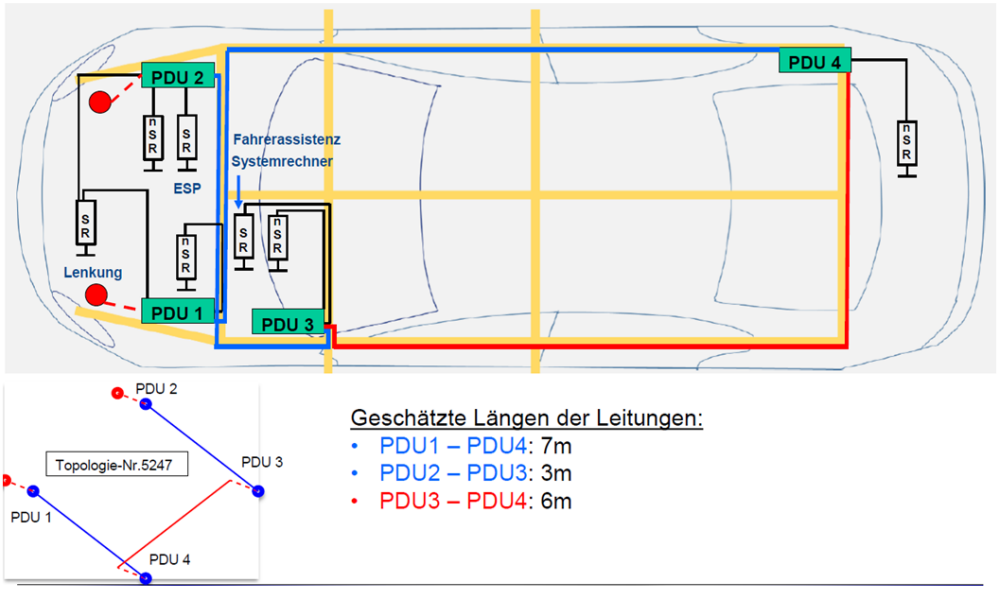
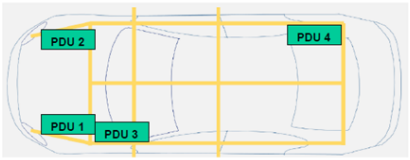


Abbildung 17: Topologie Nr. 5247 im Fahrzeug platziert.

Die Länge der Verbindungen trägt einen Teil bei der Berechnung des Kabelquerschnitts bei. Die Information über den Querschnitt der jeweiligen Leitungen spielt bei der genaueren Fehlerbetrachtung eine wichtige Rolle.

Positionen der PDUs im Fahrzeug

- PDU1: Fahrerseite Motorraum
- PDU2: Beifahrerseite Motorraum
- PDU3: Fahrerseite Armaturenbrett
- PDU4: Beifahrerseite Kofferraum



	PDU 1	PDU 2	PDU 3	PDU 4
PDU 1	[-]	3m	1m	7m
PDU 2	3m	[-]	3m	4m
PDU 3	1m	3m	[-]	6m
PDU 4	7m	4m	6m	[-]

Abbildung 18: Kabellänge bei unterschiedlichen Verlegungen im Fahrzeug.

Die Abbildung 18 zeigt die Kabellängen im Fahrzeug bei den unterschiedlichen Verlegearten im Fahrzeug. Die angegebenen Längen sind Schätzwerte und dienen dazu, den Querschnitt der Leitungen besser bestimmen zu können.

Für die bessere Berechnung des Querschnitts der Leitungen sind außer den Längenangaben auch die Nennströme von großer Bedeutung.

PDU	SR	Strom [A]	nSR [A]	Gesamt [A]
#1	Lenkung	25	10	35
#2	Lenkung(redundant) + ESP	10+25	20	55
#3	Systemrechner /Fahrerassistenzsystem	15	30	45
#4	[-]	[-]	40	40

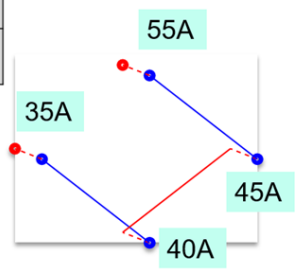


Abbildung 19: Nennströme der einzelnen Verbraucher an den PDUs.

In der Abbildung 19 sind die Nennströme der einzelnen Verbraucher an den PDUs tabellarisch aufgelistet. Die Werte der elektrischen Ströme stammen aus der einhüllenden Topologie. Die Ströme sind aufgeteilt auf die Verbrauchergruppen. So finden sich in der Tabelle die Ströme für die sicherheitsrelevanten und nicht sicherheitsrelevanten Verbraucher wieder. Die Angaben des gesamten Nennstroms sind zur Veranschaulichung in die Topologie Nr.5247 eingezeichnet.

Ausfallszenarien

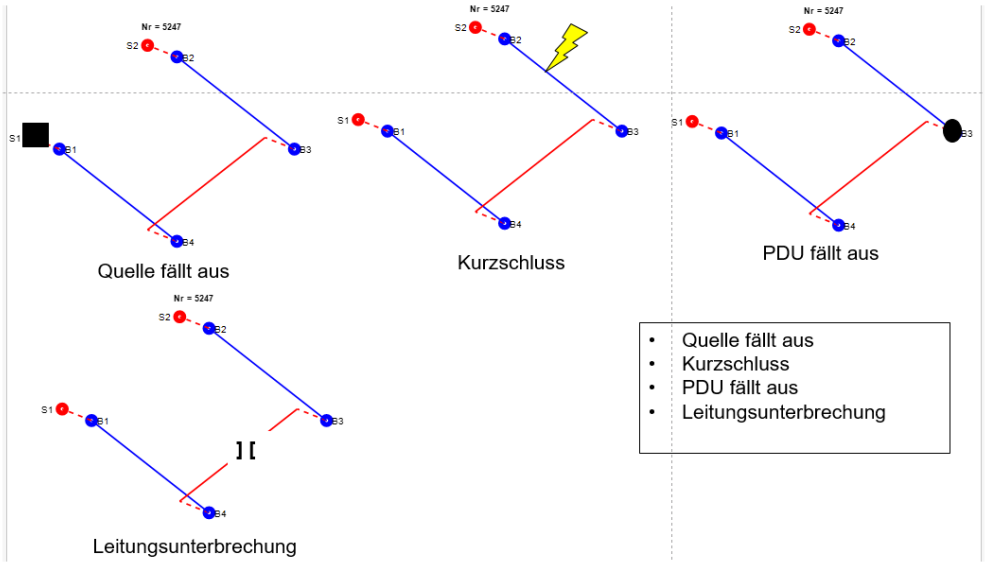


Abbildung 20: Übersicht der untersuchten Fehlerarten.

Die Abbildung 20 zeigt die untersuchten Fehlerarten. Das schwarze Rechteck symbolisiert, dass eine PDU ausgefallen ist. Der gelbe Blitz symbolisiert, dass eine Leitung einen Kurzschluss zur Masse aufweist. Der schwarze Kreis hingegen stellt einen PDU Ausfall dar. Der PDU-Ausfall wird mit einem Kurzschluss zur Masse beschrieben. Das Klammersymbol stellt symbolisch die Unterbrechung einer Leitung dar.

Kalte Redundanz

Redundante Systeme dienen der Ausfallsicherheit in einem möglichen Fehlerfall. In einem Bordnetzsystem können sowohl Leitungen als auch Quellen redundant ausgelegt werden.

Bei der Auslegung redundanter Systeme wird prinzipiell zwischen einer kalten und einer heißen Redundanz unterschieden.

Bei der heißen Redundanz sind die redundanten Pfade bzw. Quellen permanent gleichzeitig im Betrieb. Bei einem möglichen Fehlerfall wird das defekte System abgekoppelt.

Die redundanten Systeme sind hingegen bei der kalten Redundanz zwar verbaut, werden aber erst im Fehlerfall hinzugeschaltet.

Für die Behandlung von Fehlern wird in diesem Abschnitt von der kalten Redundanz ausgegangen. Die kalte Redundanz wird durch die beiden Quellen und durch redundante Leitungsverlegung ermöglicht. Mit Hilfe der verwendeten Schalter in den PDUs können die Schaltvorgänge unternommen werden. Es wird davon ausgegangen, dass ein Energiepuffer für den Schaltvorgang bei unterbrochener Versorgung (Bsp. In Form eines Kondensators) zur Verfügung steht. Des Weiteren wird davon ausgegangen, dass die Leitungen bei auftretender geänderter Strombelastbarkeit im Fehlerfall entsprechend dimensioniert sind.

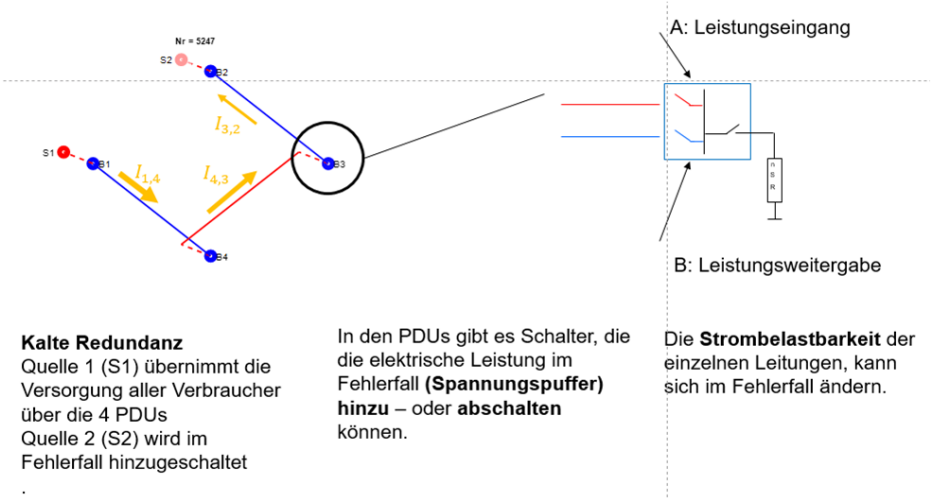
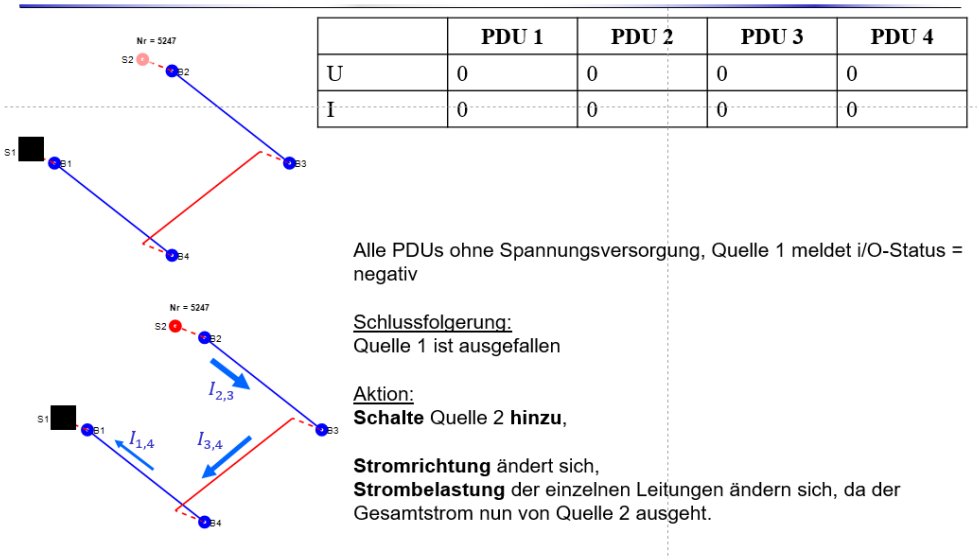


Abbildung 21: Kalte Redundanz und Schalter.

Die Abbildung 21 zeigt an einem Beispiel der Topologie-Nr.5247 die Realisierung einer kalten Redundanz. Außerdem sind in dieser Abbildung die Schalter der PDU drei dargestellt. Die Schalter werden in einem Fehlerfall angesteuert. Somit können die schaltbaren Verbindungen sowie die Backbone-Verbindung weggeschaltet werden. Außerdem kann bei der kalten Redundanz die Quelle 2 hinzugeschaltet werden, so dass eine Spannungsversorgung der im Fehlerfall noch erreichbaren PDUs gewährleistet ist.

Fehlerart: Quelle fällt aus

Anhand der Topologie-Nr.5247 soll beispielhaft das Ausfallszenario, dass die Quelle eins ausfällt beschrieben werden. Die Quelle eins sei bei der Verwendung der kalten Redundanz permanent in Betrieb. Die Quelle eins versorgt das gesamte Bordnetz mit elektrischer Energie. Die Quelle eins speist bei jeder Topologie die elektrische Energie über die PDU eins ein.



	PDU 1	PDU 2	PDU 3	PDU 4
U	0	0	0	0
I	0	0	0	0

Alle PDUs ohne Spannungsversorgung, Quelle 1 meldet i/O-Status = negativ

Schlussfolgerung:
Quelle 1 ist ausgefallen

Aktion:
Schalte Quelle 2 **hinzu**,

Stromrichtung ändert sich,
Strombelastung der einzelnen Leitungen ändern sich, da der Gesamtstrom nun von Quelle 2 ausgeht.

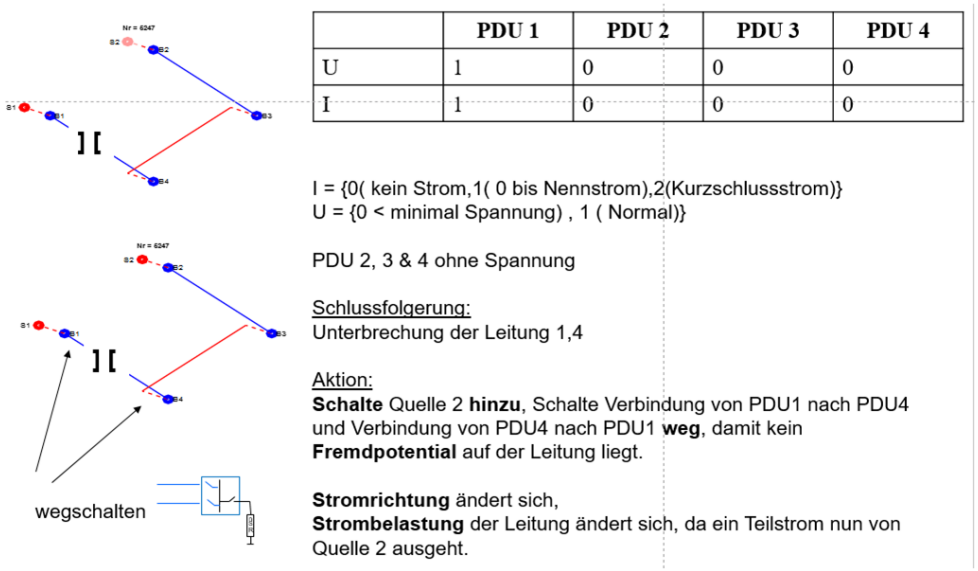
Abbildung 22: Ausfall der Quelle S1.

Die Abbildung 22 zeigt den Ausfall der Quelle S1 anhand der Beispieltoplogie-Nr.5247. Fällt die Quelle S1 aus, so kann kein Strom in dem Bordnetz fließen. Außerdem liegt kein Spannungspotential an den PDUs an. Die Quelle S2 wird nun hinzugeschaltet. Die fehlerhafte Quelle S1 wird vom Bordnetzsystem weggeschaltet. Die Quelle S2 versorgt nun das gesamte Bordnetz mit elektrischer Energie. Der benötigte elektrische Strom fließt nun von der Quelle S2 über die schaltbaren Verbindungen sowie über die Backbone–Verbindung.

Den Fehler, dass die Quelle S2 ausfällt bliebe bei der Betrachtung des einfachen Fehlerfalls unbemerkt. Die Quelle S2 wird immer nur dann hinzugeschaltet, wenn ein Fehler detektiert wird.

Fehlerart: Leitungsunterbrechung

Bei der Leitungsunterbrechung kann die benötigte elektrische Energie nicht mehr durch die fehlerhafte Leitung übertragen werden. An möglichen nachfolgenden PDUs und Verbraucher ist kein Spannungspotential mehr vorhanden.



	PDU 1	PDU 2	PDU 3	PDU 4
U	1	0	0	0
I	1	0	0	0

I = {0(kein Strom,1(0 bis Nennstrom),2(Kurzschlussstrom)}
U = {0 < minimal Spannung) , 1 (Normal)}

PDU 2, 3 & 4 ohne Spannung

Schlussfolgerung:
Unterbrechung der Leitung 1,4

Aktion:
Schalte Quelle 2 **hinzu**, Schalte Verbindung von PDU1 nach PDU4 und Verbindung von PDU4 nach PDU1 **weg**, damit kein **Fremdpotential** auf der Leitung liegt.

Stromrichtung ändert sich,
Strombelastung der Leitung ändert sich, da ein Teilstrom nun von Quelle 2 ausgeht.

Abbildung 23: Leitungsunterbrechung zwischen PDU eins und PDU vier.

Als Beispiel soll die schaltbare Leitung zwischen der PDU eins und der PDU vier unterbrochen sein. Dies ist in der Abbildung 23 grafisch dargestellt. Für diesen expliziten Fehlerfall sind die PDUs zwei, drei und vier ohne Spannungspotential und ohne Strom. Lediglich die PDU eins wird noch mit elektrischer Energie versorgt.

Für diesen Fehlerfall wird die schaltbare Leitung zwischen PDU eins und PDU vier in der PDU eins weggeschaltet. Somit liegt kein Potential auf der Leitung, welches zu weiteren Fehlern führen könnte. In der PDU vier wird die Leitung ebenfalls weggeschaltet. Dadurch kann kein Potential von der PDU vier auf die defekte Leitung weitergegeben werden.

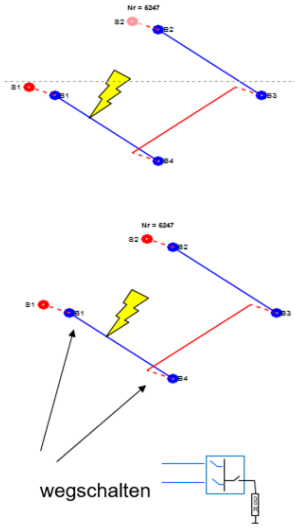
Die Quelle zwei wird nun hinzugeschaltet.

Alle PDUs sind bei dieser Fehlerart der Leitungsunterbrechung mit elektrischer Energie versorgt. Die PDU eins über die Quelle S1, die PDUs zwei, drei und vier über die Quelle S2.

Die Leitungsunterbrechungen können selbstverständlich auch in der anderen schaltbaren Verbindung sowie in der Backbone-Verbindung auftreten. Die dazugehörigen Ergebnisse befinden sich im Anhang.

Fehlerart: Kurzschluss auf Leitung

Bei der Fehlerart Kurzschluss auf einer Leitung soll der Kurzschluss immer gegen die Fahrzeugmasse gehen. Als Beispiel soll die schaltbare Leitung zwischen der PDU eins und der PDU vier einen Kurzschluss aufweisen.



PDU 1	PDU 2	PDU 3	PDU 4
Hoher Strom	Keine Spannung	Keine Spannung	Keine Spannung

PDU 2, 3 & 4 ohne Spannung, PDU1 mit vielfachem über Nennstrom (=hoher Strom)

Schlussfolgerung:
Massenkurzschluss der Leitung 1,4

Aktion:
Schalte Quelle 2 **hinzu**, Schalte Verbindung von PDU1 nach PDU4 und Verbindung von PDU4 nach PDU1 **weg**, damit kein **Fremdpotential/Kurzschluss** auf der Leitung liegt.

Stromrichtung ändert sich,
Strombelastung der Leitung ändert sich, da ein Teilstrom nun von Quelle 2 ausgeht.

Abbildung 24: Fehlerfall Kurzschluss auf der Leitung zwischen PDU eins und PDU vier.

Dieser Sachverhalt ist in der Abbildung 24 dargestellt. Für diesen expliziten Fehlerfall sind die PDUs zwei, drei und vier ohne Spannungspotential und ohne Strom. Lediglich die PDU eins wird noch mit elektrischer Energie versorgt.

Für diesen Fehlerfall wird die schaltbare Leitung zwischen PDU eins und PDU vier in der PDU eins weggeschaltet. Somit liegt kein Potential auf der Leitung, welches zu weiteren Fehlern führen könnte. Außerdem wird der Kurzschlussstrom durch das Wegschalten unterbrochen. In der PDU vier wird die Leitung ebenfalls weggeschaltet. Dadurch kann kein Potential von der PDU vier auf die defekte Leitung weitergegeben werden. Außerdem wird ein möglicher neuer Kurzschlussstrom durch die PDU vier von der Quelle S2 unterbunden.

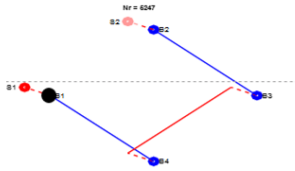
Die Quelle zwei wird nun hinzugeschaltet.

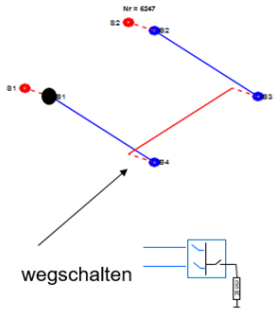
Alle PDUs sind bei dieser Fehlerart des Leitungskurzschlusses mit elektrischer Energie versorgt. Die PDU eins über die Quelle S1, die PDUs zwei, drei und vier über die Quelle S2.

Die Kurzschlüsse auf den Leitungen können selbstverständlich auch in der anderen schaltbaren Verbindung sowie in der Backbone-Verbindung auftreten. Die dazugehörigen Ergebnisse befinden sich im Anhang.

Fehlerart: PDU fällt aus

Der Fehler PDU fällt aus, beschreibt einen Kurzschluss der Potentialschiene gegen Masse. Das bedeutet, dass die PDU weder die elektrische Energie an weitere PDUs weiterleiten kann, noch können die Verbraucher an der PDU mit elektrischer Energie versorgt werden.





PDU 1	PDU 2	PDU 3	PDU 4
Keine Spannung	Keine Spannung	Keine Spannung	Keine Spannung

PDU 1, 2, 3 & 4 ohne Spannung, Quelle 1 meldet i/O= positiv

Schlussfolgerung:
PDU 1 ist ausgefallen

Aktion:
Schalte Quelle 2 **hinzu**, Schalte Verbindung von PDU4 nach PDU1 **weg**, damit kein **Fremdpotential** auf der Leitung liegt. (Verbindung PDU1 - PDU4 kann nicht weggeschaltet werden, da defekt)

Stromrichtung ändert sich,
Strombelastung der Leitung ändert sich, da ein Teilstrom nun von Quelle 2 ausgeht.

Abbildung 25: Fehlerfall PDU eins fällt aus.

Die Abbildung 25 zeigt den Fehlerfall, dass die PDU eins ausgefallen ist. Für diesen expliziten Fehlerfall sind alle PDUs ohne Spannung. Die Quelle S1 hingegen meldet ein Signal, dass diese in Ordnung sei. So kann dieser Fehler von dem Fehler, dass die Quelle eins ausgefallen ist unterschieden werden.

Der Schalter in der PDU vier schaltet die schaltbare Verbindung zwischen PDU1 und PDU4 weg. Dadurch kann der Kurzschlussstrom, welcher von der Quelle S2 über die PDU vier fliesen würde, vorzeitig unterbunden werden.

Die Quelle zwei wird nun hinzugeschaltet.

Die PDUs zwei, drei und vier werden nun von der Quelle zwei versorgt. Die PDU eins ist in diesem Fehlerfall ohne Funktion. Außerdem können die Verbraucher von der PDU eins, welche ausschließlich von der PDU eins versorgt werden, nicht mehr verwendet werden.

Die anderen PDUs können selbstverständlich auch ausfallen. Diese dazugehörigen Ergebnisse befinden sich im Anhang.

AP 2-3 Untersuchung der Erkennung und Behandlung von Fehlern

In einer konventionellen Baumstruktur erfolgen die Erkennung und die Behandlung von Fehlern mit thermischen oder mit elektronischen Sicherungen. Das Sicherungselement ist somit sowohl Messelement als auch ein Schaltelement. Bei einer entsprechenden Dimensionierung ist das Sicherungselement in der Lage einen Fehler lokal zu detektieren und erfolgreich selektiv zu trennen.

Bei einer redundanten Versorgungsstruktur wird, abhängig von Fehlerfall und ob es sich um kalte oder heiße Redundanz handelt, zusätzliche Sensorik, Aktorik und Kommunikation benötigt, um Fehlerfälle detektieren, lokalisieren, identifizieren und darauf reagieren zu können.

Anhand der Beispieltopologie-Nr.5247 wird gezeigt, wo die Sensoren und die Messpunkte sich befinden werden.

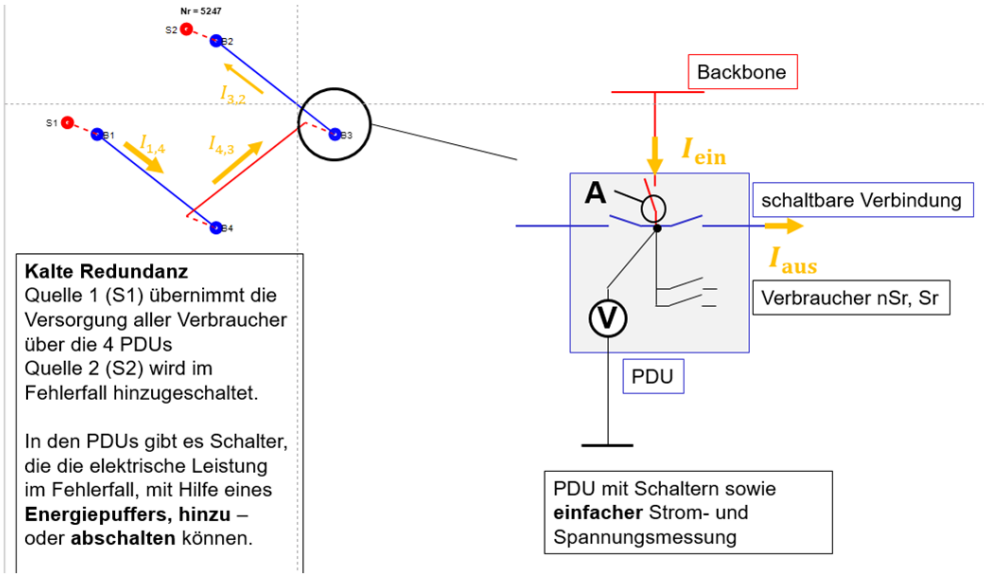


Abbildung 26: Spannungs-und Strommessung innerhalb der PDU.

Die Abbildung 26 zeigt die Schalter innerhalb der PDU drei. Dort sind außerdem eine Spannungsmessung auf der Potentialschiene und eine Strommessung quellenseitig eingezeichnet. Mit Hilfe dieser beiden Messungen in einer PDU können die Spannung und Ströme gemessen werden. So kann u.a. die Information über eine fehlende Spannung gewonnen werden. Des Weiteren können die Kurzschlussströme erfasst werden. Mit Hilfe dieser Informationen, lassen sich, für einen Bewertung auf logischer Ebene, boolesche Variablen erzeugen.

$$U(PDU\ j) = \begin{cases} 1 & \text{für Spannung ist i. O.} \\ 0 & \text{für Spannung ist n. i. O.} \end{cases} \text{mit } j = 1,..,4$$

Die Spannung sei in Ordnung für einen Bereich in dem das Bordnetz ohne Probleme funktioniert U ≈ 10V,..,14V. Fällt die Spannung unter einen Grenzwert, so dass die

Verbraucher nicht mehr arbeiten können, so ist die Spannung nicht mehr in Ordnung. U < 10V.

Für den Strom ergibt sich folgende Notation. Da unbekannt ist, ob ein Verbraucher aktiv oder inaktiv ist, ist ein Strom von I = 0A auch im fehlerfreien Fall zu erwarten. Ist der Strom um ein Vielfaches größer als der Nennstrom, so liegt ein Fehler, ggf. ein. Kurzschluss, vor.

$$I(PDU\,j) = \begin{cases} 0 & \text{für Strom ist i. O.} \\ 1 & \text{für Strom ist n. i. O.} \end{cases} \text{mit } j = 1,..,4$$

Mit Hilfe dieser Notation, lassen sich logische Funktionen generieren, welche einen Rückschluss auf den Status der jeweiligen PDUs zulassen.

											Logische Funktion (Aktion)									
										Fehler	PDU1		PDU2		PDU3		PDU4	
	Fehlerfall	U(PDU1)	U(PDU2)	U(PDU3)	U(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)	vorhanden	SS1	SL1.4	SS2*	SL2.3	SL3.2	BB3	SL4.1	BB4
1	kein Fehler	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Quelle S1 fällt aus	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
3	Quelle S2 fällt aus	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Unterbrechung L1,4	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0
5	Unterbrechung BB3,4	1	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	1
6	Unterbrechung L2,3	1	0	1	1	0	0	0	0	1	0	0	1	1	1	0	0	0
7	Kurzschluss L1,4	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0
8	Kurzschluss BB3,4	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	1
9	Kurzschluss L2,3	0	0	0	0	1	0	1	1	1	0	0	1	1	1	0	0	0
10	PDU1 fällt aus	0	0	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0
11	PDU2 fällt aus	0	0	0	0	1	1	1	1	1	0	0	0	1	1	0	0	0
12	PDU3 fällt aus	0	0	0	0	1	0	1	1	1	0	0	1	1	1	1	0	1
13	PDU4 fällt aus	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	1	1

Unterschiedliche Fehlerfälle die anhand der Sensordaten nicht **unterscheidbar** sind.
-> erforderliche Aktionen können nicht situationspassend ausgeführt werden.

Bei dem Fehlerfall *PDU fällt aus*, soll die PDU sowohl **quellen-** aus auch **senkenseitig** abgekoppelt werden, damit der Ort des Kurzschluss vom BN getrennt werden kann.

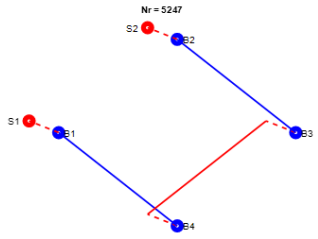


Abbildung 27: Übersicht der Spannungs- und Stromzustände in den PDUs.

Die Abbildung 27 zeigt in tabellarischer Form die Spannungs- und Stromzustände in den PDUs. Die zu erwartbaren Spannungs- und Stromzustände sind in Abhängigkeit der jeweiligen Fehlerarten und Fehlerorten aufgelistet.

Die farbig unterlegten Zeilen, zeigen auf, dass sich der Status der einzelnen Messsensoren wiederholt, obwohl unterschiedliche Fehlerfälle vorliegen.

Die Abbildung 28 zeigt, dass die Strommessung in den PDUs sowohl quellen- als auch senkenseitig erfolgt. Über die Differenz der beiden Messergebnisse, kann auf den Verbraucherstrom der angeschlossenen Verbraucher geschlossen werden. Des Weiteren können durch die erweiterte Strommessung die Fehlerarten unterschieden werden. Dieser Sachverhalt ist in der Abbildung 29 tabellarisch dargestellt.

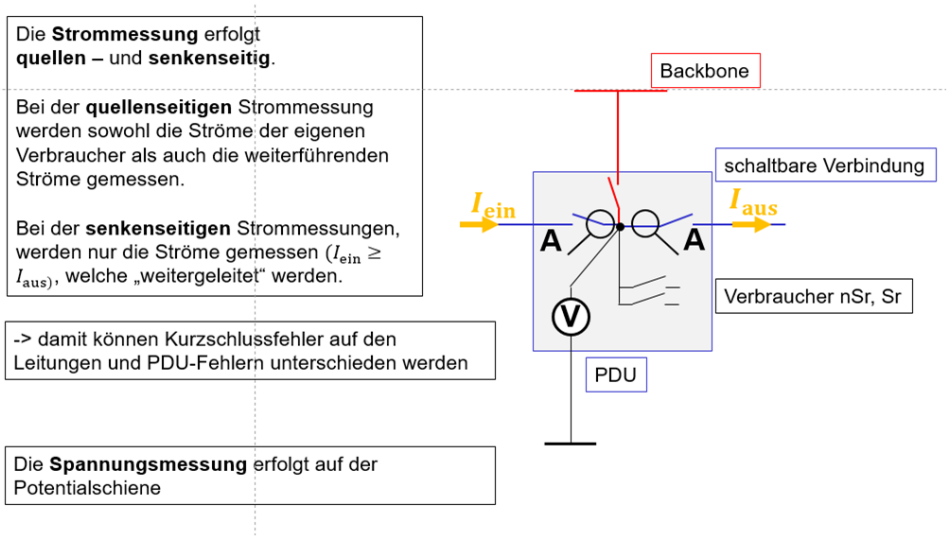
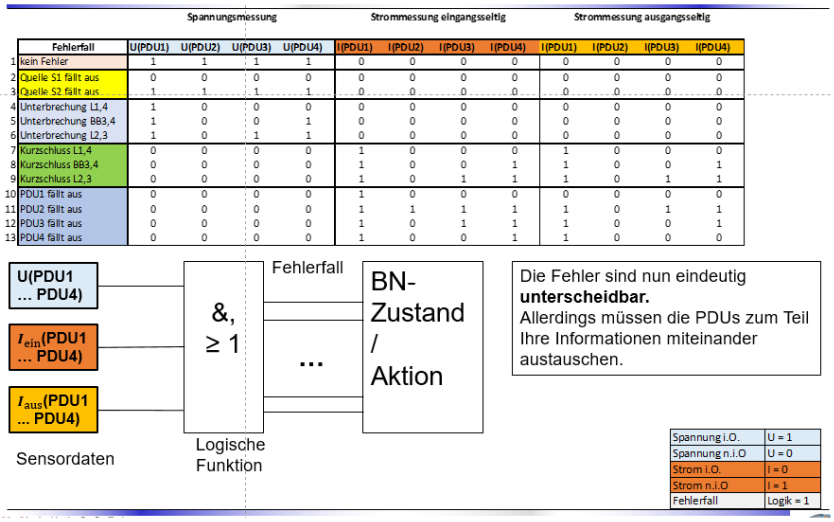


Abbildung 28: Strommessung in den PDUs sowohl quellen- als auch senkenseitig.



		Spannungsmessung				Strommessung eingangsseitig				Strommessung ausgangsseitig			
	Fehlerfall	U(PDU1)	U(PDU2)	U(PDU3)	U(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)
1	kein Fehler	1	1	1	1	0	0	0	0	0	0	0	0
2	Quelle S1 fällt aus	0	0	0	0	0	0	0	0	0	0	0	0
3	Quelle S2 fällt aus	1	1	1	1	0	0	0	0	0	0	0	0
4	Unterbrechung L1,4	1	0	0	0	0	0	0	0	0	0	0	0
5	Unterbrechung BB3,4	1	0	0	1	0	0	0	0	0	0	0	0
6	Unterbrechung L2,3	1	0	1	1	0	0	0	0	0	0	0	0
7	Kurzschluss L1,4	0	0	0	0	1	0	0	0	1	0	0	0
8	Kurzschluss BB3,4	0	0	0	0	1	0	0	1	1	0	0	1
9	Kurzschluss L2,3	0	0	0	0	1	0	1	1	1	0	1	1
10	PDU1 fällt aus	0	0	0	0	1	0	0	0	0	0	0	0
11	PDU2 fällt aus	0	0	0	0	1	1	1	1	1	0	1	1
12	PDU3 fällt aus	0	0	0	0	1	0	1	1	1	0	0	1
13	PDU4 fällt aus	0	0	0	0	1	0	0	1	1	0	0	0

Abbildung 29: Übersicht der Spannungs- und Stromzustände in den PDUs mit quellen- und senkenseitigen Strommessungen.

Erkennung von Fehlern

Mit Hilfe der verschiedenen Sensoren in den PDUs lassen sich eindeutige Aussagen über die Fehlerart und der Lokation tätigen. Aus diesen Informationen können auf die erforderlichen Maßnahmen geschlussfolgert werden. Die Abbildung 30 zeigt symbolisch, wie

die Schlussfolgerung auf die Fehlerart und Lokalisation durch einzelne Sensorinformationen von statten geht.

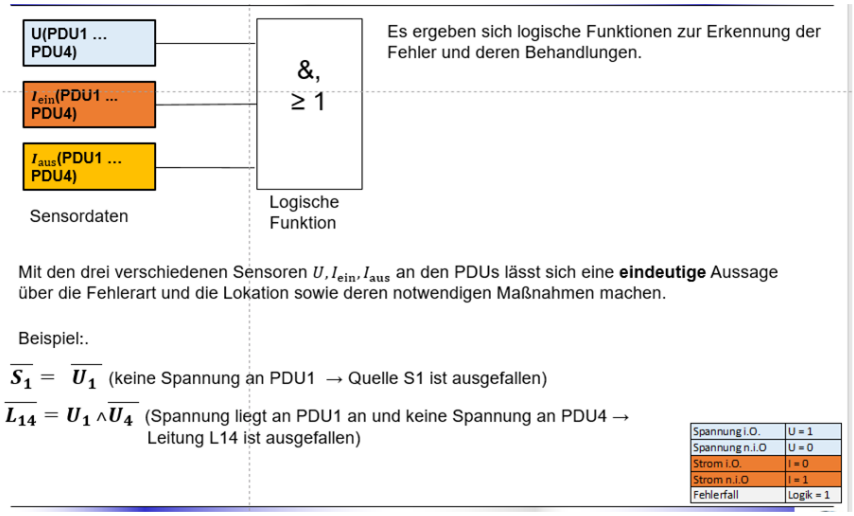


Abbildung 30: Schlussfolgerung durch Sensorinformationen.

Behandlung von Fehlern

Über die Statusinformationen von der Spannung, dem quellen- sowie dem senkenseitigen Strom lassen sich die Fehler erkennen. Außerdem können mit diesen Informationen boolesche Funktionen für die jeweiligen Schalter gebildet werden. Die Schalter in einer PDU lassen sich somit über die Informationen der Sensoren in den PDUs logisch schalten. Die Schalter sind dazu da, den Fehler zu selektiv zu trennen und ggf. eine weitere Versorgung für die verbliebenen PDUs aufzubauen. Die Abbildung 31 und Abbildung 32 zeigen die logischen Funktionen der Schalter für die Quellen, für die schaltbaren Verbindungen sowie für die Backbone-Verbindungen.

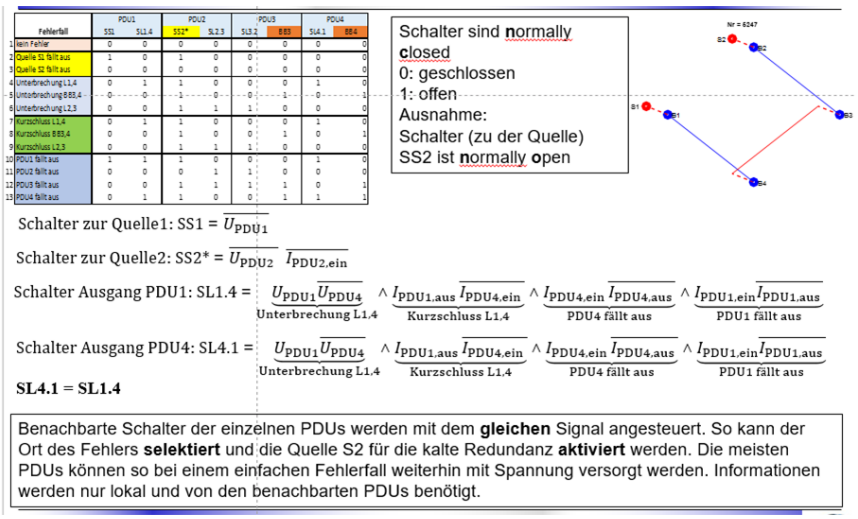


Abbildung 31: Schalteraktivitäten der Quellen und schaltbaren Verbindungen.

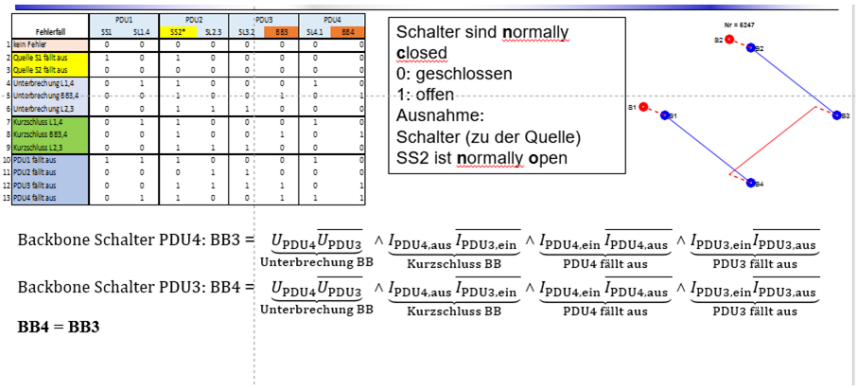
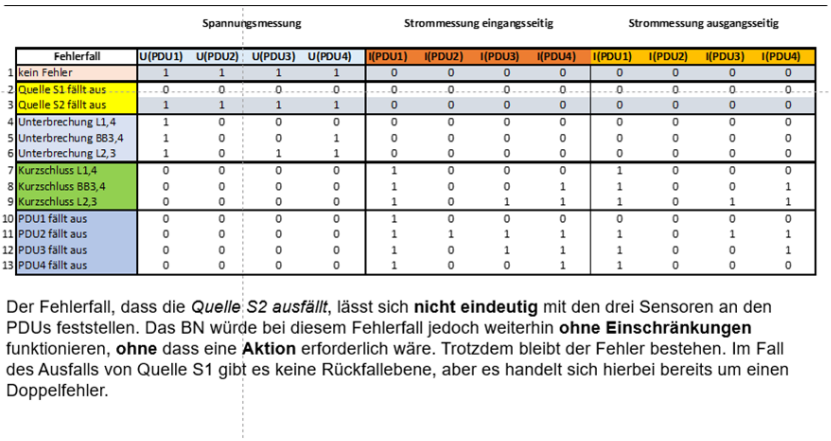


Abbildung 32 : Schalteraktivitäten der Backbone-Verbindungen.

Universität Kassel
FB16 | Fahrzeugsysteme und Grundlagen der Elektrotechnik
Prof. Dr. rer. nat. Ludwig Brabetz

Fehlerfall Quelle 2 fällt aus

Der Fehlerfall, dass die Quelle S2 ausfällt, lässt sich mit den beschriebenen Sensoren nicht eindeutig bestimmen. Das liegt daran, dass die Quelle S2 bei der kalten Redundanz, erst im Fehlerfall hinzugeschaltet wird. Das Bordnetz würde bei diesem Fehlerfall jedoch weiterhin ohne Einschränkungen funktionieren, ohne das eine Aktion erforderlich wäre. Die *Abbildung 33* stellt den Fehlerfall Quelle S2 fällt aus grafisch dar. Durch eine zusätzliche Spannungsmessung, lässt sich dieser Fehler eindeutig detektieren. Die Spannungsmessung erfolgt an dem Schalter, an dem die Quelle S2 angeschlossen ist. Hierbei ist es unabhängig ob die Quelle S2 an der PDU eins oder an der PDU2 sich befindet. Es sei erwähnt, dass es sich hierbei um einen Doppelfehler handelt. Bei diesem Fehler, würde es keine Rückfallebene für die Spannungsversorgung für das Bordnetz geben.



		Spannungsmessung				Strommessung eingangsseitig				Strommessung ausgangsseitig			
	Fehlerfall	U(PDU1)	U(PDU2)	U(PDU3)	U(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)
1	kein Fehler	1	1	1	1	0	0	0	0	0	0	0	0
2	Quelle S1 fällt aus	0	0	0	0	0	0	0	0	0	0	0	0
3	Quelle S2 fällt aus	1	1	1	1	0	0	0	0	0	0	0	0
4	Unterbrechung L1,4	1	0	0	0	0	0	0	0	0	0	0	0
5	Unterbrechung BB3,4	1	0	0	1	0	0	0	0	0	0	0	0
6	Unterbrechung L2,3	1	0	1	1	0	0	0	0	0	0	0	0
7	Kurzschluss L1,4	0	0	0	0	1	0	0	0	1	0	0	0
8	Kurzschluss BB3,4	0	0	0	0	1	0	0	1	1	0	0	1
9	Kurzschluss L2,3	0	0	0	0	1	0	1	1	1	0	1	1
10	PDU1 fällt aus	0	0	0	0	1	0	0	0	0	0	0	0
11	PDU2 fällt aus	0	0	0	0	1	1	1	1	1	0	1	1
12	PDU3 fällt aus	0	0	0	0	1	0	1	1	1	0	0	1
13	PDU4 fällt aus	0	0	0	0	1	0	0	1	1	0	0	0

Der Fehlerfall, dass die *Quelle S2 ausfällt*, lässt sich **nicht eindeutig** mit den drei Sensoren an den PDUs feststellen. Das BN würde bei diesem Fehlerfall jedoch weiterhin **ohne Einschränkungen** funktionieren, **ohne** dass eine **Aktion** erforderlich wäre. Trotzdem bleibt der Fehler bestehen. Im Fall des Ausfalls von Quelle S1 gibt es keine Rückfallebene, aber es handelt sich hierbei bereits um einen Doppelfehler.

Abbildung 33: Fehler Quelle S2 fällt aus.

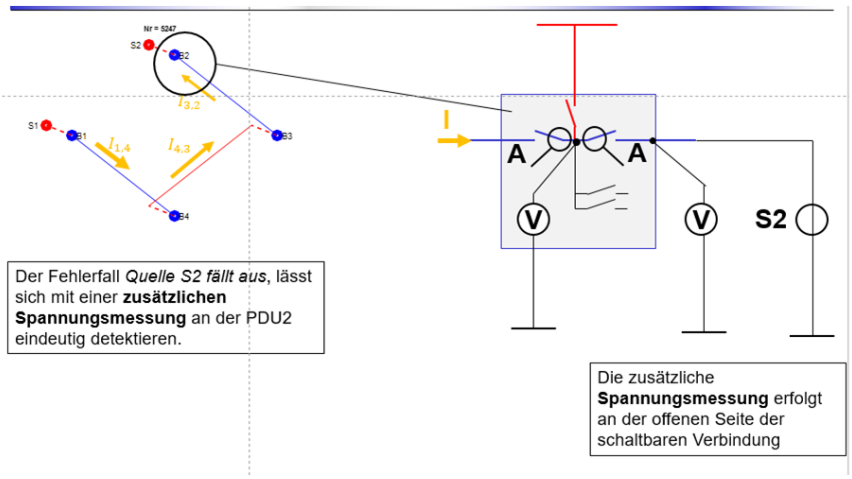
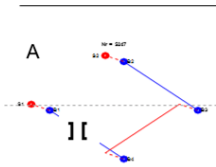
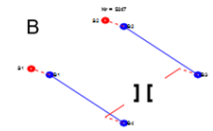


Abbildung 34. Zusätzliche Spannungsmessung an dem Schalter zur Quelle S2.

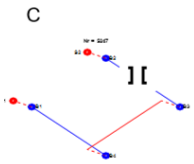


	PDU 1	PDU 2	PDU 3	PDU 4
A	1	0	0	0
B	1	Keine Spannung	Keine Spannung	Spannung
C	1	Keine Spannung	Spannung	Spannung

Schlussfolgerung:



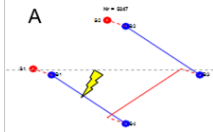
	PDU 1	PDU 2	PDU 3	PDU 4
A	L1,4 = 0			Unterbrechung
B			Unterbrechung	
C		Unterbrechung		



Aktion:
Schalte Quelle 2 **hinzu**, **Wegschalten** der Verbindungen von...

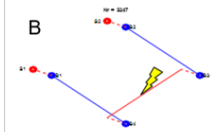
	PDU 1	PDU 2	PDU 3	PDU 4
A	open L1,4	Close S2		open L1,4
B		Close S2	Open L3,4	Open L3,4
C		Open L2,3 close S2	Open L2,3	

Abbildung 35:Fehlerart Leitungsunterbrechungen.

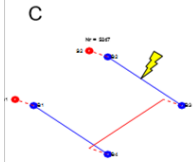


	PDU 1	PDU 2	PDU 3	PDU 4
A	Hoher Strom			
B	Hoher Strom			Hoher Strom
C	Hoher Strom		Hoher Strom	Hoher Strom

Schlussfolgerung:



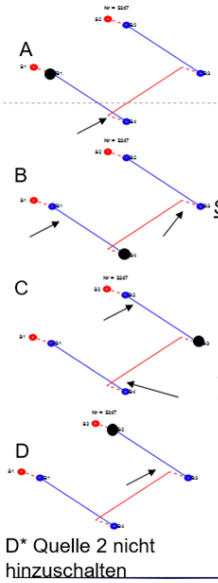
	PDU 1	PDU 2	PDU 3	PDU 4
A	Kurzschluss			Kurzschluss
B			Kurzschluss	
C		Kurzschluss		



Aktion:
Schalte Quelle 2 **hinzu**, **Wegschalten** (Vermeidung Fremdpotential/Kurschluss) der Verbindungen von...

	PDU 1	PDU 2	PDU 3	PDU 4
A	Wegschalten			Wegschalten
B			Wegschalten	Wegschalten
C		Wegschalten	Wegschalten	

Abbildung 36: Fehlerfall Kurzschluss auf den Leitungen.



	PDU 1	PDU 2	PDU 3	PDU 4
A	Keine Spannung	Keine Spannung	Keine Spannung	Keine Spannung
B	Spannung	Keine Spannung	Keine Spannung	Keine Spannung
C	Spannung	Keine Spannung	Keine Spannung	Spannung
D	Spannung	Keine Spannung	Spannung	Spannung

& Quelle 1 i/O = positiv

Schlussfolgerung:

A	Ausfall			
B				Ausfall
C			Ausfall	
D		Ausfall		

Aktion:

Schalte Quelle 2 **hinzu**, **Wegschalten** der PDU...

	PDU 1	PDU 2	PDU 3	PDU 4
A				wegschalten
B	wegschalten		wegschalten	
C		wegschalten		wegschalten
D*			wegschalten	

Abbildung 37: Fehlerfall PDU fällt aus.

	Fehlerfall	Spannungsmessung				Strommessung eingangsseitig				Strommessung ausgangsseitig				Fehler vorhanden	PDU1		PDU2		PDU3		PDU4	
		U(PDU1)	U(PDU2)	U(PDU3)	U(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)	I(PDU1)	I(PDU2)	I(PDU3)	I(PDU4)		SS1	SL1.4	SS2*	SL2.3	SL3.2	BB3	SL4.1	BB4
1	kein Fehler	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Quelle S1 fällt aus	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
3	Quelle S2 fällt aus	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Unterbrechung L1,4	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	0
5	Unterbrechung BB3,4	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	1
6	Unterbrechung L2,3	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0
7	Kurzschluss L1,4	d	d	d	d	1	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0
8	Kurzschluss BB3,4	d	d	d	d	1	0	0	1	1	0	0	1	1	0	0	1	0	0	1	0	1
9	Kurzschluss L2,3	d	d	d	d	1	0	1	1	1	0	1	1	1	0	0	1	1	1	0	0	0
10	PDU1 fällt aus	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0
11	PDU2 fällt aus	1	0	1	1	1	1	1	1	1	0	1	1	1	0	0	0	1	1	0	0	0
12	PDU3 fällt aus	1	0	0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	1	1	0	1
13	PDU4 fällt aus	1	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	0	0	1	1	1

Abbildung 38: Spannungs- und Stromzustände der PDUs in boolescher Form .

Teil 3

Simulationsautomatisierung und Datenformate zur Topologiedefinition

Sven Reitz
Thomas Markwirth

Fraunhofer Institut für Integrierte Schaltungen IIS
Institutsteil Entwicklung Adaptiver Systeme EAS

Inhaltsverzeichnis

1 Untersuchung von Datenformaten zur Beschreibung von Topologien, Komponenten und deren Beanspruchung

Anspruch im Projekt ist eine effiziente und daher automatisierte Bewertung von Bordnetztopologien. Für die Weiterverarbeitung in den Toolumgebungen der Partner im Konsortium sind gleichzeitig verschiedene Im- und Exportfunktionalitäten sowie ein einheitliches Austauschformat für die Daten zwingend.

1.1 Datenaustauschformate KBL/VEC

Zuerst wurden die Austauchformate KBL und VEC betrachtet. KBL steht für Kabelbaumliste und liegt aktuell in der Version 2 vor. Da im KBL jedoch Geometriedaten, Bauteilbeschreibungen, Vernetzungsdaten zum Erstellen der Reparaturanleitungen und Stücklisten fehlten, wurden drei zum Kabelbaummodell KBL ergänzenden Datenmodelle erstellt:

- die Komponentenbeschreibung KOMP
- das Geometriemodell GEO
- das Elektrologikmodell ELOG

Vorteil ist, dass KBL standardisiert ist und beim Projektpartner TU Dortmund Tools zur Netzlistengenerierung aus KBL vorhanden sind. Nachteilig sind jedoch die hohe Komplexität und dass es absehbar durch das neuere Format VEC abgelöst wird.

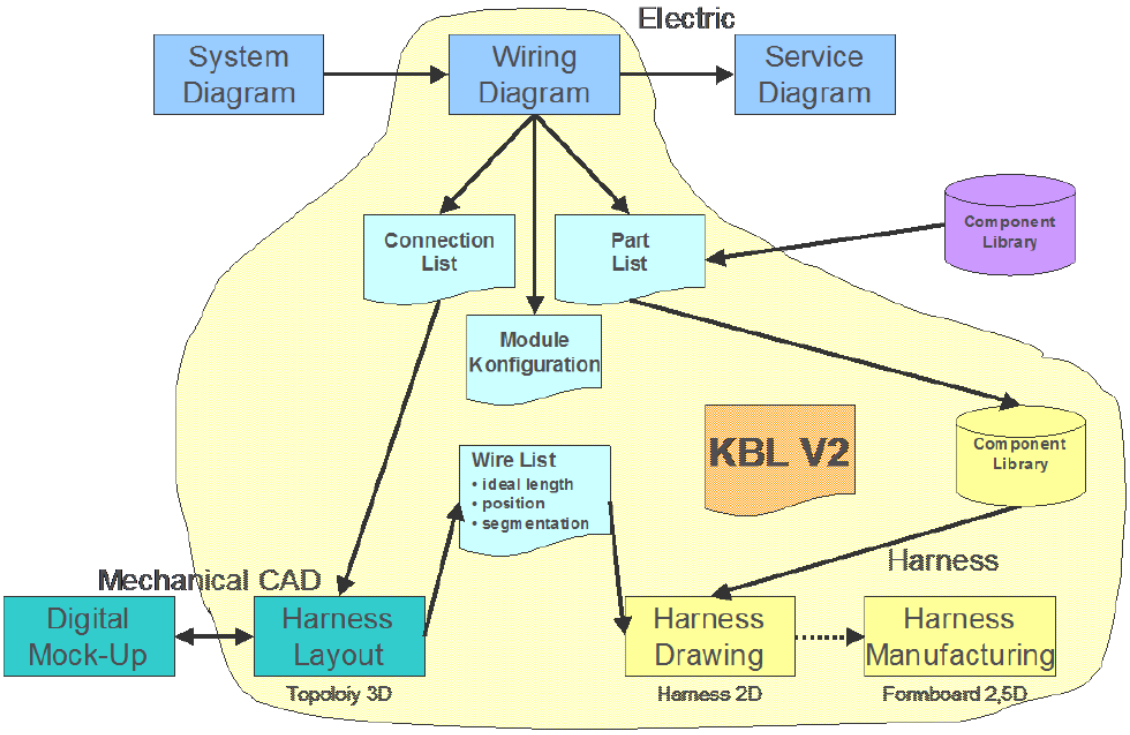


Abbildung 1: Datenaustauschformat KBL

Die vier Teilmodelle zum Austausch von Bordnetzdaten wurden konsolidiert und zu einem Gesamtmodell, dem Vehicle Electric Container (VEC) zusammengeführt. Dieses Gesamtmodell für das physische Bordnetz unterstützt mit seinen vier Teilmodellen nicht nur den internen Datentransfer zwischen den einzelnen Prozessschritten und den beteiligten Tools, sondern ist auch für den Austausch der Daten mit externen Partnern geeignet. Sogar zum Ablegen, Speichern und zur Langzeitarchivierung von physischen Bordnetzen lässt sich dieses Datenmodell verwenden [1].

Vorteilig sind bei diesem Format innerhalb des Projekts seine Standardisierung und Zukunftsfähigkeit. Nachteilig ist jedoch dessen sehr hohe Komplexität und der Umstand, dass neue Konverter bzw. Tools notwendig werden würden. Dennoch wäre eine Anwendung, wenn dieses Format bereits heute oder in Zukunft bei einem der Industriepartner eingesetzt wird.

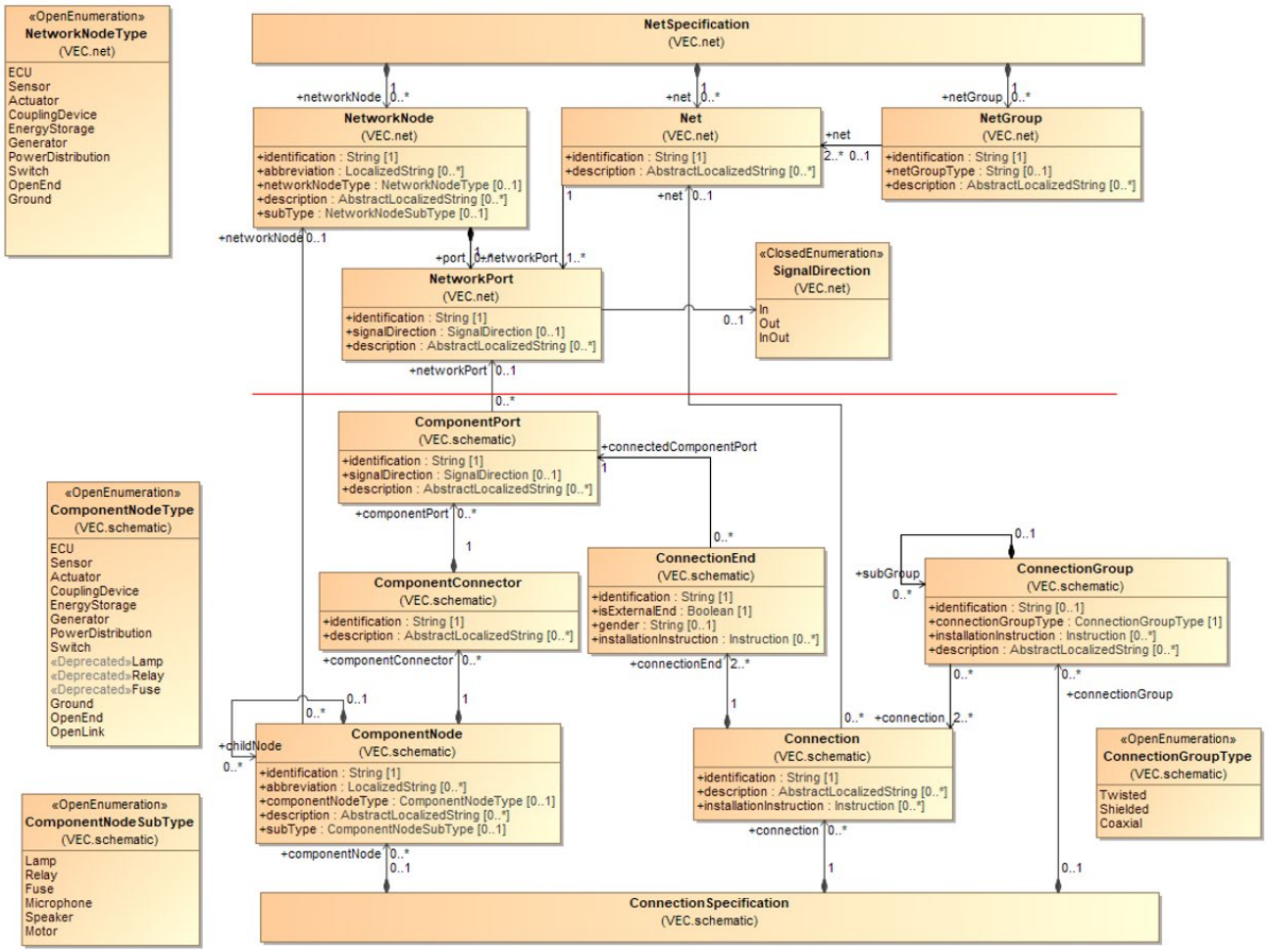


Abbildung 2: Datenaustauschformat VEC

Ziel war ein automatisierter Generierungsflow für Netzlisten zur simulativen Evaluierung bzw. Auswahl der optimalen Topologie, ausgehend einer Vorauswahl potenziell geeigneter Topologien und der durch die jeweilige Bordnetzarchitektur definierten Quellen und Senken. Die folgende Abbildung zeigt hierzu den prinzipiellen Ansatz zur Konvertierung der unterschiedlichen Formate als Ergebnis einer projektinternen Abstimmung.

Ergebis war dabei die Definition eines einheitlichen JSON-Zwischenformats. Auf dieses Format wird in den nächstem Aschnitten eingegangen. Input sind die Basis- bzw. Grundtopologie, welche in KBL oder in einem eigenen JSON-Format vorliegen können. Für KBL wurde eine Importmöglichkeit in das JSON-Format entwickelt. Die einzelnen Bordnetztopologien (zu untersuchende Topologievarianten) liegen im Matlab-Savefileformat vor und benötigten ebenso eine Entwicklung von Importfunktionen. Beschreibungen von Fehlermodellen erfolgen ebenso im JSON-Format und sind zusammen mit dem o.g. Zwischenformat Grundlage für die automatisierte Modellgenierung und Simulation. Erzeugt werden können aktuell Modelica- und Spice-(source-)codes. Nachfolgendes Bild verdeutlicht den Workflow.

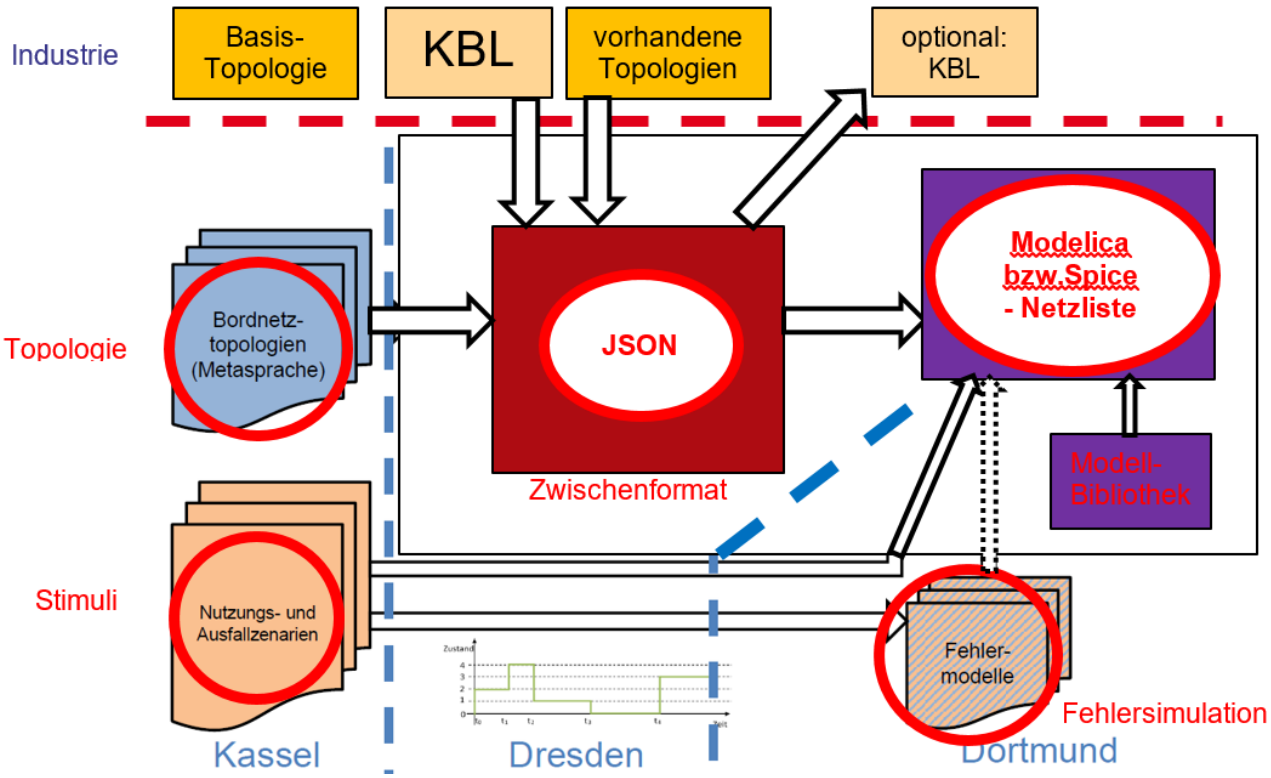


Abbildung 3: Schnittstellen und Austauschformate für die Zusammenarbeit im Projekt

1.2 JSON – ideales Format für Datenaustausch

Das JSON-Format ist auf https://www.json.org/json-de.html genau beschrieben. Hier kurz ein Auszug daraus:

JSON (JavaScript Object Notation) ist ein schlankes Datenaustauschformat, das für Anwender einfach zu lesen und zu schreiben und für Maschinen einfach zu parsen (Analysieren von Datenstrukturen) und zu generieren ist.

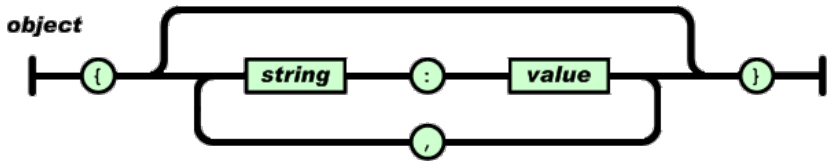
Bei JSON handelt es sich um ein Textformat, das komplett unabhängig von Programmiersprachen ist, aber vielen Konventionen folgt, die Programmieren aus der Familie der C-basierten Sprachen (inklusive C, C++, C#, Java, JavaScript, Perl, Python und vielen anderen) bekannt sind. Diese Eigenschaften machen JSON zum idealen Format für Datenaustausch.

JSON baut auf zwei Strukturen auf:

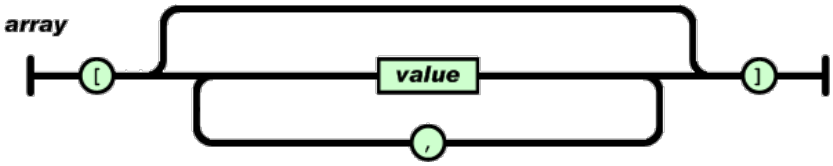
- **Name/Wert Paare**
- **Eine geordnete Liste von Werten**

In JSON gibt es:

Objekte: Ein Objekt ist eine ungeordnete Menge von Name/Wert Paaren. Ein Objekt beginnt mit **{** *geschwungene Klammer auf* und endet mit **}** *geschwungene Klammer zu*. Jedem Namen folgt ein **:** *Doppelpunkt* gefolgt vom Wert und die einzelnen Name/Wert Paare werden durch **,** *Komma* voneinander getrennt.



Ein **Array** ist eine geordnete Liste von Werten. Arrays beginnen mit **[** e*ckige Klammer auf* und enden mit **]** *eckige Klammer zu*. Werte werden durch **,** *Komma* voneinander getrennt.



Ein **Wert** kann ein Objekt, ein Array, eine Zeichenkette (string), eine Zahl oder einer der Ausdrücke **true**, **false** oder **null** sein. Diese Strukturen können ineinander verschachtelt sein.

1.3 JSON-Format für Grundtopologien

Um die extrahierten Daten von Grundtopologien zusammen mit den zu untersuchenden Topologievarianten sowie Fehlerszenarien einfach verarbeiten zu können, wurde zu deren Ablage ein JSON-Format definiert. Dies soll hier kurz erläutert werden.

Das Schlüsselwort topology leitet die Beschreibung ein. Darunter liegen dann title, nodes, sources, connections und configuration. Title ist selbserklärend. Unter den nodes sind die Komponenten wie Generator, Batterie, Sicherungen, … mit ihren Typen, IDs und allen notwendigen Parametern hinterlegt. Der wichtige Abschnitt sources enthält die Quellen, welche

dann in der Gesamtschaltung/Topologie mit den PDUs der Topologievarianten verschalten werden. Unter connections stehen sämtliche Leitungen/Wires mit ihren Typen, IDs, Start- sowie Endknoten und den Parametern wie unter anderem Länge und Querschnitt. Unter configuration können weiter Konfigurationsdaten und -varianten abgelegt werden.

Eine Grundtopologie ist im folgenden Bild dargestellt, gefolgt von einem kurzen Ausschnitt der zugehörigen Beschreibungsdatei mit den wichtigsten Abschnitten. Die komplette Beschreibungsdatei ist im Anhang aufgeführt. Es ist einfach möglich, das Beschreibungsformat an neue/weitere Bedürfnisse und Inhalte anzupassen.

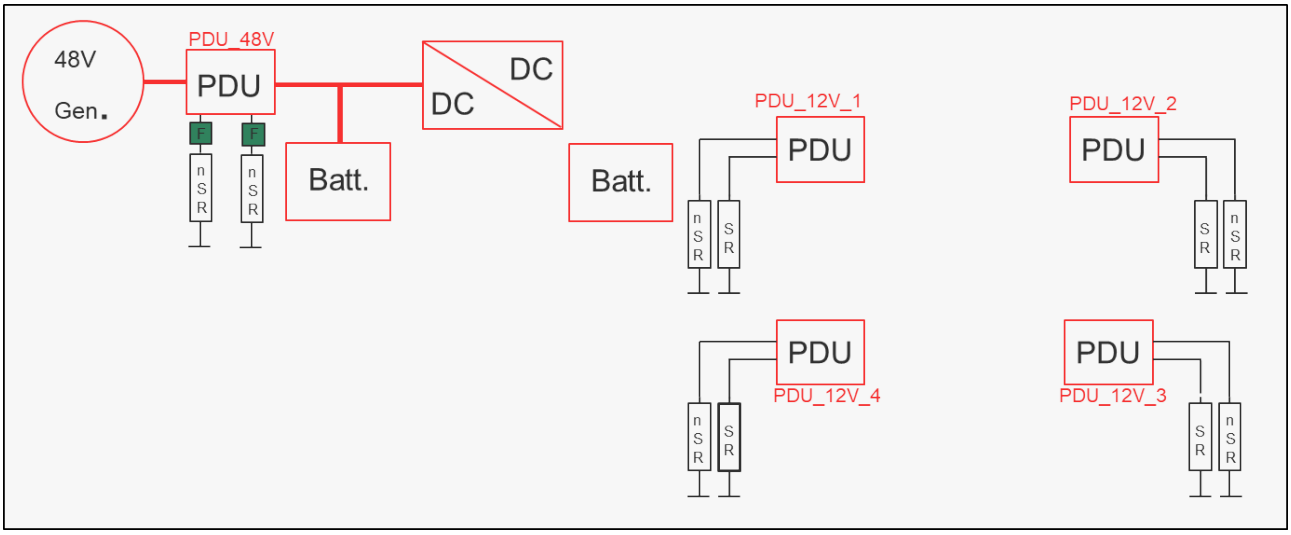


Abbildung 4: Vorgegebene Grundtopologie

```
{
  "topology": {
    "title": "Ak30 test topology",
    "nodes": [
      {
        "node": {
          "type": "Generator",
          "ID": "generator1"
        }
      },
      {
        "node": {
          "type": "Battery",
          "ID": "battery1",
          "parameters": {
            "OCV": 49,
            "number_p": 2,
            "number_s": 6
          },
          "_comment": "Batterie48V"
        }
      },
      . . .
    ],
    "sources": [
      "DCDC_Converter1.pin_p2",
      "battery2.p"
    ],
```

```
    "connections": [
      {
        "connection": {
          "type": "wire",
          "ID": "D1",
          "startnode": "generator1.p",
          "endnode": "pdu5.pin_p",
          "parameters": {
            "df_wire_cross_section": 2.5,
            "df_wire_length": 0.5
          }
        }
      },
      {
        "connection": {
          "type": "wire",
          "ID": "D2",
          "startnode": "DCDC_Converter1.pin_p1",
          "endnode": "pdu5.pin_p4",
          "parameters": {
            "df_wire_cross_section": 2.5,
            "df_wire_length": 0.5
          }
        }
      },
      . . .
    ],
    "configuration": [
      {
        "config": {
          "type": "wire_connect",
          "ID": "CFG1",
          "used-ID": "D1",
          "params": {
            "wire_cross_section": 0.5,
            "wire_length": 0.25
          }
        }
      },
      {
        "config": {
          "type": "wire_connect",
          "ID": "CFG2",
          "used-ID": "D2",
          "params": {
            "wire_cross_section": 0.5,
            "wire_length": 0.25
          }
        }
      },
      . . .
    ]
  }
}
```

1.4 KBL-Import von Grundtopologien in das JSON-Zwischenformat

Um in KBL-Dateien abgelegte Grundtopologien in der Simulation nutzen zu können, wurde ein KBL-Import in Python implementiert. Da bisher keine KBL-Dateien mit Grundtopologien

verfügbar waren, werden die bereits in der KBL-Datei enthaltenen Informationen zu Assemblies, Komponenten, Konnektoren, Wires, … ausgelesen und ausgegeben. Ein Umstieg auf Informationen zu Grundtopologien ist zu einem späteren Zeitpunkt mit nur geringem Auzfwand möglich. Nachfolgend ist ein kurzer Auszug einer KBL-Datei mit zirka 50000 Zeilen zu sehen.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <?elena version="2.8.0-b7" system="Windows 10 10.0" compilation_date="11.07.2018" compi-
    lation_time="17:12:00"?>
3.  <?extraction date="20.07.2018" time="07:53:45"?>
4.  <?pc checksum="695947884"?>
5.  <?xsd validated="true"?>
6.  <kbl:KBL_container xmlns:kbl="http://www.prostep.org/Car_electric_con-
    tainer/KBL2.3/KBLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    id="KBL_container" version_id="2.4 SR-1">
7.          <Assembly_part id="Assembly_part_1">
8.                  <Part_number>O12a714551a2</Part_number>
9.                  <Company_name>GCMC</Company_name>
10.                 <Version>16</Version>
11.                 <Abbreviation>/NULL</Abbreviation>
12.                 <Description>Kombi-Halter A01-11xC17-4.5xE01-14   </Description>
13.                 <Mass_information id="id_380_312">
14.                         <Unit_component>id_346_2</Unit_component>
15.                         <Value_component>5.0</Value_component>
16.                 </Mass_information>
17.                 <Fixing_occurrence id="Fixing_occurrence_1">
18.                         <Id>XXX_C17_001_A_1</Id>
19.                         <Part>Fixing_7</Part>
20.                 </Fixing_occurrence>
21.                 <Fixing_occurrence id="Fixing_occurrence_2">
22.                         <Id>XXX_E01_004_1</Id>
23.                         <Part>Fixing_6</Part>
24.                 </Fixing_occurrence>
25.                 <Fixing_occurrence id="Fixing_occurrence_3">
26.                         <Id>XXX_A01_006_A_1</Id>
27.                         <Part>Fixing_5</Part>
28.                 </Fixing_occurrence>
29.         </Assembly_part>
30.         . . .
31.         <Wire_protection id="Wire_protection_6">
32.                 <Part_number>O1112a319821a2</Part_number>
33.                 <Company_name>GCMC</Company_name>
34.                 <Version>3</Version>
35.                 <Abbreviation>/NULL</Abbreviation>
36.                 <Description>PP-Wellschlauch Gr. 13.0 D(i)=12.9 D(a)=15.8</Description>
37.                 <Mass_information id="id_380_289">
38.                         <Unit_component>id_346_4</Unit_component>
39.                         <Value_component>26.0</Value_component>
40.                 </Mass_information>
41.                 <Protection_type>WELLROHR</Protection_type>
42.         </Wire_protection>
43. </kbl:KBL_container>
```

Das Extrahieren der gewünschten Informationen benötigt weniger als 2 Sekunden auf einem Laptop. Von der Gesamtheit aller Datensätze sind hier beispielhaft einige wenige dargestellt, unter anderem zwei Parts, eine Komponente und ein mehradriges Kabel mit 2 Adern zu je 4mm² Querschnitt und 3,7mm Außendurchmesser.

```
1.  Assembly_part
2.                  id:  Assembly_part_1
3.         Part_number:  O12a714551a2
4.        Company_name:  GCMC
5.         Description:  Kombi-Halter A01-11xC17-4.5xE01-14
6.
7.   Assembly_part
8.                  id:  id_304_1
9.         Part_number:  5Ga108313641B
10.       Company_name:  GCMC
11.        Description:  VERBINDUNGSKABEL ABS-EPB-LWR-NIV
12. ...
13.
14. Component
15.                 id:  id_302_0
16.        Part_number:  4Da106213641C
17.       Company_name:  GCMC
18.        Description:  Arbeitskontaktrelais
19. ...
20.
21. General_wire
22.                 id:  id_327_96
23.        Part_number:  O102217971a2
24.       Company_name:  GCMC
25.        Description:  Mehradriges Kabel
26.          Wire_type:  multicore wire
27.            Core id:  id_317_20
28.   Cross_section_area:  4
29.     Outside_diameter:  3.7
30.            Core id:  id_317_21
31.   Cross_section_area:  4
32.     Outside_diameter:  3.7
```

1.5 JSON-Format zur Beschreibung von Fehlerfällen

Für die Definition der Fehlerbeschreibungen/-fälle wurde mit dem Projektpartner TU Dortmund ein JSON-Format festgelegt.

Die Parameter sind:

- library – Name der zu verwendenden Bibliothek zur Fehlersimulation
- model – Name des aktuell zu untersuchenden Modells (Nominalbeschreibung)
- monitorVoltageInstances – Bezeichnung der Instanzen von monitorVoltages im Modell (zur Darstellung deren Spannung während der Simulation)
- faultModelInstance – Defaultbezeichnung der injizierten Instanz von faultModel im Modell
- resultFileName – Name der Ergebnisdatei, welche vom Modell monitorVoltage geschrieben wird
- simulationParameters – an OpenModelica zu übergebende Simulationsparameter. startTime=0 und stoptime=1 sind default
- runs – die definierten Simulationsläufe/Untersuchungen

Jede Untersuchung hat:

- eine laufende Nummer run
- einen beschreibenden Text faultText
- den Namen des einzubauenden Fehlermodells faultModel
- Parameter für das Fehlermodell faultModelParameters
- faultModelInstance - der Instanzname, den das Fehlermodell erhalten soll (falls er von obigem Defaultnamen abweichen soll
- Wurde kein Fehlermodell faultModel angegeben, so wird nach dem Parameter faultModels geschaut, mit welchem mehrere Fehlermodelle (Mehrfachfehler) definiert werden können mit den jeweiligen Parametern
  - den Namen des einzubauenden Fehlermodells model
  - instance – der Instanzname, den das Fehlermodell erhalten soll
  - Parameter für das Fehlermodell parameters
- connections – eine Liste mit her zu stellenden Verbindungen im Nominalmodell, z.B. zwischen Fehlermodell und anderen Punkten des Modells
- disconnections – bei Bedarf eine Liste mit zu trennenden Verbindungen im Nominalmodell
- Parameter für weitere Modelle modelParameters mit jeweiligem Instanznamen instance und Parametern parameters

Folgend dargestellte Beschreibungsdatei soll kurz erläutert werden:

- Es ist die Bibliothek " TopologieVariants " zu laden
- Das Nominalmodell ist: " TopologieVariants.Examples.PDU_Test "
- Es sind live die Spannungen an der Monitorinstanz "monitorVoltage1" darzustellen
- Die Fehlermodelle werden mit Namen " actFault " instanziiert
- Die Ergebnisdatei wird als "summary_fault_simulation_*.csv" abgespeichert. Das „*“ wird für jedes Monitormodell von 00 beginnend hochgezählt
- Die Simulationsparameter für OpenModelica sind: "startTime=0", "stopTime=1", "numberOfIntervals": "300", "tolerance": "1.0e-5"

Unter runs sind 4 Läufe definiert:

- Run 1 trägt den Fehlertext „nominal“, es werden keinerlei Fehler injiziert oder Parameter geändert.
- Run 2 trägt den Fehlertext „DCDC_Converter - Pulse width too low“, es werden keinerlei Fehler injiziert, allerdings wird der Parameter "fault_type" der Instanz "DCDC_Converter" auf „2“ gesetzt.
- Run 3 trägt den Fehlertext „Wire Break“, es wird das Fehlermodell "OnBoardSystemFaultSimulation.BasicFaultModels.FaultCaseSerial" mit den Parametern "faultActivation=1" sowie "TFault=0.5" injiziert. Es bekommt von den Parametern weiter vorn den Instanznamen „faultInstance“. Die Verbindung von "resistor1.n" zu „resistor2.p“ wird aufgetrennt. Weiterhin werden zwei Verbindungen hergestellt, und zwar von "resistor1.n" zu "actFault.p" und von "actFault.n" zu „resistor2.p".

- Run 4 trägt den Fehlertext „Short“, es wird das Fehlermodell "OnBoardSystemFaultSimulation.BasicFaultModels.FaultCaseF1" mit den Parametern "faultActivation=1" sowie "TFault=0.5" injiziert. Es bekommt von den Parametern weiter vorn den Instanznamen „faultInstance“. Weiterhin werden zwei Verbindungen hergestellt, und zwar von "resistor2.n" zu "actFault.p" und von "actFault.n" zu „ground.p".

```
{
  "library": "TopologieVariants",
  "model": "TopologieVariants.Examples.PDU_Test",
  "monitorVoltageInstances": [
    "monitorVoltage1"
  ],
  "faultModelInstance": "actFault",
  "resultFileName": "summary_fault_simulation_*.csv",
  "resultfileIDs": [
    "00"
  ],
  "simulationParameters": {
    "startTime": "0.0",
    "stopTime": "1.0",
    "numberOfIntervals": "300",
    "tolerance": "1.0e-5"
  },
  "runs": [
    {
      "run": "1",
      "faultText": "nominal"
    },
    {
      "run": "2",
      "faultText": "DCDC_Converter - Pulse width too low",
      "modelParameters": [
        {
          "instance": "DCDC_Converter",
          "parameters": {
            "fault_type": "2"
          }
        }
      ]
    },
    {
      "run": "3",
      "faultText": "Wire Break",
      "faultModel": "OnBoardSystemFaultSimulation.BasicFaultModels.FaultCaseSerial",
      "faultModelParameters": {
        "faultActivation": "1",
        "TFault": "0.5"
      },
      "disConnections": [
        {
          "from": "resistor1.n",
          "to": "resistor2.p"
        }
      ],
      "connections": [
        {
          "from": "resistor1.n",
          "to": "actFault.p"
        },
```

```
        {
          "from": "resistor2.p",
          "to": "actFault.n"
        }
      ]
    },
    {
      "run": "4",
      "faultText": "Short",
      "faultModel": "OnBoardSystemFaultSimulation.BasicFaultModels.FaultCaseF1",
      "faultModelParameters": {
        "faultActivation": "1",
        "TFault": "0.5"
      },
      "connections": [
        {
          "from": "resistor2.n",
          "to": "actFault.p"
        },
        {
          "from": "ground.p",
          "to": "actFault.n"
        }
      ]
    }
  ]
}
```

2 Automatisierte Ableitung und Durchführung von Simulationsläufen

In folgender Abbildung dargestellter Workflow zur Generierung vom Modelicamodellen und Spicenetzlisten aus Datensätzen zu Grundtopologien und Topologievariaten wurde umgesetzt. Auf die einzelnen Schritte wird in den nächsten Abschnitten eingegangen.

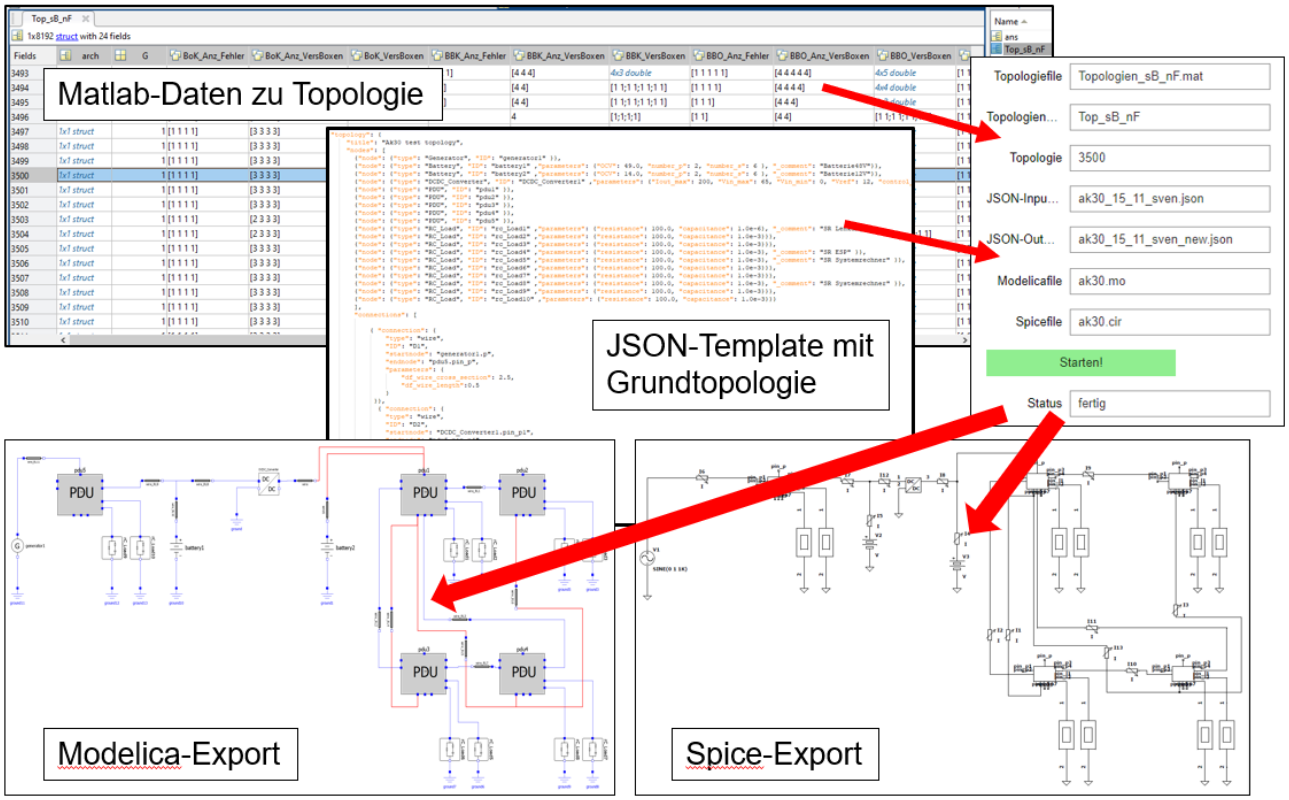
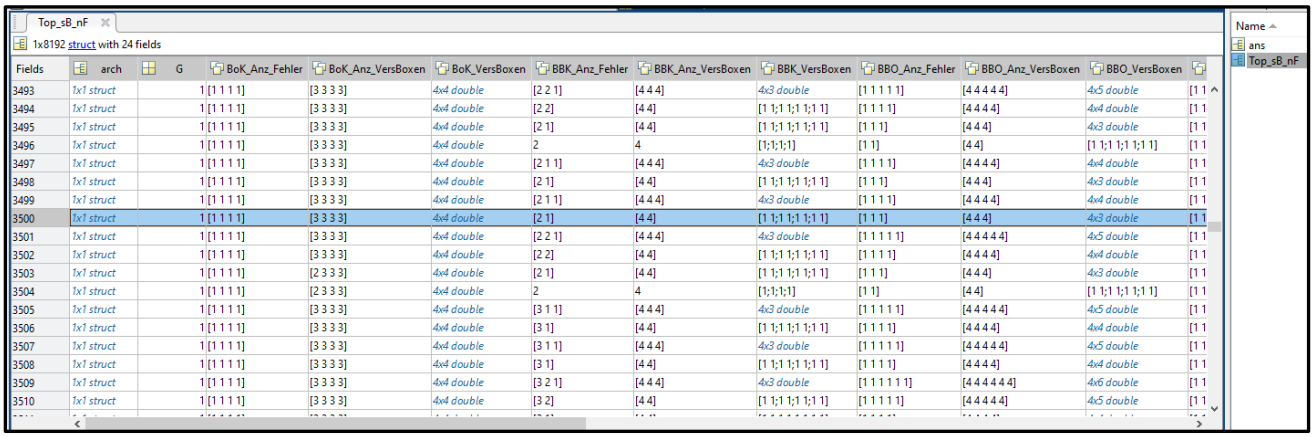


Abbildung 5: zu automatisierender Workflow

2.1 Import und Aufbereitung von Topologievarianten

Mögliche, zu untersuchende Topologievarianten werden durch den Projektpartner TU Kassel in Matlab definiert und sind somit als Matlab-Savefiles verfügbar. Folgendes Bild zeigt einen ausgewählten Bereich mit der Topologievariante 3500 von 8192 definierten Varianten in dieser Datei.



Top_sB_nF

1x8192 struct with 24 fields

Fields	arch	G	BoK_Anz_Fehler	BoK_Anz_VersBoxen	BoK_VersBoxen	BBK_Anz_Fehler	BBK_Anz_VersBoxen	BBK_VersBoxen	BBO_Anz_Fehler	BBO_Anz_VersBoxen	BBO_VersBoxen
3493	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 2 1]	[4 4 4]	4x3 double	[1 1 1 1 1]	[4 4 4 4 4]	4x5 double
3494	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 2]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1 1]	[4 4 4 4]	4x4 double
3495	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1]	[4 4 4]	4x3 double
3496	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	2	4	[1;1;1;1]	[1 1]	[4 4]	[1 1;1 1;1 1;1 1]
3497	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 1 1]	[4 4 4]	4x3 double	[1 1 1 1]	[4 4 4 4]	4x4 double
3498	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1]	[4 4 4]	4x3 double
3499	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 1 1]	[4 4 4]	4x3 double	[1 1 1 1]	[4 4 4 4]	4x4 double
3500	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1]	[4 4 4]	4x3 double
3501	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 2 1]	[4 4 4]	4x3 double	[1 1 1 1 1]	[4 4 4 4 4]	4x5 double
3502	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[2 2]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1 1]	[4 4 4 4]	4x4 double
3503	1x1 struct	1	[1 1 1 1]	[2 3 3 3]	4x4 double	[2 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1]	[4 4 4]	4x3 double
3504	1x1 struct	1	[1 1 1 1]	[2 3 3 3]	4x4 double	2	4	[1;1;1;1]	[1 1]	[4 4]	[1 1;1 1;1 1;1 1]
3505	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 1 1]	[4 4 4]	4x3 double	[1 1 1 1 1]	[4 4 4 4 4]	4x5 double
3506	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1 1]	[4 4 4 4]	4x4 double
3507	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 1 1]	[4 4 4]	4x3 double	[1 1 1 1 1]	[4 4 4 4 4]	4x5 double
3508	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 1]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1 1]	[4 4 4 4]	4x4 double
3509	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 2 1]	[4 4 4]	4x3 double	[1 1 1 1 1 1]	[4 4 4 4 4 4]	4x6 double
3510	1x1 struct	1	[1 1 1 1]	[3 3 3 3]	4x4 double	[3 2]	[4 4]	[1 1;1 1;1 1;1 1]	[1 1 1 1 1]	[4 4 4 4 4]	4x5 double

Name

ans

Top_sB_nF

Abbildung 6: Matlab-Savefile mit Topologievarianten

In Matlab lässt sich die Beschreibung dieser Topologievariante einfach ausgeben:

```
>> Top_sB_nF(3500)

ans =

  struct with fields:

                  arch: [1×1 struct]
                     G: 1
         BoK_Anz_Fehler: [1 1 1 1]
      BoK_Anz_VersBoxen: [3 3 3 3]
          BoK_VersBoxen: [4×4 double]
         BBK_Anz_Fehler: [2 1]
      BBK_Anz_VersBoxen: [4 4]
          BBK_VersBoxen: [4×2 double]
         BBO_Anz_Fehler: [1 1 1]
      BBO_Anz_VersBoxen: [4 4 4]
          BBO_VersBoxen: [4×3 double]
          LF_Anz_Fehler: [1 1 1 1]
       LF_Anz_VersBoxen: [4 4 4 4]
           LF_VersBoxen: [4×4 double]
          QF_Anz_Fehler: [1 1]
       QF_Anz_VersBoxen: [4 4]
           QF_VersBoxen: [4×2 double]
               Anz_esL: 0
               Anz_dsL: 4
                Anz_BB: 3
                Anz_sB: 4
                Kosten: 67
                Fehler: 4.0000e-09
         Fehler_per_Box: [1.0000e-09 1.0000e-09 1.0000e-09 1.0000e-09]

>> Top_sB_nF(3500).arch

ans =

  struct with fields:

     AnzQ: 2
     AnzB: 4
       PQ: {[1]  [1]}
      sVB: {[2 3 4]  [5]  [4]}
      BBB: {[3 4]  [4]  [5]}
      sVS: {[2 2 2]  [5]  [2]}
```

```
44.       sB: [1 1 1 1]
```

Die Bedeutung der einzelnen Bezeichner und deren Inhalte wurden durch die TU Kassel festgelegt und dokumentiert. Auf sie soll hier nicht weiter eingegangen werden. Diese Daten werden direkt bei der Modellgenerierung aus dem Matlab-Savefile gelesen und nicht in ein JSON-Zwischenfile abgelegt. So kann einfach auf jede beliebige Topologievariante in dieser Datei durch Vorgabe der Topologie-ID zugegriffen werden. Der Pythoncode zur Modellgenerierung wurde um entsprechende Importfunktionen erweitert. Da die Matlab-Savefiles binäre Dateien sind, wurde in Python auf die scipy-Bibliothek zurückgegriffen, welche entsprechende Funktionalitäten zur Datenextraktion bietet. Die für die ausgewählte Topologievariante 3500 extrahierten Daten sind nachfolgend aufgeführt. Dies sind unter anderem die Anzahl der mit den PDUs verschalteten Quellen, die Anzahl der PDUs, quellen-, senken- oder zweiseitig schaltbare sowie Backbone-Verbindungen und ebenso die Ausfallwahrscheinlichkeiten der einzelnen PDUs.

```
1.  Topologie ID: 3500
2.  Anzahl Quellen: 2
3.  Anzahl PDU: 4
4.  Quelle 1 verbunden mit PDU 1
5.  Quelle 2 verbunden mit PDU 1
6.  PDU 1 verbunden mit PDU 2 zweiseitig schaltbar
7.  PDU 1 verbunden mit PDU 3 zweiseitig schaltbar
8.  PDU 1 verbunden mit PDU 4 zweiseitig schaltbar
9.  PDU 3 verbunden mit PDU 4 zweiseitig schaltbar
10. PDU 1 verbunden mit PDU 3 backbone
11. PDU 1 verbunden mit PDU 4 backbone
12. PDU 2 verbunden mit PDU 4 backbone
13. PDU 1 Ausfallwahrscheinlichkeit 1e-09
14. PDU 2 Ausfallwahrscheinlichkeit 1e-09
15. PDU 3 Ausfallwahrscheinlichkeit 1e-09
16. PDU 4 Ausfallwahrscheinlichkeit 1e-09
```

Die bereits zuvor beschriebende JSON-Datei mit der Grundtopologie wird dabei zuerst eingelesen und die Daten im Speicher gehalten. Die extrahierten Daten der ausgewählten Topologievariante ergänzen nun diese Daten. Ausgewählte Datensätze zeigt der nachfolgende Code. Es wird die Verbindung zwischen Quelle 1 und PDU1 über einen Wire geschaffen. Zwischen PDU1 und PDU2 wird eine zweiseitig schaltbare Verbindung angelegt und PDU1 und PDU3 sind über eine BackBode verbunden.

```
1.  {
2.          "connection": {
3.                  "type": "wire",
4.                  "ID": "TD1",
5.                  "startnode": "DCDC_Converter1.pin_p2",
6.                  "endnode": "pdu1.pin_p",
7.                  "parameters": {
8.                          "df_wire_cross_section": 2.5,
9.                          "df_wire_length": 0.5
10.                 }
11.         }
12. },
```

```
...
{
        "connection": {
                "type": "switch_two_conn",
                "ID": "SW2C1",
                "startnode": "pdu1.pin_p2",
                "endnode": "pdu2.pin_p2",
                "parameters": {
                        "df_wire_cross_section": 2.5,
                        "df_wire_length": 0.5,
                        "Fuse": 20.0
                }
        }
},
...
{
        "connection": {
                "type": "backbone_conn",
                "ID": "B1",
                "startnode": "pdu1.pin_p5",
                "endnode": "pdu3.pin_p5",
                "parameters": {
                        "Fuse": 20.0
                }
        }
},
...
```

2.2 Modelica-Modell einer Power-Distribution-Unit (PDU)

Die Modelica-Bibliotek zur Fehlersimulation wurde in den Vorgängerprojekten immer weiter entwickelt. Für die aktuellen Untersuchungen fehlte jedoch noch das Bibliotheksmodell für die PDU. Folgende Abbildungen zeigen das Icon und den Schaltplan der PDU in OpenModelica. Die Anschlüsse p1, p3 und p7 sind nichtschaltbare Verbindungen, p2, p4 und p6 sind schaltbare Verbindungen, p5 ist der BackBone-Anschluss und l1 und l2 sind Lastanschlüsse. Anschluss p ist für die Quelle. In einer ersten Variante sind sämtliche Schalter geschlossen, die Steuerleitungen sind noch nach außen zu führen und an eine Steuerlogik anzuschließen.

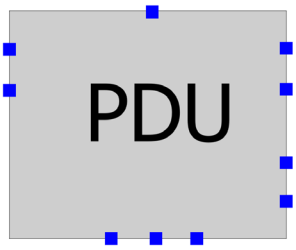


Abbildung 7: Icon einer PDU in Openmodelica

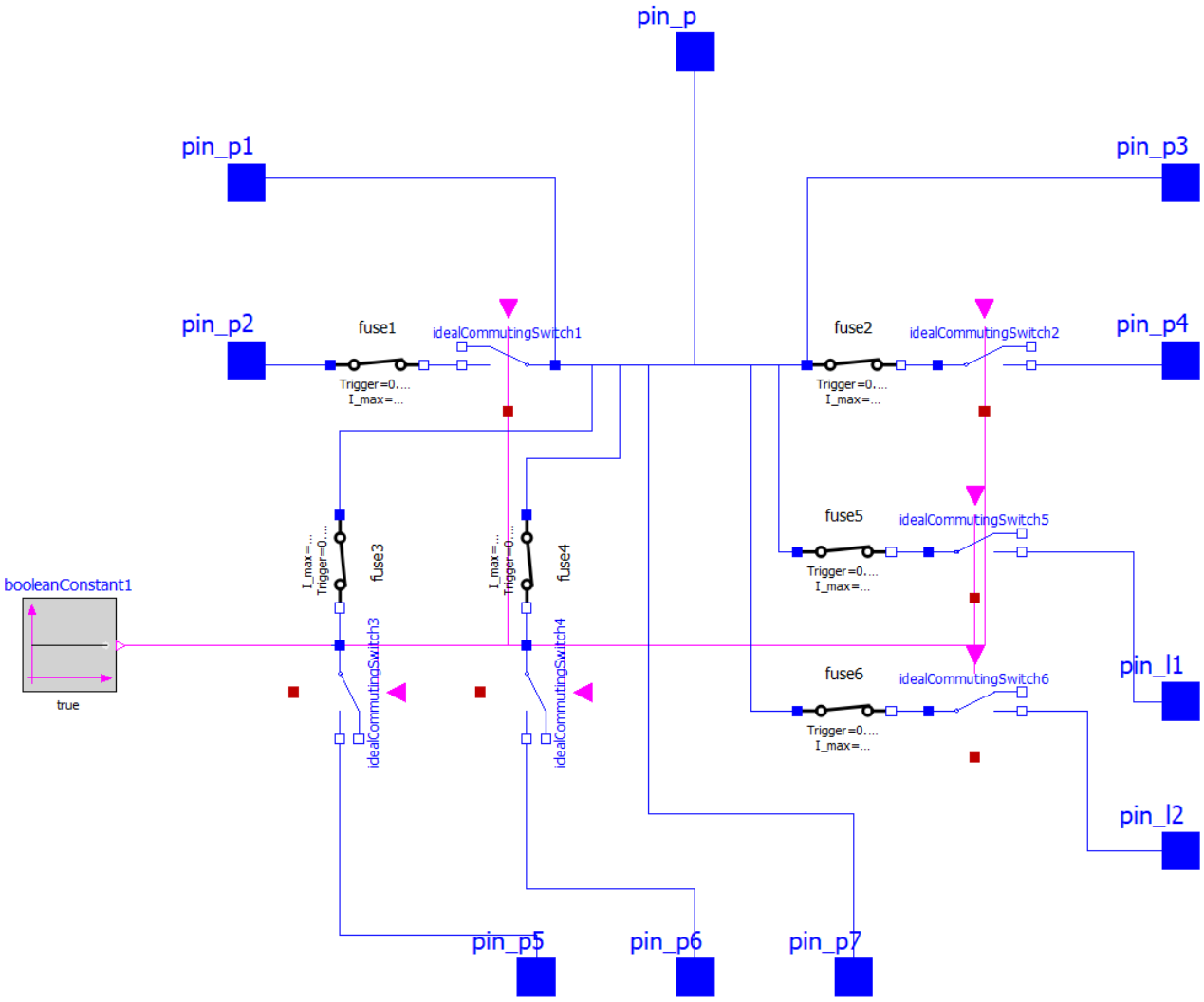


Abbildung 8: Schaltplan einer PDU in Openmodelica

2.3 Modellgenerierung aus Grundtopologie und ausgewählter Topologievariante

Es wurde Pythoncode entwickelt, welcher sowhl die JSON-Datei mit der Beschreibung der Grundtopologie sowie das Matlab-Savefile mit der Vielzahl an Topologievarianten einliest, die Daten zu einer Gesamttopologie zusammenführt und schließlich Modelicamodelle und Spicenetzlisten daraus generiert.

Zur einfacheren Bedienung wurde der Pythoncode um eine simple grafische Nutzeroberfläche ergänzt. Mögliche Eingabemöglichkeiten sind hierbei das Topologiefile (das Matlab-Savefile), der Name der Topologie in diesem Matlabcode, die zu untersuchende Topologie und das JSON-File mit der Grundtopologie. Zu Debugzwecken kann die Gesamttopologie im JSON-Format (Outputfile) gespeichert werden, sie wird an sich nur im Speicher vorgehalten. Weiterhin sind die gewünschten Dateinamen des Modelicamodells sowie der Spicenetzliste anzugeben. Mit dem Button „Starten!“ wird die Generierung gestartet, der Status wechelt nach weniger als 2 Sekunden auf „fertig“ und die beiden Modelle bzw. Netzlisten sind bereit für die Simulation.

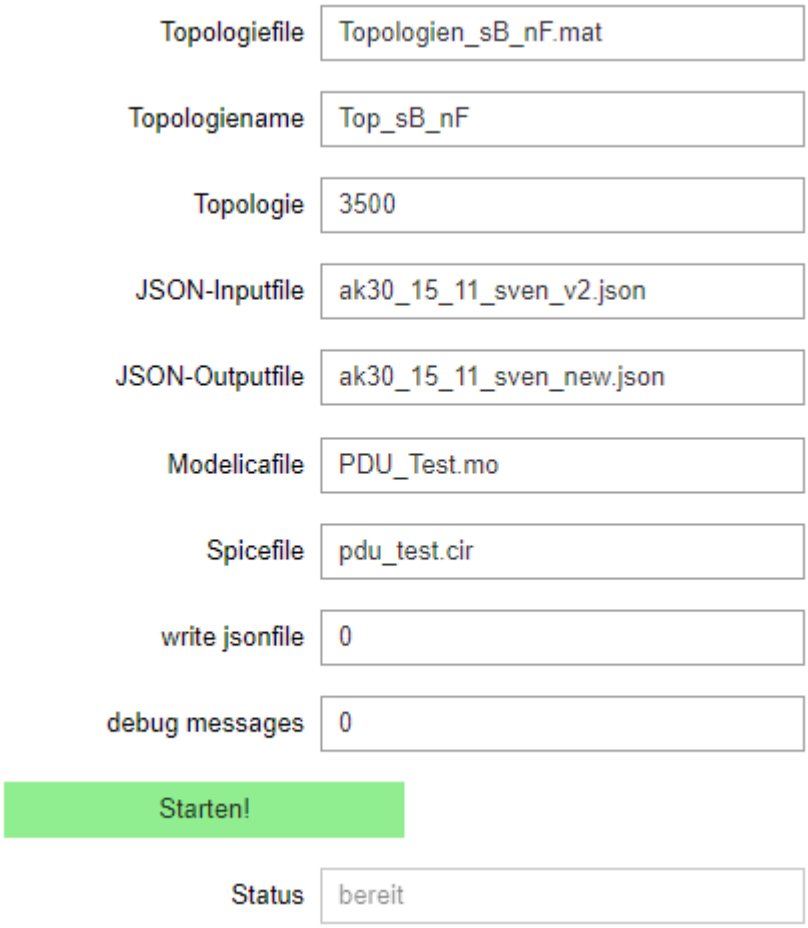


Abbildung 9: GUI zur Modellgenerierung aus Grundtopologie und ausgewählter Topologievariante

Der folgende Code zeigt das generierte Modelicamodell. Zur Veranschaulichung wurde es in OpenModelica händisch nachgebaut, da der generierte Modellcode naturgemäß keine grafische Darstellung enthält.

```
1.  model PDU_Test
2.  TopologieVariants.ComponentFaultModels.Generator generator1;
3.  TopologieVariants.ComponentFaultModels.Battery battery1(OCV = 49.0,number_p = 2,number_s
    = 6);
4.  TopologieVariants.ComponentFaultModels.Battery battery2(OCV = 14.0,number_p = 2,number_s
    = 6);
5.  TopologieVariants.ComponentFaultModels.DCDC_Converter DCDC_Converter1(Iout_max =
    200,Vin_max = 65,Vin_min = 0,Vref = 12,control_p = 50,fault_type = 0);
6.  TopologieVariants.ComponentFaultModels.PDU pdu1;
7.  TopologieVariants.ComponentFaultModels.PDU pdu2;
8.  TopologieVariants.ComponentFaultModels.PDU pdu3;
9.  TopologieVariants.ComponentFaultModels.PDU pdu4;
10. TopologieVariants.ComponentFaultModels.PDU pdu5;
11. TopologieVariants.LoadModels.RC_Load rc_Load1(resistance = 100.0,capacitance = 1e-06);
12. TopologieVariants.LoadModels.RC_Load rc_Load2(resistance = 100.0,capacitance = 0.001);
13. TopologieVariants.LoadModels.RC_Load rc_Load3(resistance = 100.0,capacitance = 0.001);
14. TopologieVariants.LoadModels.RC_Load rc_Load4(resistance = 100.0,capacitance = 0.001);
15. TopologieVariants.LoadModels.RC_Load rc_Load5(resistance = 100.0,capacitance = 0.001);
16. TopologieVariants.LoadModels.RC_Load rc_Load6(resistance = 100.0,capacitance = 0.001);
17. TopologieVariants.LoadModels.RC_Load rc_Load7(resistance = 100.0,capacitance = 0.001);
18. TopologieVariants.LoadModels.RC_Load rc_Load8(resistance = 100.0,capacitance = 0.001);
```

```
19. TopologieVariants.LoadModels.RC_Load rc_Load9(resistance = 100.0,capacitance = 0.001);
20. TopologieVariants.LoadModels.RC_Load rc_Load10(resistance = 100.0,capacitance = 0.001);
21. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D1(df_wire_cross_section =
    2.5,df_wire_length = 0.5);
22. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D2(df_wire_cross_section =
    2.5,df_wire_length = 0.5);
23. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D3(df_wire_cross_section =
    2.5,df_wire_length = 0.5);
24. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D5(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
25. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D6(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
26. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D7(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
27. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D8(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
28. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D9(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
29. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D10(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
30. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D11(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
31. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D12(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
32. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D13(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
33. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_D14(df_wire_cross_section =
    0.5,df_wire_length = 0.25);
34. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_TD1(df_wire_cross_section =
    2.5,df_wire_length = 0.5);
35. TopologieVariants.ComponentFunctionalModels.Wire_RL wire_TD2(df_wire_cross_section =
    2.5,df_wire_length = 0.5);
36. equation
37. connect(generator1.p, wire_D1.p);
38. connect(wire_D1.n, pdu5.pin_p);
39. connect(DCDC_Converter1.pin_p1, wire_D2.p);
40. connect(wire_D2.n, pdu5.pin_p4);
41. connect(battery1.p, wire_D3.p);
42. connect(wire_D3.n, pdu5.pin_p4);
43. connect(pdu5.pin_l1, wire_D5.p);
44. connect(wire_D5.n, rc_Load1.p);
45. connect(pdu5.pin_l2, wire_D6.p);
46. connect(wire_D6.n, rc_Load2.p);
47. connect(pdu1.pin_l1, wire_D7.p);
48. connect(wire_D7.n, rc_Load3.p);
49. connect(pdu1.pin_l2, wire_D8.p);
50. connect(wire_D8.n, rc_Load4.p);
51. connect(pdu2.pin_l1, wire_D9.p);
52. connect(wire_D9.n, rc_Load5.p);
53. connect(pdu2.pin_l2, wire_D10.p);
54. connect(wire_D10.n, rc_Load6.p);
55. connect(pdu3.pin_l1, wire_D11.p);
56. connect(wire_D11.n, rc_Load7.p);
57. connect(pdu3.pin_l2, wire_D12.p);
58. connect(wire_D12.n, rc_Load8.p);
59. connect(pdu4.pin_l1, wire_D13.p);
60. connect(wire_D13.n, rc_Load9.p);
61. connect(pdu4.pin_l2, wire_D14.p);
62. connect(wire_D14.n, rc_Load10.p);
63. connect(DCDC_Converter1.pin_p2, wire_TD1.p);
64. connect(wire_TD1.n, pdu1.pin_p);
65. connect(battery2.p, wire_TD2.p);
66. connect(wire_TD2.n, pdu1.pin_p);
67. connect(pdu1.pin_p2, pdu2.pin_p2);
68. connect(pdu1.pin_p4, pdu3.pin_p2);
```

```
connect(pdu1.pin_p6, pdu4.pin_p2);
connect(pdu3.pin_p4, pdu4.pin_p4);
connect(pdu1.pin_p5, pdu3.pin_p5);
connect(pdu1.pin_p5, pdu4.pin_p5);
connect(pdu2.pin_p5, pdu4.pin_p5);
end PDU_Test;
```

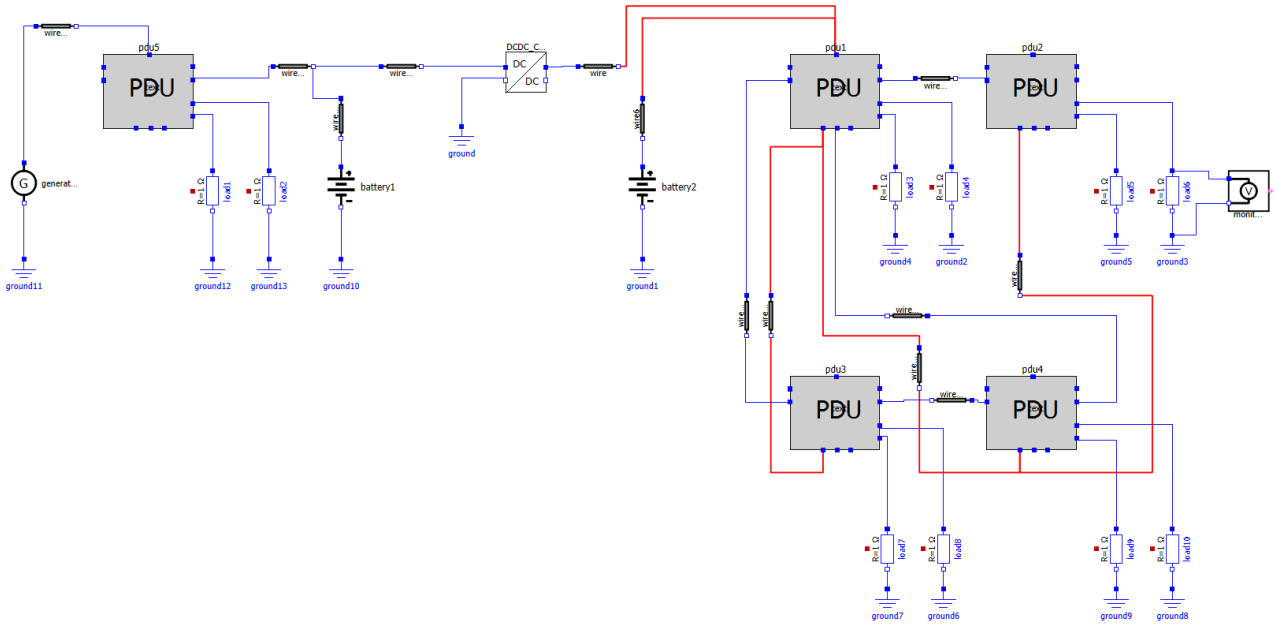


Abbildung 10: Modelicamodell der Gesamttopologie

Bei der Spicenetzliste verhält es sich analog. Der folgende Code zeigt die generierte Netzliste, zur Veranschaulichung wurde es ebenso in einer grafischen Umgebung händisch nachgebaut.

```
GENERATOR1 N1 0 GENERATOR
BATTERY1 N5 0 BATTERY OCV=49.0 number_p=2 number_s=6
BATTERY2 N29 0 BATTERY OCV=14.0 number_p=2 number_s=6
DCDC_CONVERTER1 N3 0 N27 0 DCDC_CONVERTER Iout_max=200 Vin_max=65 Vin_min=0 Vref=12 control_p=50 fault_type=0
PDU1 N30 NC1 N31 NC2 N33 N41 N35 NC3 N11 N13 Netlist_pdu.lib pdu_mod
PDU2 NC4 NC5 N32 NC6 NC7 N43 NC8 NC9 N15 N17 Netlist_pdu.lib pdu_mod
PDU3 NC10 NC11 N34 NC12 N37 N40 NC13 NC14 N19 N21 Netlist_pdu.lib pdu_mod
PDU4 NC15 NC16 N36 NC17 N38 N44 NC18 NC19 N23 N25 Netlist_pdu.lib pdu_mod
PDU5 N2 NC20 NC21 NC22 N6 NC23 NC24 NC25 N7 N9 Netlist_pdu.lib pdu_mod
RC_LOAD1 N8 0 RC_LOAD resistance=100.0 capacitance=1e-06
RC_LOAD2 N10 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD3 N12 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD4 N14 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD5 N16 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD6 N18 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD7 N20 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD8 N22 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD9 N24 0 RC_LOAD resistance=100.0 capacitance=0.001
RC_LOAD10 N26 0 RC_LOAD resistance=100.0 capacitance=0.001
WIRE_RL1 N1 N2 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5
```

```
WIRE_RL2 N3 N4 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5
WIRE_RL3 N5 N6 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5
WIRE_RL4 N7 N8 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL5 N9 N10 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL6 N11 N12 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL7 N13 N14 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL8 N15 N16 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL9 N17 N18 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL10 N19 N20 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL11 N21 N22 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL12 N23 N24 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL13 N25 N26 WIRE_RL df_wire_cross_section=0.5 df_wire_length=0.25
WIRE_RL14 N27 N28 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5
WIRE_RL15 N29 N30 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5
WIRE_RL16 N31 N32 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5 Fuse=20.0
WIRE_RL17 N33 N34 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5 Fuse=20.0
WIRE_RL18 N35 N36 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5 Fuse=20.0
WIRE_RL19 N37 N38 WIRE_RL df_wire_cross_section=2.5 df_wire_length=0.5 Fuse=20.0
WIRE_RL20 N39 N40 WIRE_RL Fuse=20.0
WIRE_RL21 N41 N42 WIRE_RL Fuse=20.0
WIRE_RL22 N43 N44 WIRE_RL Fuse=20.0
```

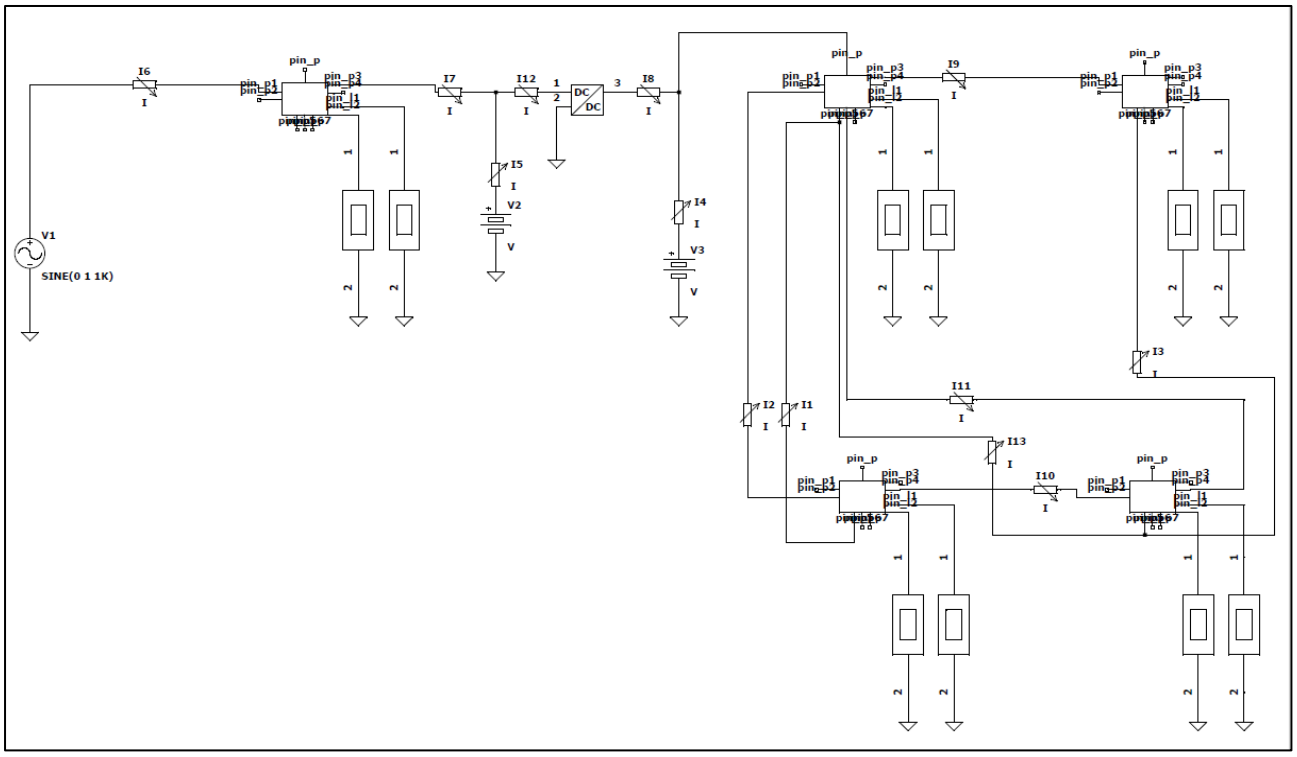


Abbildung 11: Spicemodell der Gesamttopologie

2.4 Automatisierte Simulationsabläufe mit OpenModelica

Ausgangspunkt ist hierbei der im vorherigen Projekt entwickelte Pythoncode zur automatisierten Simulation. Es wurde dabei das Python-Interface von OpenModelica genutzt. Hierzu ist parallel zu OpenModelica Python zu installieren. Im Projekt wurde Anaconda Python gewählt mit den in dieser Distribution enthaltenen juyper-Notebooks [6]. Diese ermöglichen eine elegante Darstellung des Pythoncodes und dessen Ausgaben/Visualisierungen in Form von

Webseiten. Das Modul der Pythonschnittstelle ist in der OpenModelica-Distribution enthalten, muss aber noch installiert und eingerichtet werden. Dies ist detailliert im Anhang beschrieben.

Kurz dargestellt sind dann drei Befehle in python notwendig:

1. Import der Schnittstelle zu OpenModelica (einmalig)
   ```
   from OMPython import OMCSessionZMQ
   ```
2. Initiieren einer Session mit OpenModelica (einmalig)
   ```
   omcs=OMCSessionZMQ()
   ```
3. Kommunikation mit OpenModelica / Senden von Befehlen (für jeden Befehl)
   ```
   omcs.sendExpression("getVersion()")
   ```

In diesem Fall wird die Version des aktuell installierten OpenModelicas abgefragt und ausgegeben. Dies dient als Test der erfolgreichen Installation der Schnittstelle.

Python ist sehr mächtig, es gibt eine Vielzahl an Bibliotheken, beliebige Dateien können so gelesen und geparsed werden und es ist ebenso sehr gut für Auswertungen und verteiltes/paralleles Rechnen geeignet.

Im Pythoncode werden zum einen das Modelicamodell der Gesamttopologie (Beschreibung der Simulationsaufgabe für den Nominalfall) und zum anderen die Fehlerbeschreibungen vom Projektpartner im weiter vorn vorgestellen JSON-Format gelesen. Mit diesen Informationen werden die Fehler automatisch in das Nominalmodell der Gesamttopologie des Bordnetzes injiziert, die Simulationsläufe automatisiert getstartet, auf deren Fertigstellung gewartet, live gewünschte Kurvenverläufe dargestellt und die Ergebnisse am Ende visualiert. Wird angegeben, dass verteiltes Rechnen (parallele Simulationen) gewünscht ist, um die Gesamtrechenzeit zu minimieren, so werden entsprechend viele Simulationen parallel gestartet. Die Ergebnisse werden als CSV-Dateien (z.B. mit Excel zu öffnen) zentral gesammelt abgelegt. Abbildung 12 zeigt den entsprechenden Flow.

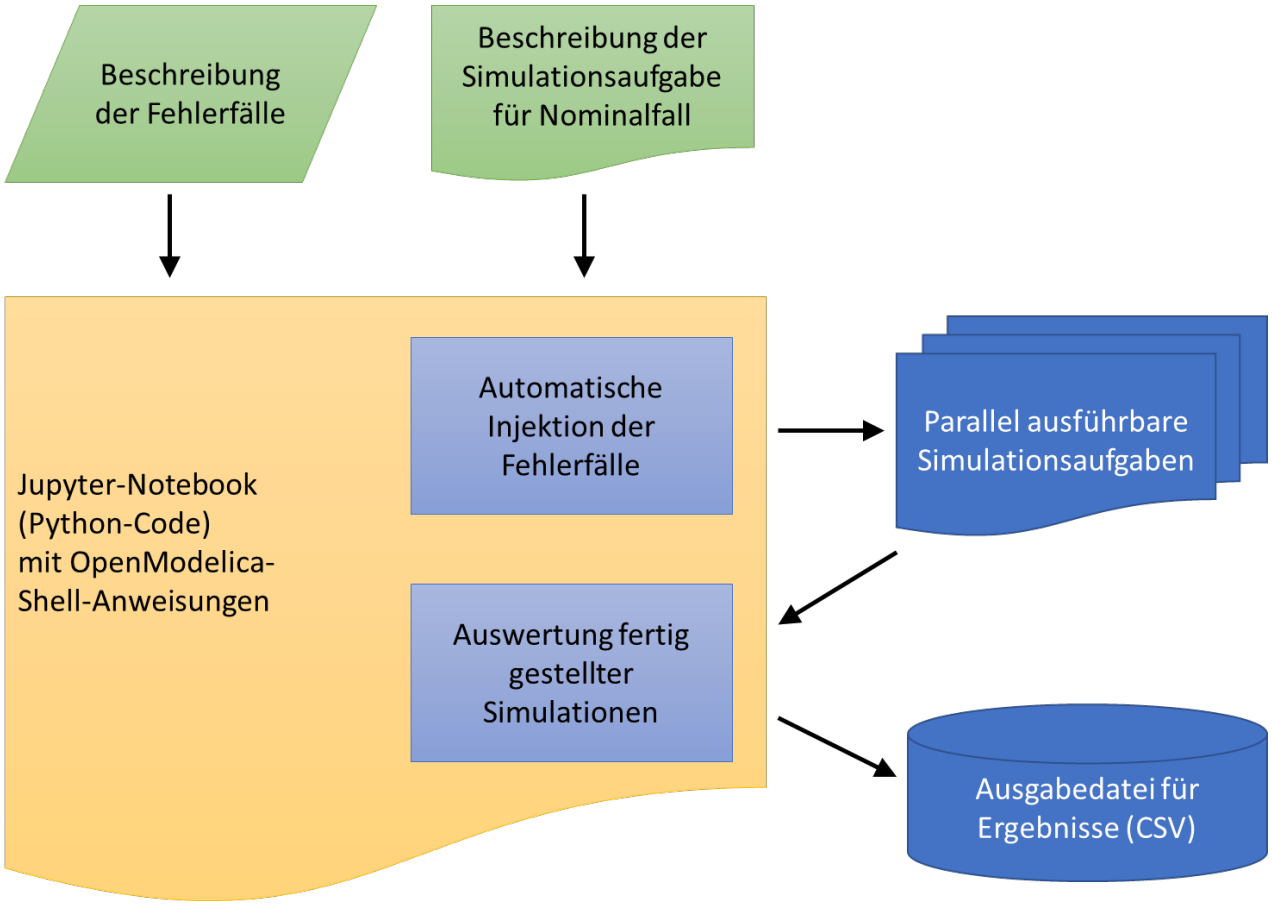
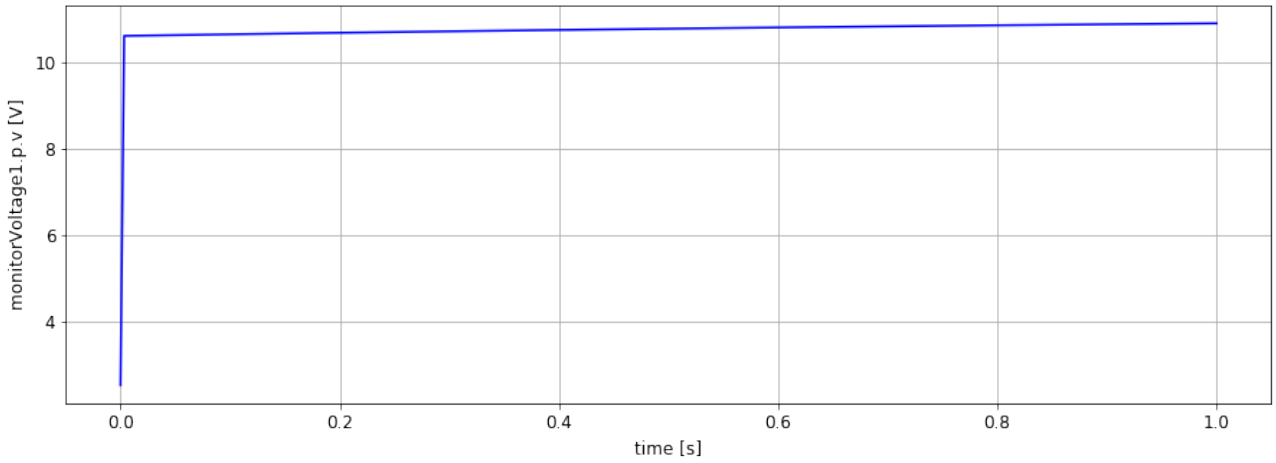


Abbildung 12: Automatisierte Fehlersimulation mit Python

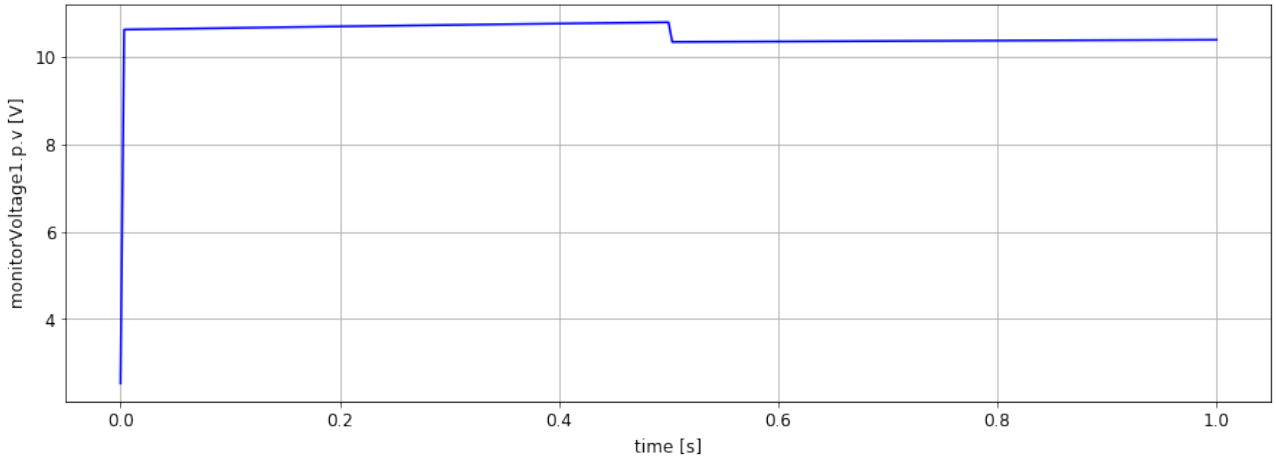
Das bei der Modellgenerierung der Gesamttopologie erzeugte Modelicamodell PDU_Test.mo wurde zum Test der aus dem Vorgängerprojekt übernommenen und aktualisierten sowie angepassten Simulationsumgebung mit den Fehlervarianten „fehlerfrei“ und „DCDC_Converter - Pulse width too low“ beaufschlagt. Die automatisierte Simulation verlief fehlerfrei und zeigte nachfolgendend dargestellten Output. Dabei wurde die Spannung an einer Last der PDU 2 aufgezeichnet.

Liveausgabe im Jupyter-Notebook während der Simulationsläufe:

```
runs: 2
run: 1 von 2
   actRun: 1
   actFaultText: nominal
   actFaultModels: []
   actConnections: []
   actDisConnections: []
   actModelParameters: []
   Simulation ended with SUCCESS.
   Seconds elapsed: 26.276962995529175
```




```
run: 2 von 2
   actRun: 2
   actFaultText: DCDC_Converter - Pulse width too low
   actFaultModels: []
   actConnections: []
   actDisConnections: []
   actModelParameters: [['DCDC_Converter', 'fault_type', '2']]
   Simulation ended with SUCCESS.
   Seconds elapsed: 27.074363708496094
```




```
Entire seconds elapsed: 53.74647855758667
```

Der nächste Schritt ist die Bereitstellung der Fehlervarianten durch die Projektpartner im abgestimmten JSON-Format und deren Anwendung auf ausgewählte Topologievarianten.

3 Zusammenfassung

Im Rahmen des hier dargestellten Teilprojekts wurden Datenaustauschformate für Kabelbäume, wie KBL und VEC, untersucht. In Zusammenarbeit mit den Projektpartnern wurde ein Zwischenformat für Grundtopologien für Bordnetze in JSON abgestimmt. Pythonroutinen wurden entwickelt, um entsprechende Einträge von KBL-Dateien zu parsen und in diesem Zwischenformat abzulegen.

Die durch Projektpartner in Matlab definierten Topologievarianten werden nun automatisch eingelesen und die Anzahl der PDUs und deren verschiedenen Verschaltungsarten berücksichtigt, vorgegebene Verbindungen mit Quellen (z.B. Batterien oder DCDC-Converter) hergestellt und Ausfallwahrscheinlichkeiten übernommen. Die in den Vorgängerprojekten stetig erweiterten Modelicabibliotheken wurden um ein Modell einer PDU ergänzt.

Aus gegebenen Grundtopologien von Bordnetzen und den Topologievarianten können über eine grafische Nutzeroberfläche direkt Modelicamodelle und Spicenetzlisten der Gesamttopologie genriert werden. Es konnte auf die Routinen zur automatisierten Ableitung und Durchführung von Simulationläufen aus dem Vorgängerprojekt zurückgegriffen werden, welche adaptiert und weiterentwickelt wurden. Mit den Projektpartnern wurde ebenfalls ein JSON-Format zur Beschreibung von Fehlerfällen im Bordnetz abgestimmt. So können die Fehlerfälle automatisiert eingelesen und die Fehler einzeln in das Modell der Gesamttopologie injiziert werden. Die Simulationsläufe werden gestartet, auf deren Fertigstellung gewartet, live gewünschte Kurvenverläufe dargestellt und die Ergebnisse am Ende visualisiert.

Wird angegeben, dass verteiltes Rechnen (parallele Simulationen) gewünscht ist, um die Gesamtrechenzeit zu minimieren, so werden entsprechend viele Simulationen parallel gestartet. Die Ergebnisse werden als CSV-Dateien (z.B. mit Excel zu öffnen) zentral gesammelt abgelegt.

4 Literatur

[1] Warum wir den VEC-Standard brauchen
Online: https://www.elektroniknet.de/automotive/software-tools/warum-wir-den-vec-standard-brauchen.123839.html

[1] KBL-Formatbeschreibung
Online: https://ecad-wiki.prostep.org/specifications/kbl/

[2] VEC-Formatbeschreibung
Online: https://ecad-wiki.prostep.org/specifications/vec/

[3] JSON-Formatbeschreibung
Online: https://www.json.org/json-en.html

[4] OpenModelica Open Source Project
Online: https://openmodelica.org/

[5] Jupyter Open Source Project
Online: https://jupyter.org/

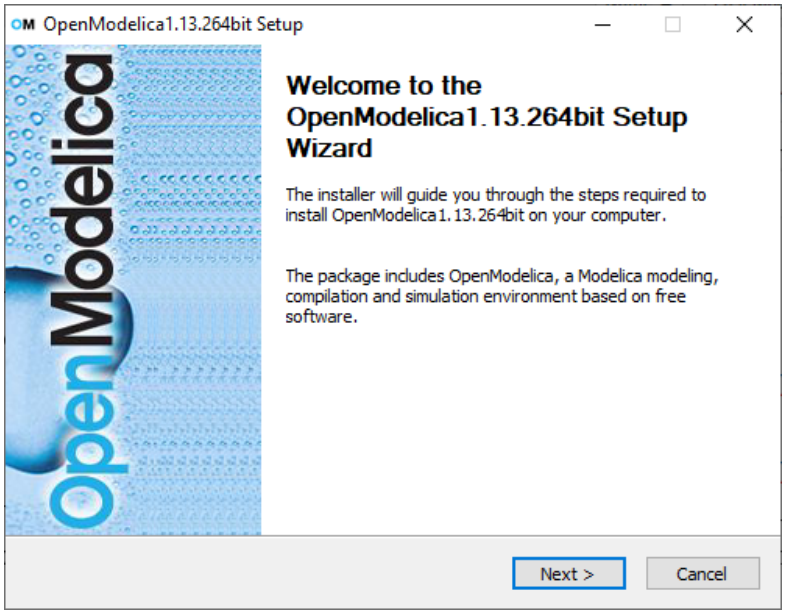
5 Anhang

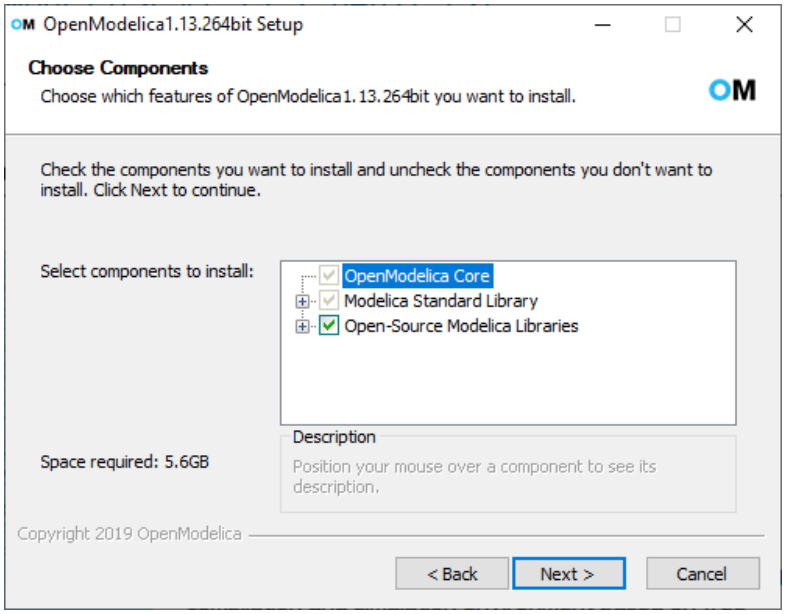
5.1 Installation OpenModelica, Anaconda Python und der Schnittstelle OMPython

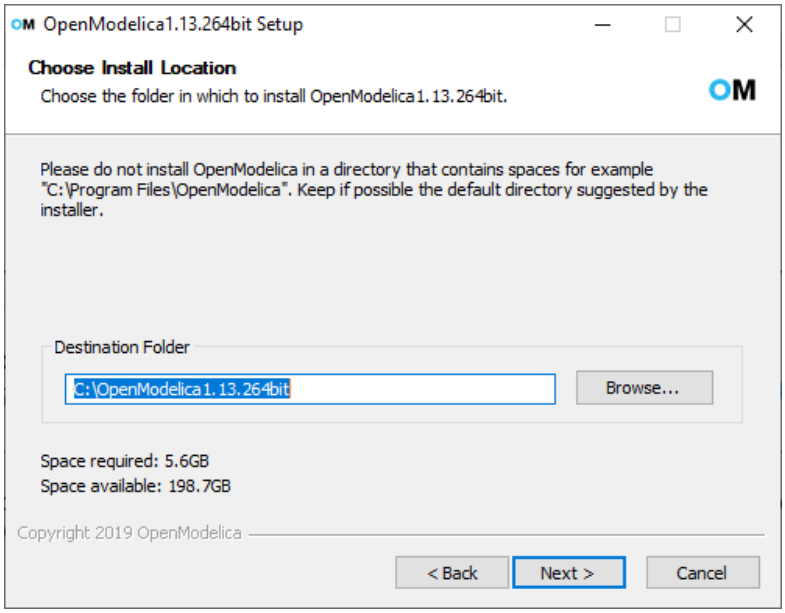
Hinweis: Bei anderen/aktuelleren Versionen beider Tools entsprechende Downloadadressen nutzen und den Pfad in Punkt 6 anpassen! Die Ausgabe in Punkt 15 ist dann auch entsprechend der aktuellen OpenModelica-Version. Wird eine 32bit-Version eines Tools genutzt, so ist auch das andere Tool in 32bit zu installieren!

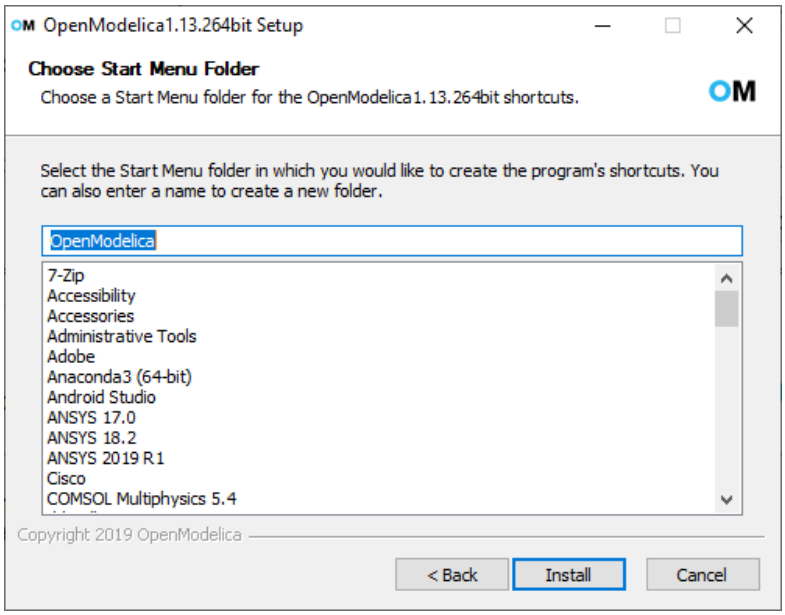
Aktuell sind OpenModelica 1.18.1 sowie Anaconda 2022.5.

1. OpenModelica 1.13.2 64bit
   Download von: `https://build.openmodelica.org/omc/builds/windows/releases/1.13/2/64bit/OpenModelica-v1.13.2-64bit.exe`
2. Anaconda 2019.3 64bit
   Download von: `https://repo.anaconda.com/archive/Anaconda3-2019.03-Windows-x86_64.exe`
3. Installation von OpenModelica mit Defaulteinstellungen

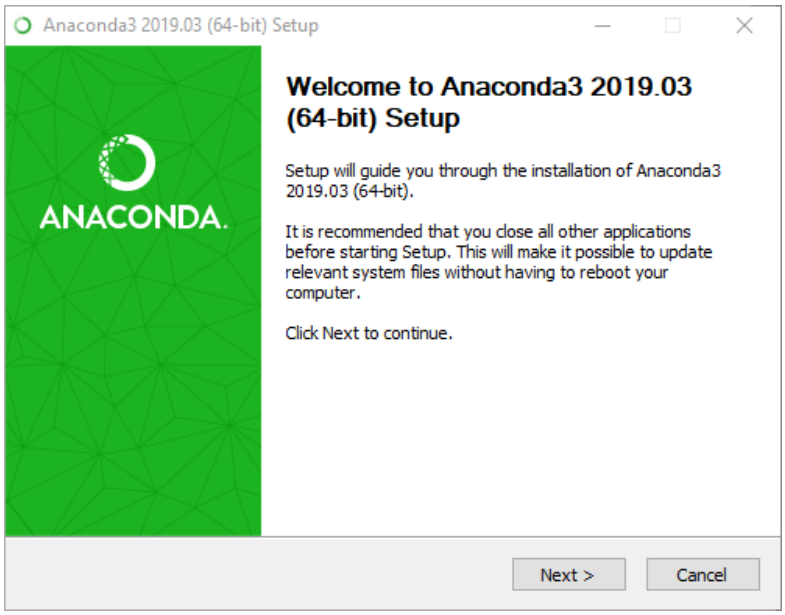


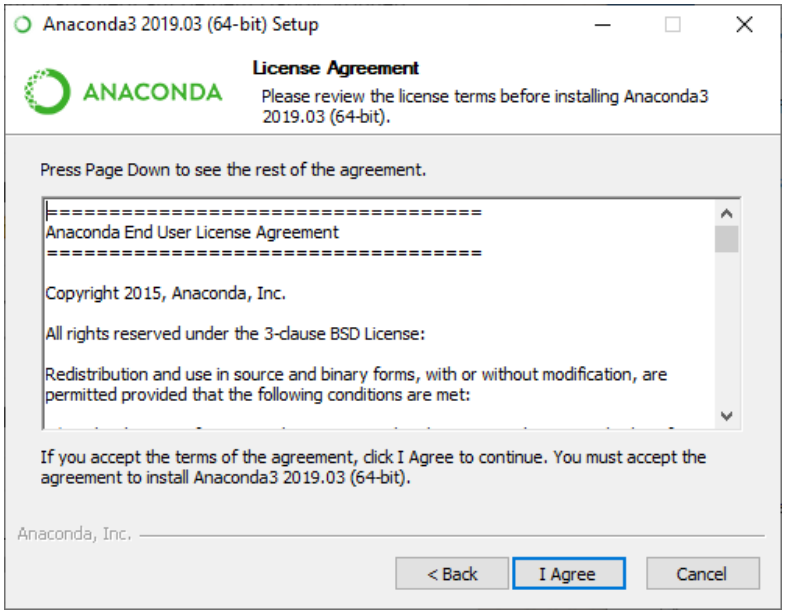

OpenModelica1.13.264bit Setup
Choose Components
Choose which features of OpenModelica1.13.264bit you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
OpenModelica Core
Modelica Standard Library
Open-Source Modelica Libraries
Description
Position your mouse over a component to see its description.
Space required: 5.6GB
Copyright 2019 OpenModelica
< Back
Next >
Cancel

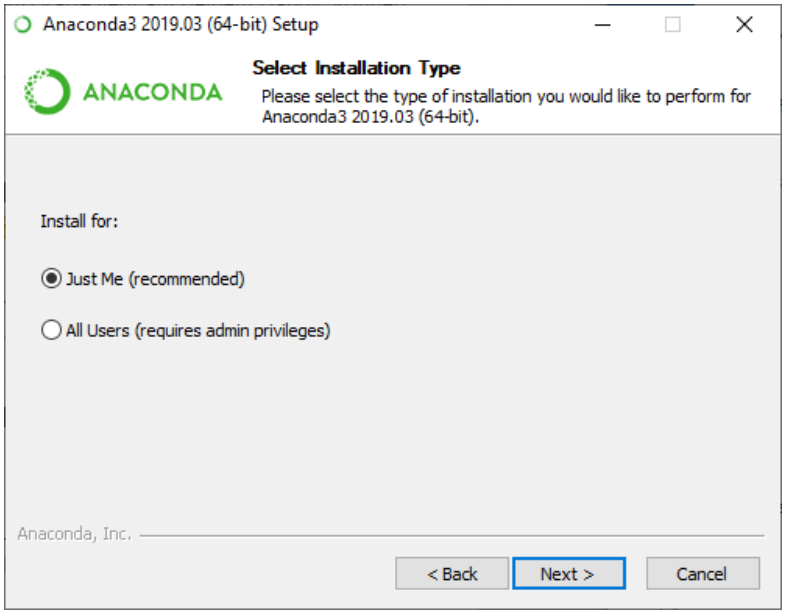

OpenModelica1.13.264bit Setup
Choose Install Location
Choose the folder in which to install OpenModelica1.13.264bit.
Please do not install OpenModelica in a directory that contains spaces for example "C:\Program Files\OpenModelica". Keep if possible the default directory suggested by the installer.
Destination Folder
C:\OpenModelica1.13.264bit
Browse...
Space required: 5.6GB
Space available: 198.7GB
Copyright 2019 OpenModelica
< Back
Next >
Cancel

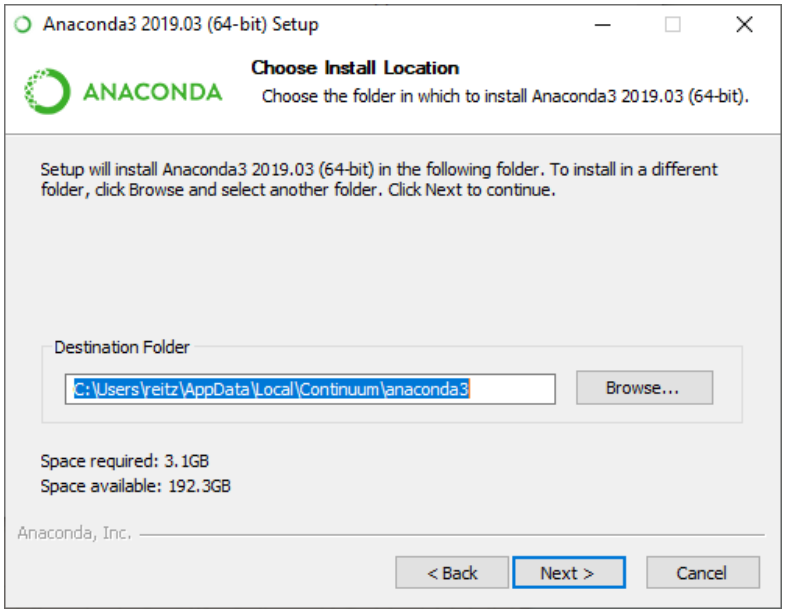

OpenModelica1.13.264bit Setup
Choose Start Menu Folder
Choose a Start Menu folder for the OpenModelica1.13.264bit shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
OpenModelica
7-Zip
Accessibility
Accessories
Administrative Tools
Adobe
Anaconda3 (64-bit)
Android Studio
ANSYS 17.0
ANSYS 18.2
ANSYS 2019 R1
Cisco
COMSOL Multiphysics 5.4
Copyright 2019 OpenModelica
< Back
Install
Cancel

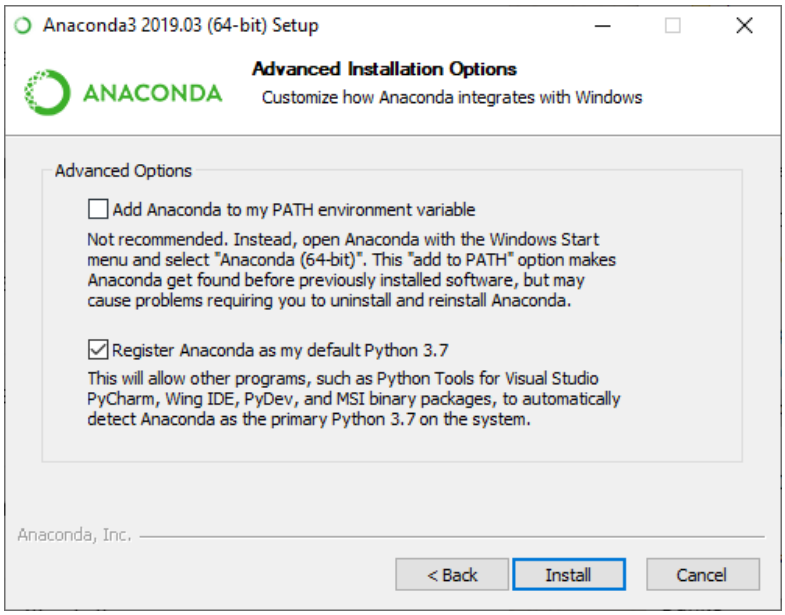
4. Installation von Anaconda mit Defaulteinstellungen



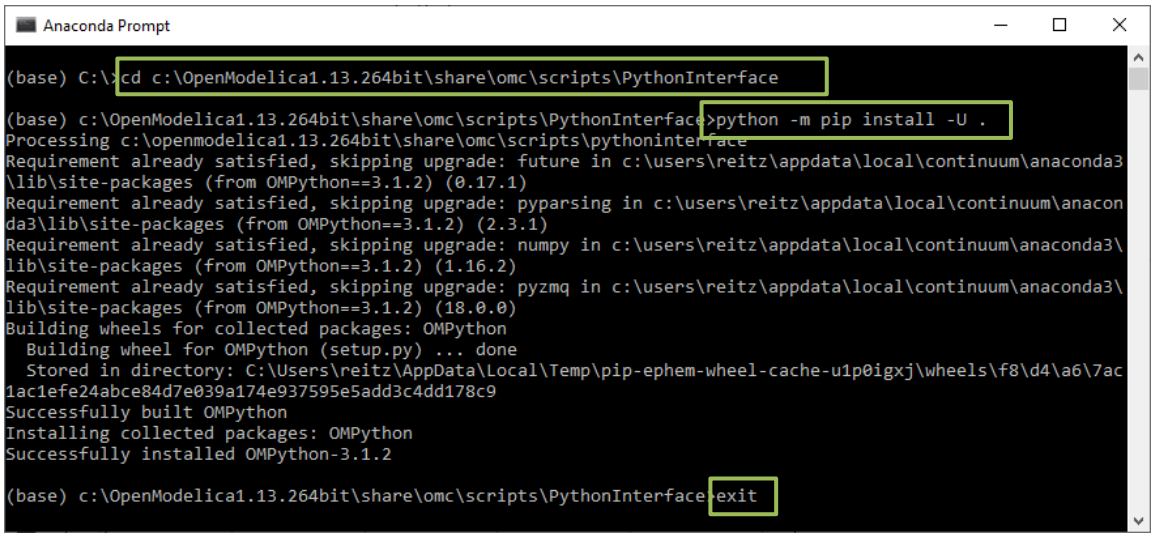



Anaconda3 2019.03 (64-bit) Setup
ANACONDA
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2019.03 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel

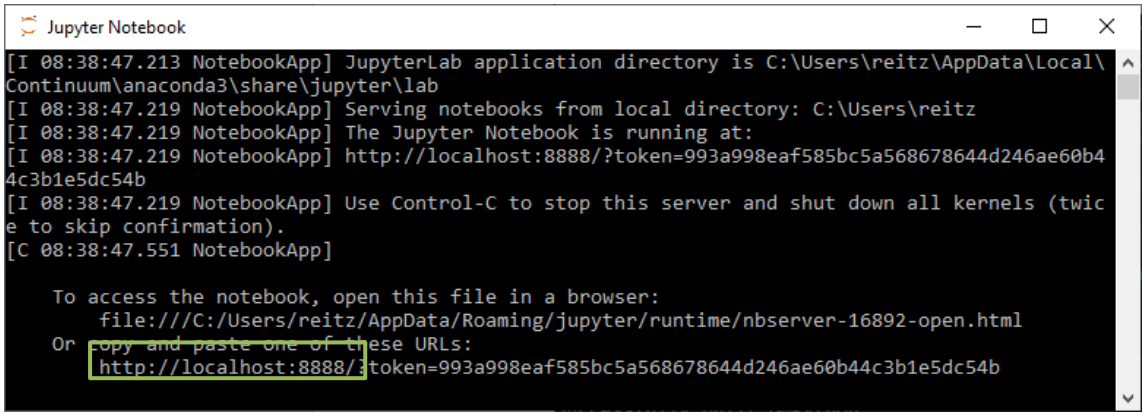

Anaconda3 2019.03 (64-bit) Setup
ANACONDA
Choose Install Location
Choose the folder in which to install Anaconda3 2019.03 (64-bit).
Setup will install Anaconda3 2019.03 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Users\reitz\AppData\Local\Continuum\anaconda3
Browse...
Space required: 3.1GB
Space available: 192.3GB
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2019.03 (64-bit) Setup
ANACONDA
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Add Anaconda to my PATH environment variable
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.
Register Anaconda as my default Python 3.7
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.
Anaconda, Inc.
< Back
Install
Cancel

5. Anaconda Prompt aus Windows-Startmenü starten
6. `cd c:\OpenModelica1.13.264bit\share\omc\scripts\PythonInterface`
7. `python -m pip install -U .`
8. Der Punkt am Ende des letzten Befehls ist wichtig! Die Ausgaben sollten fehlerfrei sein.
9. Anaconda Prompt mit `exit` beenden



10. Jupyter Notebook aus Windows-Startmenü starten



11. Webbrowser öffnet einen neuen Tab mit dem Jupyter Browser und zeigt die Dateien im Nutzerverzeichnis an. Falls dies nicht automatisch geschieht, ist die Adresse „http://localhost:8888“ im Browser zu öffnen

12. Im Tab Files geht man rechts auf New und wählt Python 3
13. Ein neues Notebook öffnet sich
14. In die Eingabe gibt man ein:
```
from OMPython import OMCSessionZMQ
omcs=OMCSessionZMQ()
omcs.sendExpression("getVersion()")
```
15. Mit `Shift-Enter` oder Klick auf `Run` im oberen Menü wird die Eingabe ausgeführt und es kommt der Output: `'v1.13.2 (64-bit)'` bei funktionierender Schnittstelle zwischen Python und OpenModelica

```
In [1]: from OMPython import OMCSessionZMQ
        omcs = OMCSessionZMQ()
        omcs.sendExpression("getVersion()")

        2019-05-03 07:51:30,945 - OMPython - INFO - OMC Server is up and running
        ica.port.054b6ca187fe43609d338ced237f3dab pid=17532

Out[1]: 'v1.13.2 (64-bit)'
```

5.2 JSON-Datei für die Grundtopologie

```
{
  "topology": {
    "title": "Ak30 test topology",
    "nodes": [
      {
        "node": {
          "type": "Generator",
          "ID": "generator1"
        }
      },
      {
        "node": {
          "type": "Battery",
          "ID": "battery1",
          "parameters": {
            "OCV": 49,
            "number_p": 2,
            "number_s": 6
          },
          "_comment": "Batterie48V"
        }
      },
      {
        "node": {
          "type": "Battery",
          "ID": "battery2",
          "parameters": {
            "OCV": 14,
            "number_p": 2,
            "number_s": 6
          },
          "_comment": "Batterie12V"
        }
      },
      {
        "node": {
          "type": "DCDC_Converter",
          "ID": "DCDC_Converter1",
          "parameters": {
            "Iout_max": 200,
            "Vin_max": 65,
            "Vin_min": 0,
            "Vref": 12,
            "control_p": 50,
            "fault_type": 0
          },
          "_comment": "DCDC"
        }
      },
      {
        "node": {
          "type": "PDU",
          "ID": "pdu1"
        }
      },
      {
        "node": {
          "type": "PDU",
          "ID": "pdu2"
        }
      },
      {
        "node": {
```

```
64.           "type": "PDU",
65.           "ID": "pdu3"
66.         }
67.       },
68.       {
69.         "node": {
70.           "type": "PDU",
71.           "ID": "pdu4"
72.         }
73.       },
74.       {
75.         "node": {
76.           "type": "PDU",
77.           "ID": "pdu5"
78.         }
79.       },
80.       {
81.         "node": {
82.           "type": "RC_Load",
83.           "ID": "rc_Load1",
84.           "parameters": {
85.             "resistance": 100,
86.             "capacitance": 0.000001
87.           },
88.           "_comment": "SR Lenkung"
89.         }
90.       },
91.       {
92.         "node": {
93.           "type": "RC_Load",
94.           "ID": "rc_Load2",
95.           "parameters": {
96.             "resistance": 100,
97.             "capacitance": 0.001
98.           }
99.         }
100.            },
101.            {
102.              "node": {
103.                "type": "RC_Load",
104.                "ID": "rc_Load3",
105.                "pdu": 1,
106.                "parameters": {
107.                  "resistance": 100,
108.                  "capacitance": 0.001
109.                }
110.              }
111.            },
112.            {
113.              "node": {
114.                "type": "RC_Load",
115.                "ID": "rc_Load4",
116.                "pdu": 1,
117.                "parameters": {
118.                  "resistance": 100,
119.                  "capacitance": 0.001
120.                },
121.                "_comment": "SR ESP"
122.              }
123.            },
124.            {
125.              "node": {
126.                "type": "RC_Load",
127.                "ID": "rc_Load5",
128.                "pdu": 2,
```

```
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                },
                "_comment": "SR Systemrechner"
              }
            },
            {
              "node": {
                "type": "RC_Load",
                "ID": "rc_Load6",
                "pdu": 2,
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                }
              }
            },
            {
              "node": {
                "type": "RC_Load",
                "ID": "rc_Load7",
                "pdu": 3,
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                }
              }
            },
            {
              "node": {
                "type": "RC_Load",
                "ID": "rc_Load8",
                "pdu": 3,
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                },
                "_comment": "SR Systemrechner"
              }
            },
            {
              "node": {
                "type": "RC_Load",
                "ID": "rc_Load9",
                "pdu": 4,
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                }
              }
            },
            {
              "node": {
                "type": "RC_Load",
                "ID": "rc_Load10",
                "pdu": 4,
                "parameters": {
                  "resistance": 100,
                  "capacitance": 0.001
                }
              }
            }
          ],
          "sources": [
```

```
            "DCDC_Converter1.pin_p2",
            "battery2.p"
          ],
          "connections": [
            {
              "connection": {
                "type": "wire",
                "ID": "D1",
                "startnode": "generator1.p",
                "endnode": "pdu5.pin_p",
                "parameters": {
                  "df_wire_cross_section": 2.5,
                  "df_wire_length": 0.5
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D2",
                "startnode": "DCDC_Converter1.pin_p1",
                "endnode": "pdu5.pin_p4",
                "parameters": {
                  "df_wire_cross_section": 2.5,
                  "df_wire_length": 0.5
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D3",
                "startnode": "battery1.p",
                "endnode": "pdu5.pin_p4",
                "parameters": {
                  "df_wire_cross_section": 2.5,
                  "df_wire_length": 0.5
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D5",
                "startnode": "pdu5.pin_l1",
                "endnode": "rc_Load1.p",
                "parameters": {
                  "df_wire_cross_section": 0.5,
                  "df_wire_length": 0.25
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D6",
                "startnode": "pdu5.pin_l2",
                "endnode": "rc_Load2.p",
                "parameters": {
                  "df_wire_cross_section": 0.5,
                  "df_wire_length": 0.25
                }
              }
            },
            {
```

```
259.               "connection": {
260.                 "type": "wire",
261.                 "ID": "D7",
262.                 "startnode": "pdu1.pin_l1",
263.                 "endnode": "rc_Load3.p",
264.                 "parameters": {
265.                   "df_wire_cross_section": 0.5,
266.                   "df_wire_length": 0.25
267.                 }
268.               }
269.             },
270.             {
271.               "connection": {
272.                 "type": "wire",
273.                 "ID": "D8",
274.                 "startnode": "pdu1.pin_l2",
275.                 "endnode": "rc_Load4.p",
276.                 "parameters": {
277.                   "df_wire_cross_section": 0.5,
278.                   "df_wire_length": 0.25
279.                 }
280.               }
281.             },
282.             {
283.               "connection": {
284.                 "type": "wire",
285.                 "ID": "D9",
286.                 "startnode": "pdu2.pin_l1",
287.                 "endnode": "rc_Load5.p",
288.                 "parameters": {
289.                   "df_wire_cross_section": 0.5,
290.                   "df_wire_length": 0.25
291.                 }
292.               }
293.             },
294.             {
295.               "connection": {
296.                 "type": "wire",
297.                 "ID": "D10",
298.                 "startnode": "pdu2.pin_l2",
299.                 "endnode": "rc_Load6.p",
300.                 "parameters": {
301.                   "df_wire_cross_section": 0.5,
302.                   "df_wire_length": 0.25
303.                 }
304.               }
305.             },
306.             {
307.               "connection": {
308.                 "type": "wire",
309.                 "ID": "D11",
310.                 "startnode": "pdu3.pin_l1",
311.                 "endnode": "rc_Load7.p",
312.                 "parameters": {
313.                   "df_wire_cross_section": 0.5,
314.                   "df_wire_length": 0.25
315.                 }
316.               }
317.             },
318.             {
319.               "connection": {
320.                 "type": "wire",
321.                 "ID": "D12",
322.                 "startnode": "pdu3.pin_l2",
323.                 "endnode": "rc_Load8.p",
```

```
                "parameters": {
                  "df_wire_cross_section": 0.5,
                  "df_wire_length": 0.25
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D13",
                "startnode": "pdu4.pin_l1",
                "endnode": "rc_Load9.p",
                "parameters": {
                  "df_wire_cross_section": 0.5,
                  "df_wire_length": 0.25
                }
              }
            },
            {
              "connection": {
                "type": "wire",
                "ID": "D14",
                "startnode": "pdu4.pin_l2",
                "endnode": "rc_Load10.p",
                "parameters": {
                  "df_wire_cross_section": 0.5,
                  "df_wire_length": 0.25
                }
              }
            }
          ],
          "configuration": [
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG1",
                "used-ID": "D1",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG2",
                "used-ID": "D2",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG3",
                "used-ID": "D3",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
```

```
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG4",
                "used-ID": "D4",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG5",
                "used-ID": "D5",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG6",
                "used-ID": "D6",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG7",
                "used-ID": "D7",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG8",
                "used-ID": "D8",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG9",
                "used-ID": "D9",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
```

```
454.             },
455.             {
456.               "config": {
457.                 "type": "wire_connect",
458.                 "ID": "CFG10",
459.                 "used-ID": "D10",
460.                 "params": {
461.                   "wire_cross_section": 0.5,
462.                   "wire_length": 0.25
463.                 }
464.               }
465.             },
466.             {
467.               "config": {
468.                 "type": "wire_connect",
469.                 "ID": "CFG11",
470.                 "used-ID": "D11",
471.                 "params": {
472.                   "wire_cross_section": 0.5,
473.                   "wire_length": 0.25
474.                 }
475.               }
476.             },
477.             {
478.               "config": {
479.                 "type": "wire_connect",
480.                 "ID": "CFG12",
481.                 "used-ID": "D12",
482.                 "params": {
483.                   "wire_cross_section": 0.5,
484.                   "wire_length": 0.25
485.                 }
486.               }
487.             },
488.             {
489.               "config": {
490.                 "type": "wire_connect",
491.                 "ID": "CFG13",
492.                 "used-ID": "D13",
493.                 "params": {
494.                   "wire_cross_section": 0.5,
495.                   "wire_length": 0.25
496.                 }
497.               }
498.             },
499.             {
500.               "config": {
501.                 "type": "wire_connect",
502.                 "ID": "CFG14",
503.                 "used-ID": "D14",
504.                 "params": {
505.                   "wire_cross_section": 0.5,
506.                   "wire_length": 0.25
507.                 }
508.               }
509.             },
510.             {
511.               "config": {
512.                 "type": "wire_connect",
513.                 "ID": "CFG10",
514.                 "used-ID": "SW2C1",
515.                 "params": {
516.                   "wire_cross_section": 0.5,
517.                   "wire_length": 0.25
518.                 }
```

```
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG11",
                "used-ID": "SW2C2",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG12",
                "used-ID": "SW2C3",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            },
            {
              "config": {
                "type": "wire_connect",
                "ID": "CFG13",
                "used-ID": "SW1C1",
                "params": {
                  "wire_cross_section": 0.5,
                  "wire_length": 0.25
                }
              }
            }
          ]
        }
      }
```

5.3 JSON-Datei für das Gesamtbordnetzmodell

```
{
    "topology": {
        "title": "Ak30 test topology",
        "nodes": [
            {
                "node": {
                    "type": "Generator",
                    "ID": "generator1"
                }
            },
            {
                "node": {
                    "type": "Battery",
                    "ID": "battery1",
                    "parameters": {
                        "OCV": 49.0,
                        "number_p": 2,
                        "number_s": 6
                    },
                    "_comment": "Batterie48V"
                }
```

```
            },
            {
                "node": {
                    "type": "Battery",
                    "ID": "battery2",
                    "parameters": {
                        "OCV": 14.0,
                        "number_p": 2,
                        "number_s": 6
                    },
                    "_comment": "Batterie12V"
                }
            },
            {
                "node": {
                    "type": "DCDC_Converter",
                    "ID": "DCDC_Converter1",
                    "parameters": {
                        "Iout_max": 200,
                        "Vin_max": 65,
                        "Vin_min": 0,
                        "Vref": 12,
                        "control_p": 50,
                        "fault_type": 0
                    },
                    "_comment": "DCDC"
                }
            },
            {
                "node": {
                    "type": "PDU",
                    "ID": "pdu1"
                }
            },
            {
                "node": {
                    "type": "PDU",
                    "ID": "pdu2"
                }
            },
            {
                "node": {
                    "type": "PDU",
                    "ID": "pdu3"
                }
            },
            {
                "node": {
                    "type": "PDU",
                    "ID": "pdu4"
                }
            },
            {
                "node": {
                    "type": "PDU",
                    "ID": "pdu5"
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load1",
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 1e-06
```

```
                    },
                    "_comment": "SR Lenkung"
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load2",
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 0.001
                    }
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load3",
                    "pdu": 1,
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 0.001
                    }
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load4",
                    "pdu": 1,
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 0.001
                    },
                    "_comment": "SR ESP"
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load5",
                    "pdu": 2,
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 0.001
                    },
                    "_comment": "SR Systemrechner"
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load6",
                    "pdu": 2,
                    "parameters": {
                        "resistance": 100.0,
                        "capacitance": 0.001
                    }
                }
            },
            {
                "node": {
                    "type": "RC_Load",
                    "ID": "rc_Load7",
                    "pdu": 3,
```

```
                            "parameters": {
                                "resistance": 100.0,
                                "capacitance": 0.001
                            }
                        }
                    },
                    {
                        "node": {
                            "type": "RC_Load",
                            "ID": "rc_Load8",
                            "pdu": 3,
                            "parameters": {
                                "resistance": 100.0,
                                "capacitance": 0.001
                            },
                            "_comment": "SR Systemrechner"
                        }
                    },
                    {
                        "node": {
                            "type": "RC_Load",
                            "ID": "rc_Load9",
                            "pdu": 4,
                            "parameters": {
                                "resistance": 100.0,
                                "capacitance": 0.001
                            }
                        }
                    },
                    {
                        "node": {
                            "type": "RC_Load",
                            "ID": "rc_Load10",
                            "pdu": 4,
                            "parameters": {
                                "resistance": 100.0,
                                "capacitance": 0.001
                            }
                        }
                    }
                ],
                "sources": [
                    "DCDC_Converter1.pin_p2",
                    "battery2.p"
                ],
                "connections": [
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D1",
                            "startnode": "generator1.p",
                            "endnode": "pdu5.pin_p",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D2",
                            "startnode": "DCDC_Converter1.pin_p1",
                            "endnode": "pdu5.pin_p4",
                            "parameters": {
```

```
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D3",
                            "startnode": "battery1.p",
                            "endnode": "pdu5.pin_p4",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D5",
                            "startnode": "pdu5.pin_l1",
                            "endnode": "rc_Load1.p",
                            "parameters": {
                                "df_wire_cross_section": 0.5,
                                "df_wire_length": 0.25
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D6",
                            "startnode": "pdu5.pin_l2",
                            "endnode": "rc_Load2.p",
                            "parameters": {
                                "df_wire_cross_section": 0.5,
                                "df_wire_length": 0.25
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D7",
                            "startnode": "pdu1.pin_l1",
                            "endnode": "rc_Load3.p",
                            "parameters": {
                                "df_wire_cross_section": 0.5,
                                "df_wire_length": 0.25
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "D8",
                            "startnode": "pdu1.pin_l2",
                            "endnode": "rc_Load4.p",
                            "parameters": {
                                "df_wire_cross_section": 0.5,
                                "df_wire_length": 0.25
                            }
                        }
                    },
```

```
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D9",
                        "startnode": "pdu2.pin_l1",
                        "endnode": "rc_Load5.p",
                        "parameters": {
                            "df_wire_cross_section": 0.5,
                            "df_wire_length": 0.25
                        }
                    }
                },
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D10",
                        "startnode": "pdu2.pin_l2",
                        "endnode": "rc_Load6.p",
                        "parameters": {
                            "df_wire_cross_section": 0.5,
                            "df_wire_length": 0.25
                        }
                    }
                },
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D11",
                        "startnode": "pdu3.pin_l1",
                        "endnode": "rc_Load7.p",
                        "parameters": {
                            "df_wire_cross_section": 0.5,
                            "df_wire_length": 0.25
                        }
                    }
                },
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D12",
                        "startnode": "pdu3.pin_l2",
                        "endnode": "rc_Load8.p",
                        "parameters": {
                            "df_wire_cross_section": 0.5,
                            "df_wire_length": 0.25
                        }
                    }
                },
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D13",
                        "startnode": "pdu4.pin_l1",
                        "endnode": "rc_Load9.p",
                        "parameters": {
                            "df_wire_cross_section": 0.5,
                            "df_wire_length": 0.25
                        }
                    }
                },
                {
                    "connection": {
                        "type": "wire",
                        "ID": "D14",
                        "startnode": "pdu4.pin_l2",
```

```
                            "endnode": "rc_Load10.p",
                            "parameters": {
                                "df_wire_cross_section": 0.5,
                                "df_wire_length": 0.25
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "TD1",
                            "startnode": "DCDC_Converter1.pin_p2",
                            "endnode": "pdu1.pin_p",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "wire",
                            "ID": "TD2",
                            "startnode": "battery2.p",
                            "endnode": "pdu1.pin_p",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "switch_two_conn",
                            "ID": "SW2C1",
                            "startnode": "pdu1.pin_p2",
                            "endnode": "pdu2.pin_p2",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5,
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "switch_two_conn",
                            "ID": "SW2C2",
                            "startnode": "pdu1.pin_p4",
                            "endnode": "pdu3.pin_p2",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5,
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "switch_two_conn",
                            "ID": "SW2C3",
                            "startnode": "pdu1.pin_p6",
                            "endnode": "pdu4.pin_p2",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
```

```
                                "df_wire_length": 0.5,
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "switch_two_conn",
                            "ID": "SW2C4",
                            "startnode": "pdu3.pin_p4",
                            "endnode": "pdu4.pin_p4",
                            "parameters": {
                                "df_wire_cross_section": 2.5,
                                "df_wire_length": 0.5,
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "backbone_conn",
                            "ID": "B1",
                            "startnode": "pdu1.pin_p5",
                            "endnode": "pdu3.pin_p5",
                            "parameters": {
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "backbone_conn",
                            "ID": "B2",
                            "startnode": "pdu1.pin_p5",
                            "endnode": "pdu4.pin_p5",
                            "parameters": {
                                "Fuse": 20.0
                            }
                        }
                    },
                    {
                        "connection": {
                            "type": "backbone_conn",
                            "ID": "B3",
                            "startnode": "pdu2.pin_p5",
                            "endnode": "pdu4.pin_p5",
                            "parameters": {
                                "Fuse": 20.0
                            }
                        }
                    }
                ],
                "configuration": [
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG1",
                            "used-ID": "D1",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
```

```
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG2",
                        "used-ID": "D2",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
                {
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG3",
                        "used-ID": "D3",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
                {
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG4",
                        "used-ID": "D4",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
                {
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG5",
                        "used-ID": "D5",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
                {
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG6",
                        "used-ID": "D6",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
                {
                    "config": {
                        "type": "wire_connect",
                        "ID": "CFG7",
                        "used-ID": "D7",
                        "params": {
                            "wire_cross_section": 0.5,
                            "wire_length": 0.25
                        }
                    }
                },
```

```
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG8",
                            "used-ID": "D8",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG9",
                            "used-ID": "D9",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG10",
                            "used-ID": "D10",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG11",
                            "used-ID": "D11",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG12",
                            "used-ID": "D12",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG13",
                            "used-ID": "D13",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
```

```
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG14",
                            "used-ID": "D14",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG10",
                            "used-ID": "SW2C1",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG11",
                            "used-ID": "SW2C2",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG12",
                            "used-ID": "SW2C3",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    },
                    {
                        "config": {
                            "type": "wire_connect",
                            "ID": "CFG13",
                            "used-ID": "SW1C1",
                            "params": {
                                "wire_cross_section": 0.5,
                                "wire_length": 0.25
                            }
                        }
                    }
                ]
            }
        }
```

Teil 4

Topologie-Auswahl durch Anwendung der Bewertungskriterien

Prof. Dr.-Ing. Stephan Frei
M. Sc. Michael Gerten
M. Sc. Marvin Rübartsch

Technische Universität Dortmund
Arbeitsgebiet Bordsysteme

Inhaltsverzeichnis

1 Einleitung

Durch die zunehmende Automatisierung des Kfz ergeben sich neue Sicherheitsanforderungen. Besonders im Hinblick auf das hochautomatisierte Fahren, in dem es auch für sicherheitsrelevante Funktionen keine mechanische Rückfallebene mehr geben wird, ist die Gewährleistung eines sicheren Zustands auch im Falle eines Komponentenfehlers oder Ausfalls sicherzustellen. Dies bedeutet, dass auch erhöhte Anforderungen an das Energiebordnetz gestellt werden. Das Verhalten muss „fail-operational“ sein, sodass auch im Fehlerfall die Versorgung sicherheitskritischer Komponenten so lange aufrechterhalten werden kann, bis das Fahrzeug einen sicheren Zustand erreicht hat. Hierdurch ergibt sich die Notwendigkeit der Untersuchung neuartiger, hochverfügbarer Bordnetztopologien und intelligenter Absicherungskonzepte. Klassische Konzepte, wie die Absicherung der Leitungen mit einfachen Schmelzsicherungen, können die erhöhten Sicherheitsanforderungen aufgrund ihrer langsamen, passiven Funktionsweise und fehlender Diagnosemöglichkeiten nicht erfüllen und werden zunehmend durch intelligente elektronische Sicherungen (eFuses bzw. Smart Fuses) ersetzt.

Bei der Auslegung zukünftiger Energiebordnetze muss daher die Versorgungsstabilität von sicherheitskritischen Verbrauchern bereits zu Beginn des Entwicklungsprozesses berücksichtigt werden. Dazu ist eine umfassende Analyse potentieller Topologien nötig in der die Auswirkungen der Vielzahl der möglichen Fehler auf das Gesamtsystem untersucht und evaluiert werden müssen. Da dies ist in experimentellen Untersuchungen nicht mit akzeptablem Aufwand möglich ist, spielen Simulationen hier eine zentrale Rolle.

Durch den Arbeitskreis AK 30 wurde bereits 2015 das Forschungsvorhaben „Simulationsgestützte Methodik zum Entwurf intelligenter Energiesteuerung in zukünftigen Kfz-Bordnetzen“ initiiert, welches zu wichtigen Erkenntnissen und Modellen für die Bordnetzsimulation geführt hat, welche in der Modellierungssprache Modelica implementiert wurden [1]. Die im Rahmen dieses Vorhabens entstandene Modellbibliothek, mit der das funktionale Verhalten in Bordnetzen bei Verwendung intelligenter Stromverteiler untersucht werden kann, ist der Ausgangspunkt weiterführender Arbeiten [2]. Im Nachfolgeprojekt „Simulationsgestützte Analyse und Bewertung der Fehlertoleranz von Kfz-Bordnetzen“ wurden die bestehenden Funktionsmodelle um realistische Modelle wichtiger möglicher Fehler ergänzt, die Änderungen von Parametern infolge von Ausfällen in den Komponenten aufgrund von Alterungsprozessen oder mechanischen Einwirkungen berücksichtigen [3]. Neben den grundlegenden Topologieanalysen und der Realisierung eines realen Bordnetzdemonstrators wurde außerdem ein Automatisierungsframework entwickelt, um die zu untersuchenden Fehler in die Simulationsmodelle einzuprägen und automatisiert zu simulieren.

Die bereits erarbeiteten Methoden sind zwar geeignet, Bordnetze im Hinblick auf ihre Fehlertoleranz zu untersuchen, jedoch sind die Simulationen, die in der Simulationsumgebung OpenModelica [4] realisiert werden, zu zeitaufwändig um aus einer sehr großen Anzahl an möglichen Bordnetzarchitekturen die geeignetsten auszuwählen. Ziel des Arbeitspakets 4 im Forschungsprojekt „Methodische Ansätze zur Auswahl von Bordnetzstrukturen mit erhöhten Zuverlässigkeitsanforderungen“ ist es daher, effiziente Methoden zu entwickeln, um eine

schnelle Vorauswahl vielversprechender Topologien zu ermöglichen. In diesem Arbeitspaket werden dazu verschiedene Verfahren untersucht, um die Auswirkungen von Fehlern auf das Bordnetz durch einfache Berechnungen schnell abschätzen zu können. Nach einer messtechnischen Validierung des entwickelten Verfahrens wird außerdem eine Bewertungsmethodik vorgeschlagen, um die einzelnen generierten Simulationsergebnisse zu einer Gesamtmetrik der Topologie zusammenzufassen. Abschließend wird mit dem implementierten und automatisierten Workflow eine konkrete Beispieltopologie aus dem Arbeitskreis analysiert.

2 Methodik zur effizienten Auswahl von Bordnetz-Topologien

Der grundsätzliche Ablauf der im Projekt entstandenen Methodik zur Vorauswahl von Bordnetzarchitekturen ist in Abbildung 1 dargestellt. Eingangsparameter sind die zu untersuchende Topologie und Informationen über die sicherheitsrelevanten Verbraucher, sowie die zu verwendenden Kriterien zur Evaluation der Spannungsstabilität. Auf Basis dieser Informationen sollen in der ersten Stufe die Lastspannungen, die aus den zu untersuchenden Fehlerszenarien resultieren, zunächst durch Rechteckpulse approximiert werden. Ziel dieser Worstcase-Abschätzung ist es, die unkritischen Szenarien nach einer ersten Evaluation bereits auszusortieren. Nur die potentiell kritischen Fehlerszenarien werden anschließend in der zweiten Stufe mithilfe einer transienten Schaltungssimulation genauer berechnet. Diese transiente Simulation weist weiterhin Vereinfachungen und Worstcase-Annahmen im Vergleich zur Modelica-Simulation auf. In der letzten Stufe des Workflows müssen schließlich nur noch wenige zeitaufwändige Modelica-Simulation durchgeführt werden. Hierbei werden nur noch die Szenarien betrachtet, die im Vorfeld bereits als kritische Fehlerszenarien identifiziert werden konnten. Abschließend wird aus den simulierten Spannungsverläufen eine Gesamtbewertung der Spannungsstabilität der Topologie berechnet.

Die einzelnen Schritte des dargestellten Workflows werden in den folgenden Unterkapiteln im Detail beschrieben.

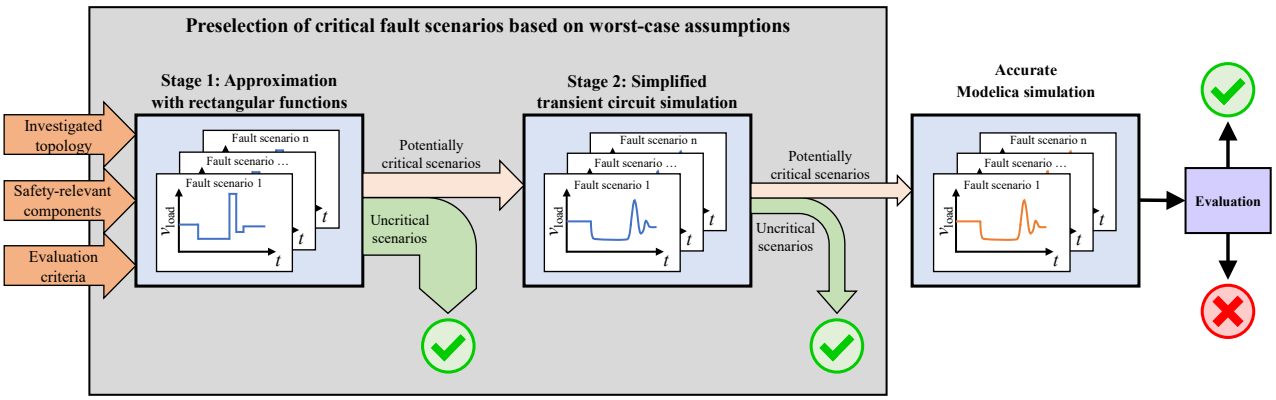


Abbildung 1: Schematischer Überblick des entwickelten Topologie-Evaluierungs-Workflows

Die entwickelten Methoden und Algorithmen sind in MATLAB implementiert und zu einer Code-Bibliothek zusammengefasst. Die Ordnerstruktur der einzelnen Funktionen ist in Abbildung 2 dargestellt. *topology_evaluation.m* stellt hierbei das Hauptskript dar, welches die restlichen Skripte aufruft und in welchem die wichtigsten Parameter definiert werden. Zur Durchführung der Schaltungssimulation wird zusätzlich die MATLAB/Octave-basierte Eigenentwicklung *Networkanalysis* verwendet. Die wichtigsten Funktionen werden in den entsprechenden folgenden Abschnitten vorgestellt.

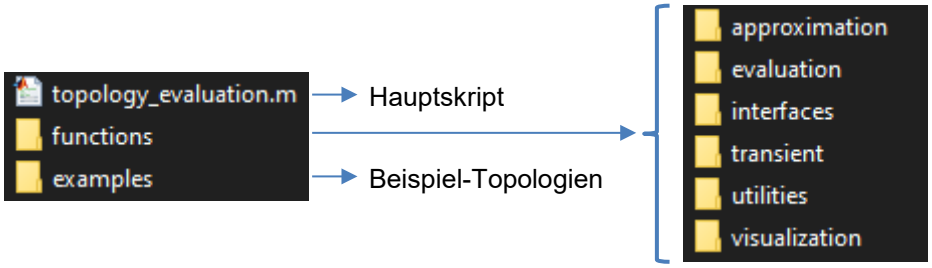


Abbildung 2: Struktur der Code-Bibliothek

2.1 Schaltungssimulation

Die statischen und transienten Schaltungssimulationen, die im Folgenden detaillierter beschrieben werden, werden mit dem vorhandenen Tool *Networkanalysis* in MATLAB/Octave durchgeführt. Dieses Tool basiert auf der Methode der modifizierten Knotenpotentialanalyse [5]. Für die Schaltungsdefinition ist eine Netzliste notwendig, welche aus den einzelnen Bauelementen (z.B. passive Bauelemente, Schalter, …) und deren Anordnung zueinander besteht. Vom Projektpartner Fraunhofer wird die zu untersuchende Bordnetztopologie in Form einer Netzliste erzeugt, welche auf den Modelldefinitionen der Modelica-Bibliothek basiert. Zur Weiterverarbeitung müssen diese Modelle also zunächst in jeweiligen **E**rsatz**s**chalt**b**ild-Elemente (**ESB**) konvertiert werden. Hierfür wurde die Funktion *interfaces/convert_netlist.m* erstellt, deren Funktionsweise in Abbildung 3 schematisch dargestellt ist. Die Modelldefinitionen des Schnittstellenformats, welche den Modelica-Modellen des vorangegangenen Projekts entsprechen, werden zunächst zeilenweise eingelesen; anschließend werden die entsprechend parametrierten ESB-Elemente in der Ziel-Netzliste erzeugt und miteinander verbunden. Die verwendete Namenskonventionen sind in Tabelle 1 dargestellt. Mit dieser resultierenden Netzliste wird nun der vorgestellte Vorselektions-Workflow durchgeführt.

Tabelle 1: Namenskonventionen für Netzlisten-Elemente

Komponente	**Netzlisten-Bezeichnung**
Leitung XX	R1XX / L1XX
Last XX	R2XX / C2XX, ESR: R7XX
Innenwiderstand XX	R3XX
Kurzschluss-/Leerlauffehler XX	R4XX
Generatorwiderstand XX	R5XX
DC/DC-Widerstand XX	R6XX

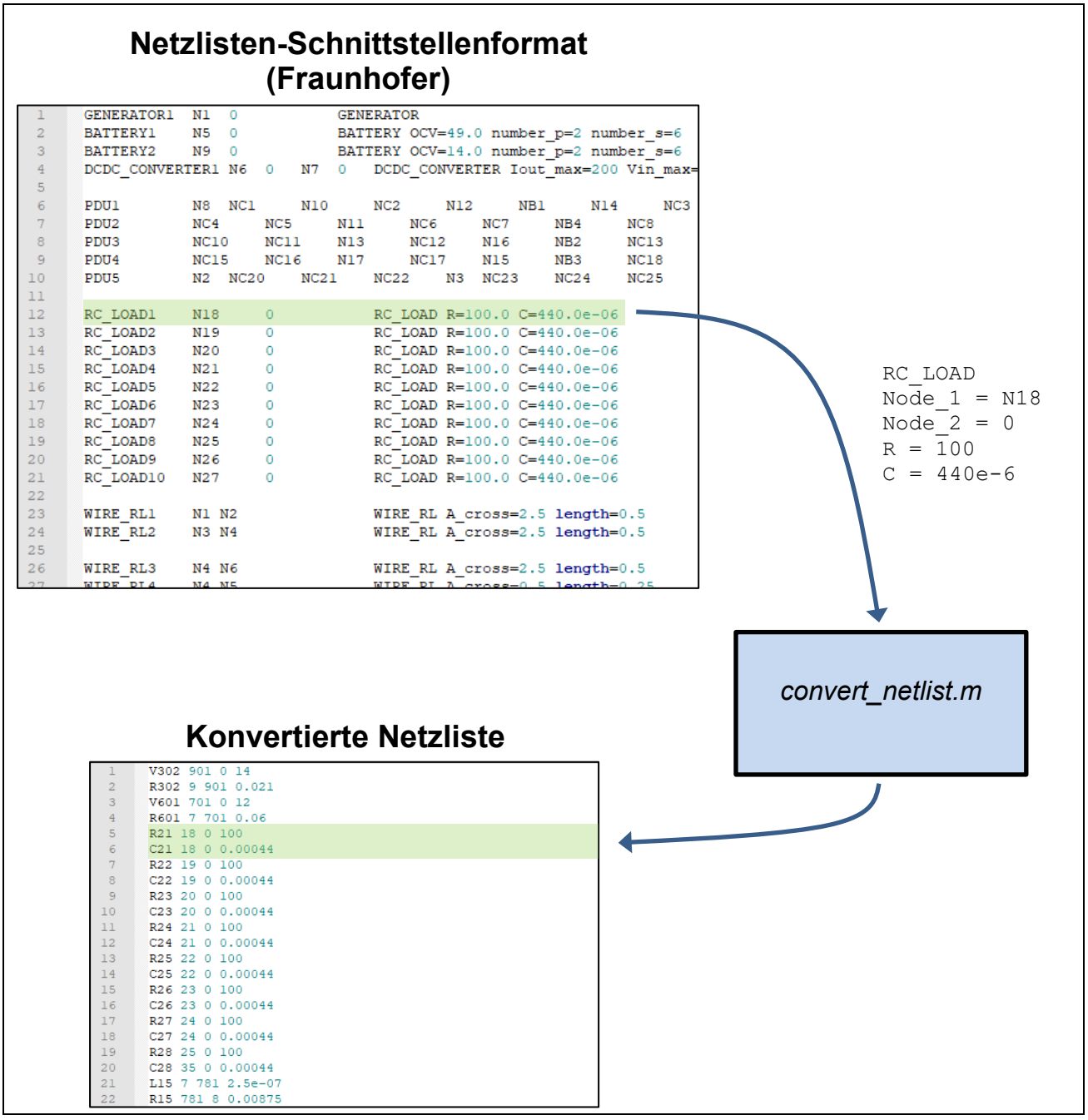


Abbildung 3: Schematischer Ablauf der Netzlisten-Konvertierung zur Weiterverarbeitung

2.2 Approximation durch Rechteckpulse

Schaltvorgänge (z. B. durch Sicherungen) und Leitungsfehler führen zu transienten Spannungs- und Stromänderungen innerhalb des Energiebordnetzes. Es wird angenommen, dass das System nach diesen transienten Ereignissen zurück in einen quasi-statischen Zustand konvergiert. Um den Rechenaufwand für die Durchführung einer großen Anzahl von transienten Simulationen zu vermeiden, wird das Fehlerverhalten des Systems daher zunächst angenähert indem die transienten Pulse durch statische Simulationen und analytische Berechnungen unter Worstcase-Annahmen approximiert werden. Wie in Abbildung 4 veranschaulicht ist, werden die Über- und Unterspannungen, die aufgrund der Schaltvorgänge an

den Lasten entstehen, somit durch eine rechteckige Funktion angenähert. Es müssen also zum einen die statischen Spannungen im Nominal- und Fehlerfall berechnet werden, außerdem sind die, durch die Sicherungen ausgelösten, Schaltpulse abzuschätzen.

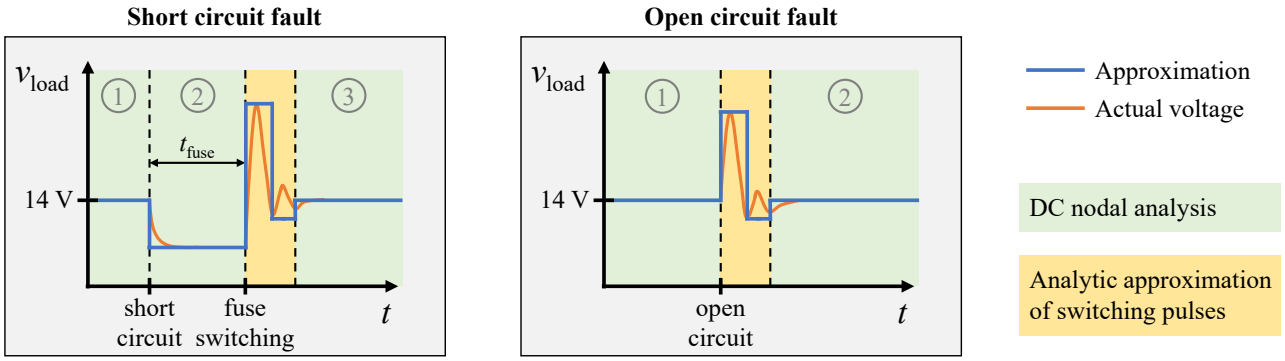


Abbildung 4: Exemplarische Lastspannungsverläufe eines Kurzschluss- (links) und Leerlauffehlers (rechts) inklusive Approximation durch Rechteckpulse

2.2.1 Statische Spannungen

Die statischen Spannungen können mithilfe einer statischen Simulation der Netzliste der Topologie bestimmt werden. Eine DC-Simulation der nominalen Netzliste liefert die nominalen Lastspannungen (Bereich 1 in Abbildung 4). Der statische Zustand, in den das System nach einem Leitungsfehler konvergiert, kann ebenfalls durch eine DC-Simulation ermittelt werden, indem der Kurzschluss oder die Unterbrechung der Leitung in die Netzliste integriert wird (Bereich 2 in Abbildung 4). Der statische Zustand, der sich nach dem Schalten einer Sicherung ergibt, kann analog dazu simuliert werden, indem der Zustand der Sicherung innerhalb der Netzliste angepasst wird (Bereich 3, Abbildung 4).

2.2.2 Schaltpulse

Die Unterbrechung von Strömen durch Sicherungen oder Leitungsbrüche kann steile Stromflanken verursachen. In Kombination mit den Leitungsinduktivitäten können diese Schwingungen im System anregen und zu großen Überspannungen an den Verbrauchern führen (Abbildung 4, gelber Bereich). Veranschaulichen lässt sich dieses Verhalten durch das in Abbildung 5 dargestellte, vereinfachte Ersatzschaltbild. Durch Öffnen des Schalters kann darüber kein Strom mehr fließen. Durch die im Magnetfeld gespeicherte Energie treibt die Leitungsinduktivität den Strom aber zunächst weiter. Zum Schaltzeitpunkt t_0 wirkt diese daher wie eine ideale Stromquelle, welche den initialen Strom I_{t0} in die übrig gebliebene Last treibt. An dieser Last wird daher ein Überspannungspuls U_1 beobachtet (grüne Kurve). Ziel der in diesem Kapitel durchgeführten Approximation ist es nun, diesen Puls durch eine Rechteckfunktion in Amplitude und Zeitkonstante abzuschätzen (blaue Kurve).

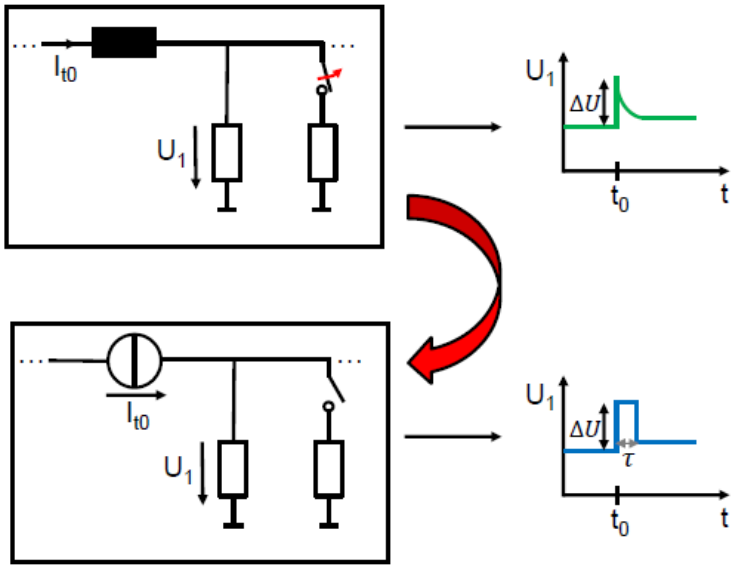


Abbildung 5: Entstehung von Spannungspulsen durch Schaltvorgänge im Bordnetz

Bei dieser Abschätzung werden Worstcase-Annahmen getroffen, um die Pulse nach oben abzuschätzen und keine kritischen Fälle zu übersehen. Für eine rechnerisch effiziente Annäherung der Systemreaktion auf die Schaltvorgänge werden analytische Überlegungen angestellt. Ergänzend zu Abbildung 5 müssen in einem realistischeren Bordnetz-Szenario mehrere Induktivitäten der unterschiedlichen Last- und Versorgungspfade berücksichtigt werden. Zunächst wird hierfür der Fall von zwei Last- und zwei Versorgungspfaden näher betrachtet (siehe Abbildung 6). Bevor die eigentlich gesuchten Lastspannungspulse berechnet werden können, soll zunächst untersucht werden, wie sich die Zweigströme im Schaltmoment aufteilen. Wie sich in Voruntersuchungen mit etablierten Netzwerksimulationsprogrammen herausgestellt hat, sind hierfür lediglich die Leitungsinduktivitäten von signifikanter Bedeutung. Daher werden die Lastimpedanzen für die nachfolgende Herleitung der Berechnungsvorschrift vernachlässigt und nur die Leitungen mit ihren Induktivitäten und Widerständen betrachtet (Index S: Versorgungsleitung (supply), Index L: Lastleitung).

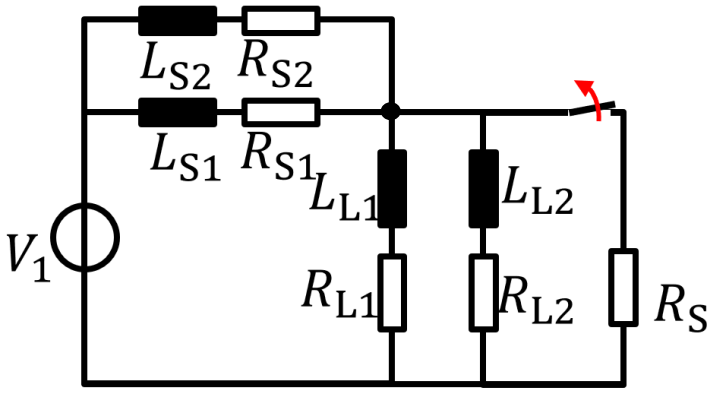


Abbildung 6: Betrachtetes Beispiel zur Herleitung der Zweigströme im Schaltmoment

Die Maschengleichung für die Versorgungsleitungen lautet in diesem Fall:

$$R_{\mathrm{S1}} I_{\mathrm{S1}} + L_{\mathrm{S1}} \frac{dI_{\mathrm{S1}}}{dt} = R_{\mathrm{S2}} I_{\mathrm{S2}} + L_{\mathrm{S2}} \frac{dI_{\mathrm{S2}}}{dt} \tag{1}$$

Durch Transformation in den Laplace-Bereich und Betrachtung für $t \to 0$ (Grenzwertsatz) und der Differentiationsregel ergibt sich daraus unter Berücksichtigung der Zweigströme vor dem Schaltmoment $I_{\mathrm{S}i}^{\mathrm{pre}}$:

$$\begin{gathered} \lim_{s\to\infty} s \cdot \left[(R_{\mathrm{S1}} + sL_{\mathrm{S1}}) I_{\mathrm{S1}}(s) - L_{\mathrm{S1}} I_{\mathrm{S1}}^{\mathrm{pre}}\right] = \lim_{s\to\infty} s \cdot \left[(R_{\mathrm{S2}} + sL_{\mathrm{S2}}) I_{\mathrm{S2}}(s) - L_{\mathrm{S2}} I_{\mathrm{S2}}^{\mathrm{pre}}\right] \\ L_{\mathrm{S1}} \left(\lim_{s\to\infty} s I_{\mathrm{S1}}(s) - I_{\mathrm{S1}}^{\mathrm{pre}}\right) = L_{\mathrm{S2}} \left(\lim_{s\to\infty} s I_{\mathrm{S2}}(s) - I_{\mathrm{S2}}^{\mathrm{pre}}\right) \end{gathered} \tag{2}$$

Mit $I_{\mathrm{S}i}(t=0) = \lim_{s\to\infty} s I_{\mathrm{S}i}(s) = I_{\mathrm{S}i}^{\mathrm{post}}$ folgt daraus ein Zusammenhang für die Stromdifferenz $\Delta I_{\mathrm{S}i}$ unmittelbar vor und nach dem Schaltvorgang:

$$L_{\mathrm{S1}}\left(I_{\mathrm{S1}}^{\mathrm{post}} - I_{\mathrm{S1}}^{\mathrm{pre}}\right) = L_{\mathrm{S2}}\left(I_{\mathrm{S2}}^{\mathrm{post}} - I_{\mathrm{S2}}^{\mathrm{pre}}\right) \tag{3}$$

$$\Leftrightarrow L_{\mathrm{S1}} \Delta I_{\mathrm{S1}} = L_{\mathrm{S2}} \Delta I_{\mathrm{S2}} \tag{4}$$

Analog lässt sich derselbe Zusammenhang auch für die Lastzweige herleiten:

$$\Leftrightarrow L_{\mathrm{L1}} \Delta I_{\mathrm{L1}} = L_{\mathrm{L2}} \Delta I_{\mathrm{L2}} \tag{5}$$

Des Weiteren gilt für die Stromsumme durch den erzwungenen Ausgleich im Schaltmoment:

$$I_{\mathrm{ges}}^{\mathrm{post}} = I_{\mathrm{S1}}^{\mathrm{post}} + I_{\mathrm{S2}}^{\mathrm{post}} = I_{\mathrm{L1}}^{\mathrm{post}} + I_{\mathrm{L2}}^{\mathrm{post}} \tag{6}$$

Um die konkrete Aufteilung der Ströme auf die einzelnen Zweige nach dem Schaltvorgang zu bestimmen, ist eine weitere Gleichung notwendig. Dafür wird die Masche aus Quelle, Spannungsabfall V_{S} über den Versorgungsleitungen und V_{L} über den Lastleitungen im Laplace-Bereich betrachtet:

$$V_1/s = V_{\mathrm{S}}(s) + V_{\mathrm{L}}(s) \tag{7}$$

Unmittelbar nach dem Schalten gilt unter Anwendung des Anfangswertsatzes damit:

$$\lim_{s\to\infty} s \cdot \frac{V_1}{s} = \lim_{s\to\infty} s \cdot \left(V_{\mathrm{S}}(s) + V_{\mathrm{L}}(s)\right) \tag{8}$$

$$\Rightarrow V_1 = \frac{(L_{S1}R_{V2} + R_{S1}L_{S2})\left(I_{ges}^{post} - I_{ges}^{pre}\right) + L_{S1}L_{S2}\frac{dI_{ges}}{dt}|_{t=0+} + V_S^{pre}(L_{S1} + L_{S2})}{L_{S1} + L_{S2}}$$
$$+ \frac{(L_{L1}R_{L2} + R_{L1}L_{S2})\left(I_{ges}^{post} - I_{ges,L}^{pre}\right) + L_{L1}L_{L2}\frac{dI_{ges,L}}{dt}|_{t=0+} + V_L^{pre}(L_{L1} + L_{L2})}{L_{L1} + L_{L2}}$$

$$= \frac{(L_{S1}R_{S2} + R_{S1}L_{S2})\Delta I_{ges} + V_V^{pre}(L_{S1} + L_{S2})}{L_{S1} + L_{S2}}$$
$$+ \frac{(L_{L1}R_{L2} + R_{L1}L_{L2})\Delta I_{ges,L} + V_L^{pre}(L_{L1} + L_{L2})}{L_{L1} + L_{L2}}$$
$$+ \left(\frac{L_{S1}L_{S2}}{L_{S1} + L_{S2}}\frac{dI_{ges}}{dt}|_{t=0+} + \frac{L_{L1}L_{L2}}{L_{L1} + L_{L2}}\frac{dI_{ges,L}}{dt}|_{t=0+}\right)$$

Die rot markierten Terme divergieren für einen idealen Schaltvorgang, müssen sich also gegenseitig kompensieren, damit die endliche Spannung V_1 resultieren kann:

$$0 = \lim_{s\to\infty} \frac{L_{S1}L_{S2}}{L_{S1} + L_{S2}}\left(sI_{ges}(s) - I_{ges}^{pre}\right) + \frac{L_{L1}L_{L2}}{L_{L1} + L_{L2}}\left(sI_{ges}(s) - I_{ges,L}^{pre}\right) \quad (9)$$

$$\Rightarrow \frac{L_{S1}L_{S2}}{L_{S1} + L_{S2}}\Delta I_{ges,S} + \frac{L_{L1}L_{L2}}{L_{L1} + L_{L2}}\Delta I_{ges,L} = 0 \quad (10)$$

Aus den Gleichungen (4), (5), (6) und (10), ergibt sich ein lineares Gleichungssystem (LGS), um die vier unbekannten Zweigströme unmittelbar nach dem Schalten zu bestimmen.

Die gefundenen Zusammenhänge lassen sich auf beliebig viele Zweige erweitern. Ein solcher verallgemeinerter Systemknoten ist in Abbildung 7 dargestellt. Hier wird angenommen, dass dieser Knoten über *m* verschiedene Versorgungspfade und *n* Lastpfade verfügt. Der zu untersuchende Schaltvorgang findet dabei in einem weiteren Lastpfad statt. Der Strom, der in diesem Pfad abgeschaltet wird, wird mit I_{sw} bezeichnet.

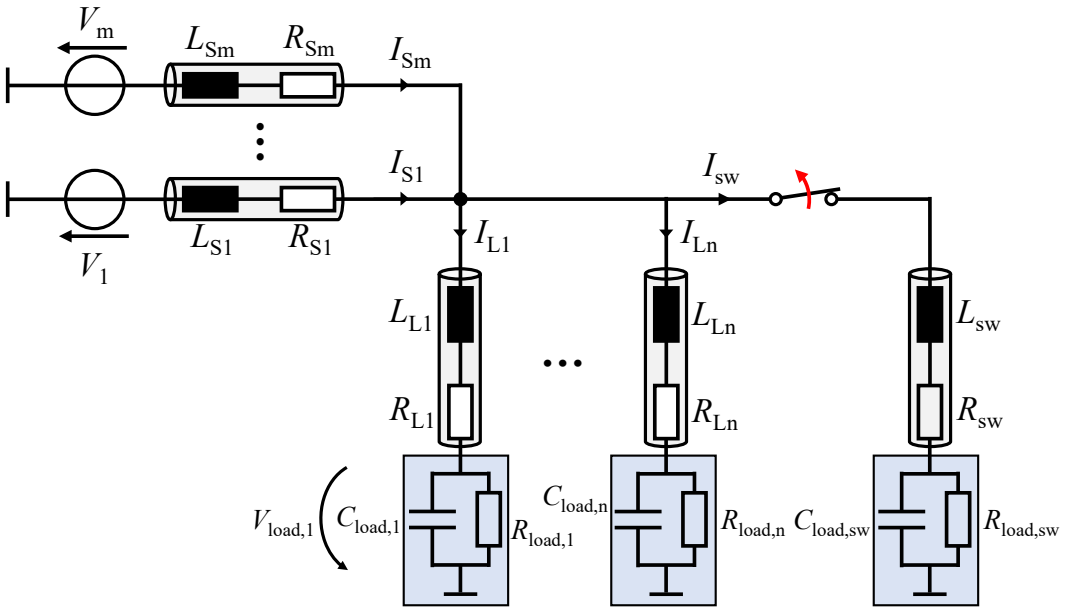


Abbildung 7: Allgemeiner Systemknoten mit *m* Versorgungspfaden, *n* Lastpfaden und einem zusätzlich abgeschalteten Lastknoten

Das hergeleitete Gleichungssystem wird für diesen allgemeinen Fall erweitert, sodass für die $m+n$ unbekannten Ströme im Schaltmoment die folgenden $m+n$ unabhängigen Gleichungen gelten:

$$(L_{S1} \parallel \cdots \parallel L_{Sm}) \sum_{i=1}^{m} \left(I_{Si}^{post} - I_{Si}^{pre}\right) + (L_{L1} \parallel \cdots \parallel L_{Ln}) \sum_{j=1}^{n} \left(I_{Lj}^{post} - I_{Lj}^{pre}\right) = 0 \tag{11}$$

$$\sum_{i=1}^{m} I_{Si}^{post} = \sum_{j=1}^{n} I_{Lj}^{post} \tag{12}$$

$$\begin{aligned} \Delta I_{S1} L_{S1} &= \cdots = \Delta I_{Sm} L_{Sm} \\ \Delta I_{L1} L_{L1} &= \cdots = \Delta I_{Lm} L_{Lm} \end{aligned} \tag{13}$$

Dieses LGS kann in Matrix-Vektor-Schreibweise umformuliert werden:

$$\underbrace{\begin{bmatrix} (L_{S1} \parallel \cdots \parallel L_{Sm}) & (L_{S1} \parallel \cdots \parallel L_{Sm}) & \dots & (L_{L1} \parallel \cdots \parallel L_{Ln}) & (L_{L1} \parallel \cdots \parallel L_{Ln}) & \dots \\ 1 & 1 & \dots & -1 & -1 & \dots \\ L_{S1} & -L_{S2} & \dots & 0 & 0 & \dots \\ 0 & L_{S2} & \dots & 0 & 0 & \dots \\ \vdots & \vdots & \dots & \vdots & \vdots & \ddots \\ 0 & 0 & \dots & L_{L1} & -L_{L2} & \ddots \\ \vdots & \vdots & \dots & \vdots & \vdots & \ddots \end{bmatrix}}_{\boldsymbol{A}} \cdot \underbrace{\begin{bmatrix} I_{S1}^{post} \\ I_{S2}^{post} \\ \vdots \\ I_{L1}^{post} \\ I_{L2}^{post} \\ \vdots \end{bmatrix}}_{\boldsymbol{x}}$$

$$= \underbrace{\begin{bmatrix} I_{S1}^{pre}(L_{S1} \parallel \cdots \parallel L_{Sm}) + I_{S2}^{pre}(\dots) + \cdots \\ 0 \\ I_{S1}^{pre} L_{S1} - I_{S2}^{pre} L_{S2} \\ I_{S2}^{pre} L_{S2} - I_{S3}^{pre} L_{S3} \\ \vdots \\ I_{L1}^{pre} L_{L1} - I_{L2}^{pre} L_{L2} \\ \vdots \end{bmatrix}}_{\boldsymbol{b}} \tag{14}$$

Die Matrix ***A*** enthält somit die bekannten Leitungsinduktivitäten; der Lösungsvektor ***b*** enthält zusätzlich noch die bekannten Zweigströme vor dem Schaltvorgang. Die gesuchten Ströme nach dem Schaltvorgang im Vektor $\boldsymbol{x}$ können demnach durch Lösung des LGS bestimmt werden:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{15}$$

Somit ist der Strompeak durch die Last direkt nach dem Schaltvorgang bekannt. Gesucht ist allerdings die maximale Lastspannung. Im Falle eines rein ohmschen Verbrauchers, kann diese über das ohmsche Gesetz bestimmt werden:

$$V_{\mathrm{load},j} = R_{\mathrm{load},j} \cdot I_{Lj}^{post} \tag{16}$$

In der Regel ist allerdings näherungsweise von einem RC-Verhalten der Verbraucher auszugehen, wie beispielsweise in einem Steuergerät mit Eingangskapazität. Hier muss der Ladevorgang der Kapazität mitberücksichtigt werden, um die maximale Lastspannung zu bestimmen. Um diesen abzuschätzen, wird von folgendem ESB ausgegangen:

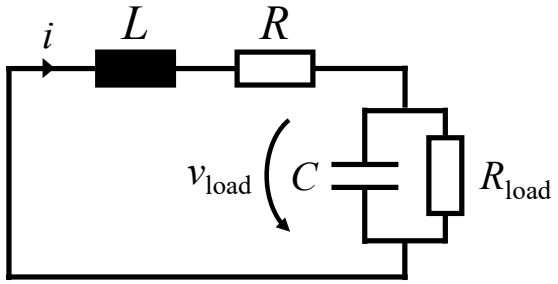


Abbildung 8: ESB zur analytischen Abschätzung des Spannungspeaks an RC-Lasten

Der Aufladevorgang der Kapazität kann nun durch Aufstellen und Lösen der Differentialgleichung dieser Schaltung analytisch bestimmt werden. Um diese Berechnung zu vereinfachen,

werden keine parallelen Last- oder Versorgungspfade berücksichtigt. Damit die Lastspannung durch diese Vereinfachung nicht unterschätzt wird, sind Worstcase-Annahmen bezüglich des Stroms und der Leitungsimpedanz zu treffen; da die Wechselwirkung zwischen den parallelen Lastpfaden nicht berücksichtigt wird, wird der Laststrom unmittelbar nach dem Schaltvorgang als Summe aller Lastströme angenommen. Für die Anfangsbedingung $i(t = 0)$ gilt also:

$$i(0) = I_{\mathrm{L1}}^{\mathrm{post}} + \cdots + I_{\mathrm{Ln}}^{\mathrm{post}} \tag{17}$$

Die Induktivität setzt sich aus der Lastleitungsinduktivität und der Induktivität der Versorgungsleitungen zusammen. Als Worstcase-Abschätzung wird hier daher, die größte Induktivität aller Versorgungsleitungen verwendet:

$$L = L_{\mathrm{L}} + \max(L_{\mathrm{S1}}, \dots, L_{\mathrm{Sm}}) \tag{18}$$

Zugleich setzt sich der Widerstand aus dem Lastleitungswiderstand und dem minimalen Widerstand der Versorgungsleitungen zusammen, da dies der geringsten Dämpfung des RLC-Kreises und somit dem Worstcase der Lastspannung entspricht:

$$R = R_{\mathrm{L}} + \min(R_{\mathrm{S1}}, \dots, R_{\mathrm{Sm}}) \tag{19}$$

Die analytische Lösung der Lastspannung v_{load} liefert den folgenden Ausdruck:

$$v_{\mathrm{load}}(t) = \underbrace{\frac{2i(0)LR_{\mathrm{load}}}{\sqrt{\sigma}}}_{A} \cdot \exp\left(-\underbrace{\left(\frac{L + CRR_{load}}{2CLR_{load}}\right)}_{B} t \right) \cdot \sin\left(\underbrace{\frac{\sqrt{\sigma}}{2CLR_{\mathrm{load}}}}_{C} \cdot t \right) \tag{20}$$

mit

$$\sigma = 2CLRR_{\mathrm{load}} + 4CLRR_{\mathrm{load}}^2 - C^2R^2R_{\mathrm{load}}^2 - L^2 \tag{21}$$

Für die Abschätzung durch Rechteckpulse sollen nun jeweils der erste positive und negative Peak dieser gedämpften Schwingung berücksichtigt werden. Die Zeitpunkte $t_{\max}$ und $t_{\min}$ dieser Peaks können bestimmt werden, indem die Ableitung zu null gesetzt wird:

$$\frac{d}{dt} v_{\mathrm{load}}(t) = \frac{d}{dt} Ae^{-Bt} \sin(Ct) \stackrel{!}{=} 0 \tag{22}$$

$$\Rightarrow \qquad t_{\max} = \frac{\arctan\frac{A}{B}}{A} \qquad \wedge \qquad t_{\min} = \frac{\pi + \arctan\frac{A}{B}}{A} \tag{23}$$

Die maximale (und minimale) Lastspannung infolge des Schaltvorgangs kann berechnet werden, indem der Ausdruck $v_{\mathrm{load}}(t_{\max})$ (und $v_{\mathrm{load}}(t_{\min})$) ausgewertet wird.

Für die Pulsdauer der beiden Amplituden wird die Zeitkonstante der gedämpften Schwingung verwendet:

$$\tau_{\mathrm{RC}} = \frac{1}{B} \tag{24}$$

Analog wird für den Fall einer rein ohmschen Last die in (16) bestimmte Spannung für die Zeitkonstante τ_{R} approximiert:

$$\tau_{\mathrm{R}} = \frac{L}{R} \tag{25}$$

Mit den in diesem Kapitel vorgestellten Methoden lassen sich also Schaltpulse durch die in Abbildung 7 dargestellten Strukturen durch einfaches Lösen von linearen Gleichungssystemen und der Auswertung analytischer Ausdrücke abschätzen. Eine klare Trennung von Versorgungs- und Lastleitungen, wie sie hier bisher angenommen wurde, ist allerdings in der Realität nicht immer gegeben. Dies ist zum Beispiel bei einer Topologie mit mehreren Power Distribution Units (PDUs), die beispielsweise hierarchisch oder ringförmig verbunden sind, der Fall. Aus diesem Grund müssen hier weitere Worstcase-Abschätzungen getroffen werden, um keine kritischen Pulse zu übersehen; konkret sollen alle PDUs auf die bekannte Struktur aus Abbildung 7 reduziert werden. Hierfür werden zunächst alle Verbindungen, die weitere PDUs versorgen, vernachlässigt. Es wird daher angenommen, dass der komplette Schaltstrom in Richtung der Lasten der aktuell betrachteten PDU getrieben wird. Dies wird für jede PDU unter Annahme des maximalen Versorgungsstroms der einzelnen PDUs wiederholt. Somit wird die Spannung aller Lasten nach oben abgeschätzt, damit auch bei komplexeren Topologien keine kritischen Fälle übersehen werden.

2.2.3 Algorithmus

Die in den vorangegangenen Unterkapiteln beschriebenen Methoden werden zu einem Algorithmus erweitert, welcher das Fehlerverhalten einer Bordnetztopologie für alle möglichen Kurzschluss- und Leerlauffehler durch rechteckförmige Pulse approximiert. Die entsprechende Funktion lautet *approximate_fault_behavior.m* und ihr Ablauf ist in Abbildung 9 schematisch dargestellt.

Zunächst werden iterativ alle möglichen Leitungsbrüche untersucht; der Leitungsbruch wird an der betreffenden Stelle in die Netzliste eingebracht und statisch simuliert. Anschließend werden alle Systemknoten geprüft: Ist der Strom in den Knoten hinein im Fehlerfall kleiner als im Nominalzustand, hat in einem der Lastpfade ein Schaltvorgang (bzw. der Leerlauffehler) stattgefunden. Die Lastspannungen werden für diesen Knoten nun, wie in Abschnitt 2.2.2 beschrieben, durch Lösen des LGS und Auswertung der analytischen Abschätzungen berechnet. Abschließend wird geprüft, ob im fehlerhaften Zustand eine Sicherung überlastet ist und auslösen wird. Ist dies der Fall, wird die Routine rekursiv für diesen Schaltvorgang ausgeführt, bis kein weiterer Schaltvorgang mehr auftritt. Dies ist möglich, da das Auslösen einer Sicherung analog zum Leerlauffehler eine leitende, stromführende Verbindung trennt.

Als nächstes werden Kurzschlussfehler an allen möglichen Systemknoten untersucht; analog zur Leerlauf-Routine wird nach Einfügen des Fehlers in die Netzliste zunächst eine statische

Simulation des Fehlerzustands durchgeführt. Führt der Kurzschlussfehler dazu, dass eine Sicherung schaltet, wird dieser Schaltvorgang durch Aufruf der Leerlauf-Routine mit der fehlerbehafteten Netzliste rekursiv berechnet, bis keine weitere Sicherung mehr schaltet.

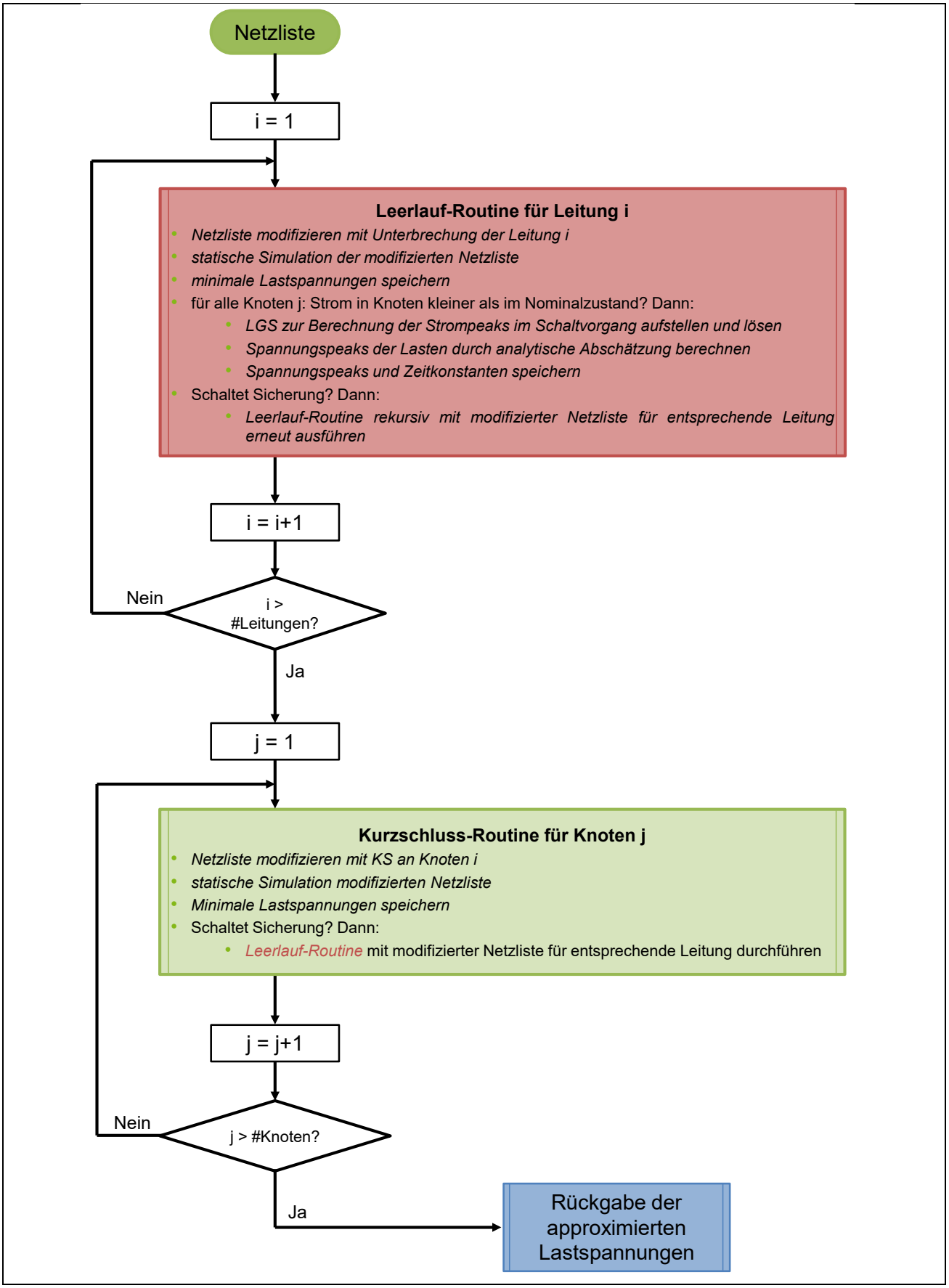


Abbildung 9: Schematischer Ablauf der Funktion *approximate_fault_behavior.m* zur Approximation durch Rechteckpulse

2.2.4 Einfaches Beispiel

Die entwickelte Methodik, insbesondere die Abschätzung der Schaltpulse, soll nun an einem einfachen Beispiel untersucht werden. Hierfür wird die in Abbildung 10 dargestellte Topologie verwendet. Sie besteht aus einer Quelle, vier Leitungen und drei Lasten. Die fünf Systemknoten sind durch rote Zahlen gekennzeichnet. Die auf Ebene der Netzliste vorhandenen ESB-Elemente und die dafür notwendigen zusätzlichen Knoten sind in Leitung 1 angedeutet (Knoten **12**1 als Verbindung zwischen Systemknoten **1** und **2**).

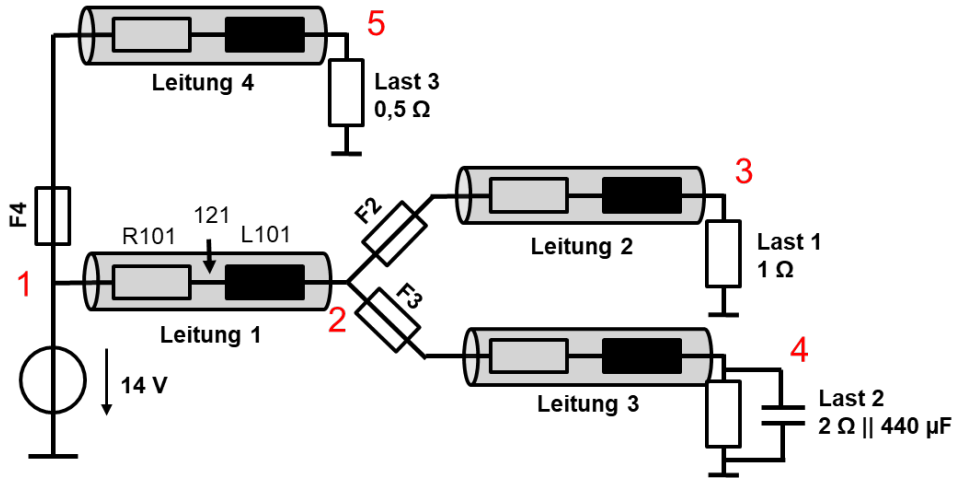
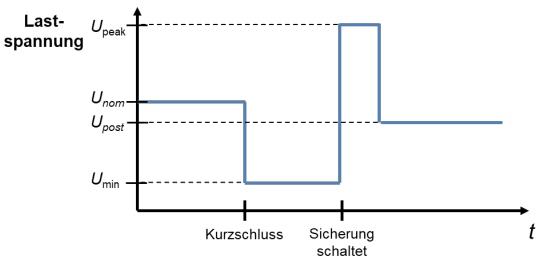


Abbildung 10: Einfache Topologie zur beispielhaften Anwendung der entwickelten Methodik

Diese Topologie wird mit dem im vorherigen Abschnitt beschriebenen Algorithmus analysiert. Die resultierenden Lastspannungen für alle untersuchten Kurzschlussfehler sind in Abbildung 11 dargestellt.



		Kurzschlussfehler				
		Knoten 1	Knoten 2	Knoten 3	Knoten 4	Knoten 5
Last 1	U_{min}	13,5 V	0 V	0 V	11,5 V	13,5 V
	U_{peak}	--	--	0 V	192,5 V	13,5 V
	U_{post}	0 V	0 V	0 V	13,5 V	13,5 V
Last 2	U_{min}	13,75 V	0 V	11 V	0 V	13,75 V
	U_{peak}	--	--	36 V	0 V	13,75 V
	U_{post}	0 V	0 V	13,75 V	0 V	13,75 V
Last 3	U_{min}	13,5 V	13,5 V	13,5 V	13,5 V	0 V
	U_{peak}	--	--	13,5 V	13 V	0 V
	U_{post}	0 V	0 V	13,5 V	13,5 V	0 V

Abbildung 11: Approximierte Spannungen für Kurzschlussfehler-Szenarien der beispielhaften Topologie

Für eine Validierung der approximierten Schaltpulse, welche durch die Auslösevorgänge der Sicherungen erzeugt werden, wird eine transiente Simulation der Topologie in LTspice durchgeführt. Beispielhaft ist dieser Vergleich für den Fehlerfall Kurzschluss an Knoten 3 und in Abbildung 12 für die Spannung der Last 2 dargestellt. Durch den Kurzschlussfehler entsteht ein hoher Kurzschlussstrom, der die Sicherung F2 zum Auslösen bringt. Durch diesen Schaltvorgang wird eine Oszillation mit signifikantem Spannungspeak an der ohmsch-kapazitiven Last 2 verursacht. Wie zu erkennen ist, liefert die rechteckförmige Approximation eine sehr gute Abschätzung der erstehenden Spannungspeaks und deren Zeitkonstanten.

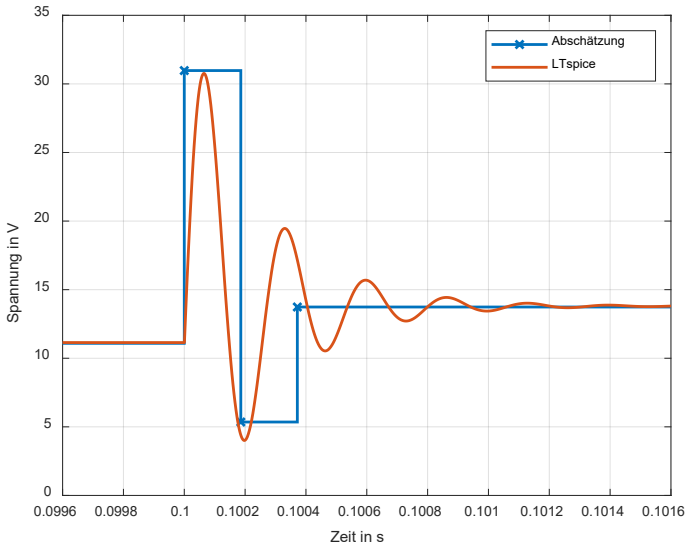


Abbildung 12: Spannung Last 2 bei Kurzschluss an Knoten 3. Vergleich der Approximation des Schaltpulses der Sicherung mit transienter LTspice-Simulation

2.3 Transiente Schaltungssimulation

Die in der ersten Stufe der Vorselektion als potentiell kritisch identifizierten Szenarien, also Szenarien, die zu Unter- oder Überspannungen an sicherheitsrelevanten Lasten führen, welche die definierten Kriterien überschreiten, werden nun weiter analysiert. Alle Szenarien, bei denen die Rechteckapproximation zu unkritischen Spannungen führt oder bei denen eine dauerhafte Unterbrechung der Stromversorgung festgestellt wird (definiert als dauerhafte statische Versorgungsspannung < 100 mV), müssen nicht weiter untersucht werden. Für eine genauere Analyse wird in der zweiten Stufe der Vorauswahl eine transiente Schaltungssimulation der jeweiligen Fehlerfälle durchgeführt. Wie bei den DC-Schaltungssimulationen wird auch bei der transienten Simulation die MNA auf der Grundlage der Netzliste der Topologie verwendet. Die Ergebnisse der transienten Simulation sind deutlich näher an der Realität als in der vorherigen Stufe, da viel mehr Zeitschritte simuliert werden und die Worstcase-Annahmen für die analytischen Betrachtungen in Abschnitt 2.2 hier nicht nötig sind. Die Berechnung einer transienten Simulation mit vielen einzelnen Zeitschritten dauert zwar länger als die wenigen notwendigen Berechnungen in der ersten Stufe, ist aber immer noch schneller als eine genaue Modelica-Simulation mit zusätzlichen nichtlinearen Effekten und unvermeidbarer Kompilierung der Modelle.

2.4 Schnittstelle zur Modelica-Umgebung

Die Szenarien, welche nach den beiden Stufen der Vorselektion als potentiell kritisch identifiziert wurden, müssen jetzt abschließend untersucht werden. Hierfür werden die im Vorgängerprojekt entwickelten Workflows und Simulationsmodelle in Modelica als Referenz verwendet [3]. Im Rahmen dieses vorangegangenen Projekts wurde vom Projektpartner Fraunhofer ein Automatisierungs-Workflow entwickelt, mit dem Fehlersimulationen durchgeführt werden können. Ziel ist es nun, die Schnittstelle zwischen der Modelica-Umgebung und der hier beschriebenen Vorselektion ebenfalls zu automatisieren. Die durchzuführenden Modelica-Fehlersimulationen werden in einer JSON-Datei definiert. Ein beispielhafter Ausschnitt ist in Abbildung 13 dargestellt. Der hier beschriebene eingefügte Leitungsfehler wird parametriert und die hierfür notwendigen Änderungen in der Bordnetztopologie werden unter „disConnections“ und „connections“ definiert.

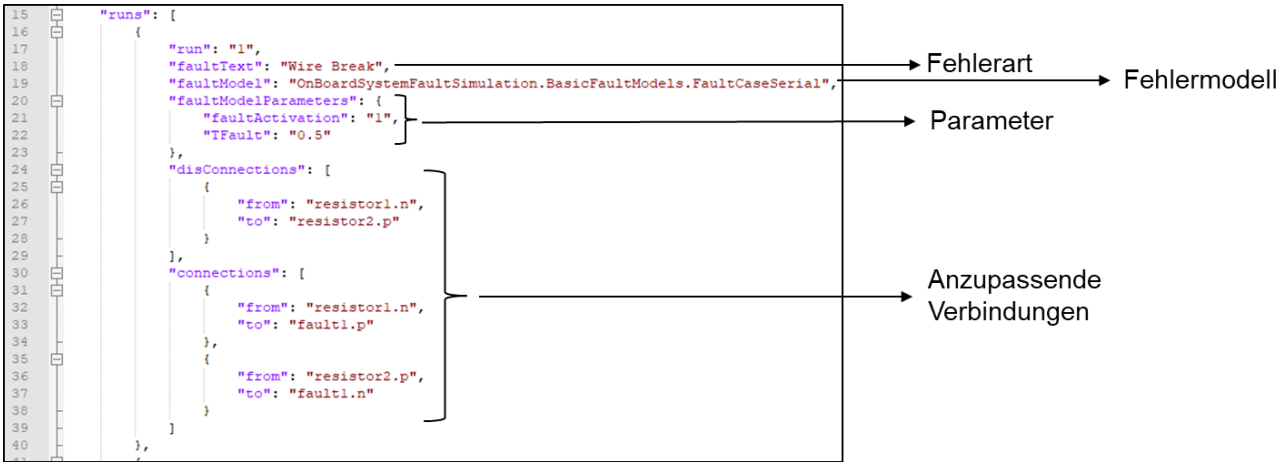


Abbildung 13: Exemplarischer Ausschnitt einer JSON-Datei zur Durchführung einer automatisierten Fehlersimulation in Modelica

Es ist also eine JSON-Datei zu generieren, um die in der Vorselektion übrig gebliebenen Fehler in die anschließende Modelica-Simulation zu integrieren. Zu diesem Zweck wurde die Funktion *interfaces/create_JSON_faultinjection.m* implementiert, welche eine JSON-Datei erzeugt und die benötigten Einträge für die Fehlersimulation integriert.

3 Messtechnische Validierung

Die in Kapitel 2 vorgestellte Methode der Vorselektion von Bordnetztopologien wurde auch messtechnisch validiert. Dazu wird ein Kurzschlussfehler-Szenario in einem Teilbordnetz in einem realen Laboraufbau untersucht und mit den simulierten Ergebnissen der Vorselektion verglichen. Das dafür verwendete Teilbordnetz ist in Abbildung 14 dargestellt. Es besteht aus einer 14 V-Quelle (angenommener DC-DC-Konverter, im Prüfstand repräsentiert durch eine Laborquelle), einer PDU und 3 Lasten. Last 1 ist ein Widerstand von 0,25 Ω (Annahme eines rein ohmschen Verbrauchers mit einer Stromaufnahme von 56 A). Last 2 ist eine Parallelschaltung von einer 3 mF Kapazität und einem 10 Ω Widerstand (Annahme eines ohmsch-kapazitiven Verbrauchers). Last 3 ist eine Parallelschaltung von einer 10 mF Kapazität und einem 1 Ω Widerstand. Diese Last kann als eine Nachbildung von mehreren an die PDU angeschlossenen Verbrauchern gesehen werden. Durch Zusammenfassung dieser parallelen Lasten ergibt sich die hohe Kapazität von 10 mF. Die Leitung ist mit 7 cm als sehr kurz angenommen. Die Leitungsquerschnitte sind entsprechend der Nominalströme ausgelegt. Last 1 und 2 sind durch elektronische Sicherungen (eFuses) von Infineon (Last 1) beziehungsweise STMicroelectronic (Last 2) abgesichert.

Das betrachtete Fehlerszenario ist ein Kurzschluss der Last 1.

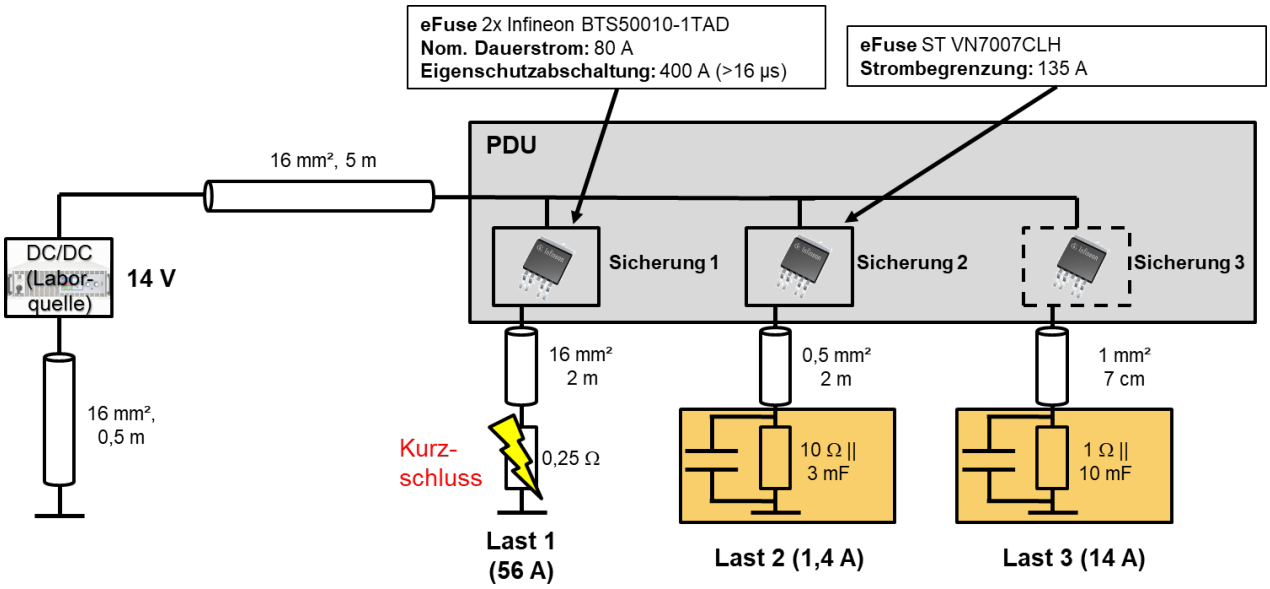


Abbildung 14: Schematische Darstellung des zur messtechnischen Validierung verwendeten Teilbordnetzes

Neben der Messung dieses Fehlerszenarios im Laborprüfstand wird für das Teilbordnetz ebenfalls ein Simulationsmodell in Form einer Netzliste erzeugt und simulativ in den beiden Stufen der Vorselektion untersucht. Abbildung 15 zeigt alle drei verschiedenen Spannungsverläufe, die sich für die Lasten 2 und 3 infolge des Fehlers ergeben.

In der Messung ist zunächst zu erkennen, dass mit Eintritt des Kurzschlussfehlers bei $t = 0$, die Spannung an beiden Lasten aufgrund des ansteigenden Kurzschlussstromes auf etwa 11 V (Last 2) bzw. unter 10 V (Last 3) absinkt. Durch das Auslösen der Sicherung 1 steigt

die Spannung wieder an und es ist ein durch die Leitungsinduktivitäten verursachter Spannungspuls zu beobachten. Aufgrund der großen Lastkapazitäten im Gesamtsystem beträgt die Amplitude dieses Pulses allerdings nur etwa 15 V. Durch die transiente Schaltungssimulation wird dieses Verhalten insbesondere an Last 2 gut nachgebildet. Besonders von Interesse ist zudem das Ergebnis der rechteckförmigen Approximation. Diese darf die realen Verläufe nicht unterschätzen, sodass durch die Vorselektion keine kritischen Fälle übersehen werden. Diese Anforderung ist im untersuchten Szenario erfüllt. Durch die getroffenen Abschätzungen sind die positiven und negativen Spannungspeaks deutlich stärker ausgeprägt als in der Realität und werden somit nach oben abgeschätzt; es würde in diesem Fall also kein kritischer Verlauf übersehen werden. Allerdings zeigt sich auch, dass die Abschätzung der Zeitkonstanten der Schaltpulse für den vorliegenden Fall mehrerer paralleler Lasten noch verbessert werden kann.

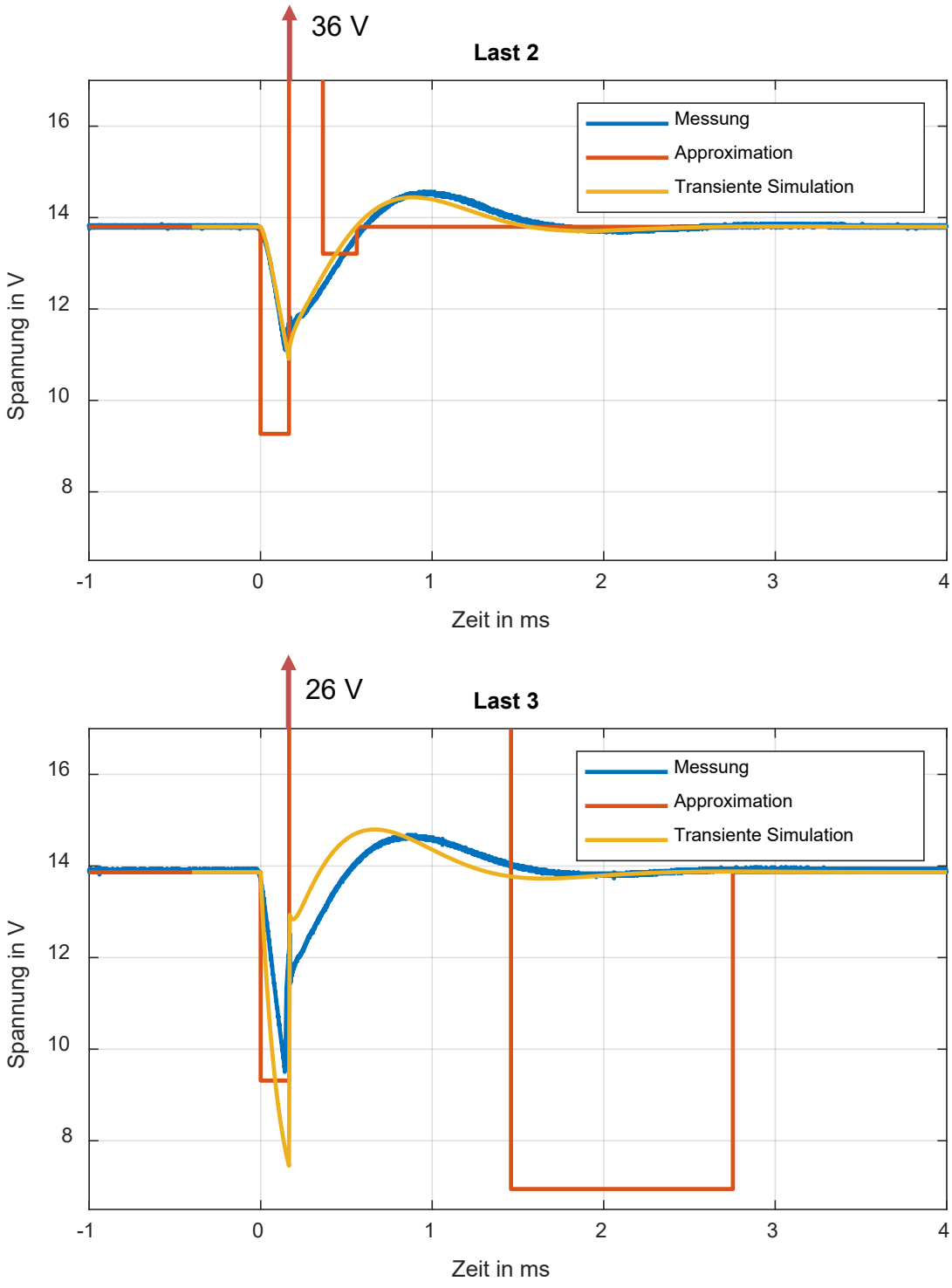


Abbildung 15: Lastspannungen an Last 2 (oben) und Last 3 (unten) bei Kurzschluss der Last 1. Vergleich zwischen Messung (blau), rechteckige Approximation (orange) und transienter Schaltungssimulation (gelb)

4 Automatisierte Bewertung

Ziel des Workflows ist es, auf Basis der simulierten Spannungsverläufe die Fehlertoleranz der Bordnetztopologie zu bewerten. Verschiedene Topologien können so bezüglich ihres Fehlerverhaltens miteinander verglichen werden, damit die robustesten Topologien ausgewählt werden können. Hierfür muss eine Bewertungsmethodik entwickelt werden.

Der grundsätzliche Ablauf einer solchen Bewertung ist in Abbildung 16 dargestellt. Die Simulation liefert für eine Vielzahl an Fehlerszenarien Spannungsverläufe für die unterschiedlichen Lasten des Bordnetzes. Diese Spannungsverläufe können zunächst einzeln bewertet werden. Um aus der Vielzahl an einzelnen Bewertungen nun eine Gesamtmetrik zu berechnen, welche eine aussagekräftige Beurteilung des gesamten Bordnetzes ermöglicht, sind diese Einzelbewertungen geeignet zu gewichten.

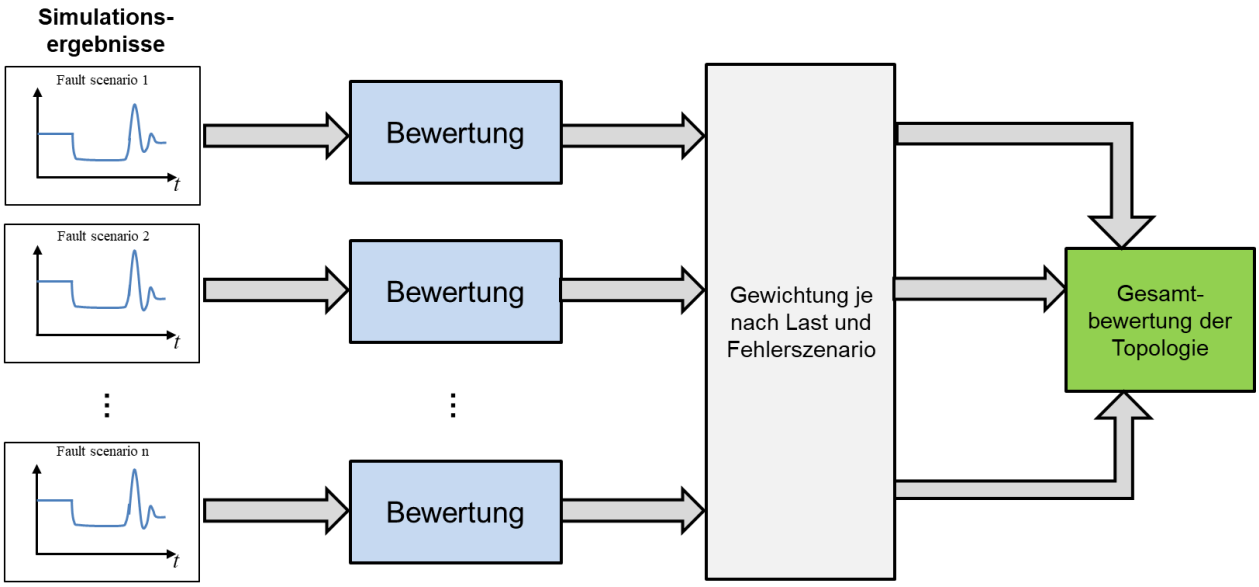


Abbildung 16: Prinzipieller Ablauf zur Bewertung der Spannungsstabilität des Gesamtbordnetzes

4.1 Bewertung einzelner Spannungspulse

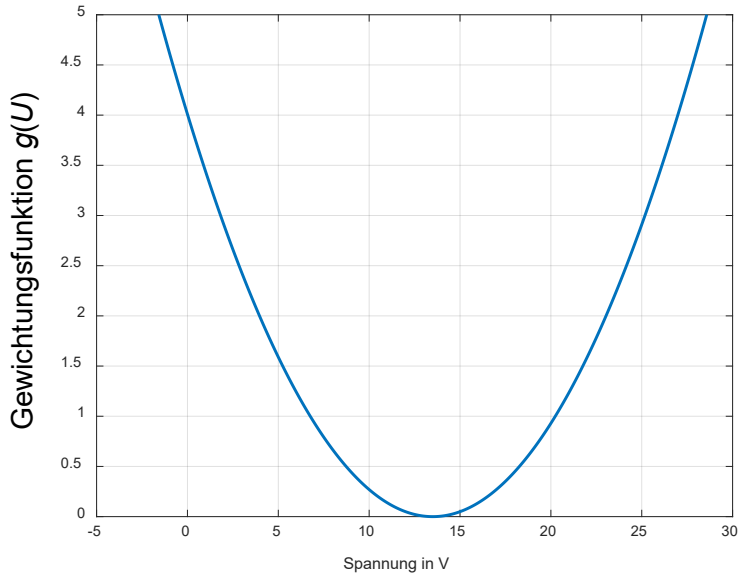
Fehler und Schaltvorgänge im Bordnetz können, wie bereits erläutert, Unter- und Überspannungen an den einzelnen Verbrauchern verursachen. Diese Abweichungen der nominalen Versorgungsspannung müssen nun in Ihrer Schwere beurteilt werden. Da hierfür kein einheitliches Verfahren existiert, sind verschiedene Herangehensweisen denkbar.

Eine mögliche Orientierung zur Bewertung einzelner Spannungsverläufe liefern einschlägige Normen zur Prüfung von Bordnetzkomponenten. Eine wichtige Rolle spielt hier die LV 124 [6] beziehungsweise daran angelehnte herstellerspezifische Standards (z. B. VW 80000 [7], BMW GS 95024-2-1). In diesen Normen sind Testpulse für Bordnetz-Verbraucher definiert, welche die Komponenten ohne Verlust ihrer Funktionalität aushalten müssen. Davon ausgehend können nun Pulse, welche über diese Anforderungen hinaus gehen, als kritisch bewertet werden. So schreibt die LV 124 für sicherheitskritische Verbraucher beispielsweise einen

statischen fehlerfreien Spannungsbereich von 9 – 16 V vor; Spannungen über 27 V werden nicht von den Tests abgedeckt und können generell als kritisch angesehen werden, ebenso wie Spannungseinbrüche unter 6 V für mehr als 100 µs.

Der Nachteil einer Bewertung anhand dieser Kriterien ist, dass diese nur eine binäre Entscheidung über die Spannungsstabilität ermöglichen. Um verschiedene Topologien miteinander vergleichen zu können, sind wertekontinuierliche Kriterien sinnvoller. In diesem Zusammenhang existieren in der Literatur verschiedene Ansätze zur Bewertung von Spannungsverläufen, sowohl im Kontext der Kfz-Energiebordnetze (u. a. in [8] und [9]) als auch im Kontext der Power Quality in der Energieversorgung (z. B. [10]).

Als ein mögliches Kriterium soll im Folgenden ein auf [8] basierender Ansatz verwendet werden. Die zugrundeliegende Idee ist, die einzelnen Spannungssamples der Messungen, bzw. Simulationen umso stärker zu gewichten, je stärker sie von der nominalen Spannung abweichen. Dies wird durch eine parabelförmige Gewichtungsfunktion realisiert, welche in Abbildung 17 dargestellt ist.



$$g(U) = 0{,}022 \cdot (U - 14\ \mathrm{V})^2$$

Abbildung 17: Funktion zur Gewichtung abweichender Spannungslevel

Die einzelnen gewichteten Spannungssamples werden schließlich zu einer gewichteten Summe zusammengefasst, welche der Bewertung des Spannungsverlaufs entspricht. Um die unterschiedlichen Sampledauern bei nicht äquidistanten Zeitschritten zu berücksichtigen, wird im Unterschied zu [8] die folgende Berechnungsvorschrift verwendet:

$$G = \sum_{i=1}^{n} g(U_i)\Delta t_i \tag{26}$$

Als weitere Möglichkeit der Bewertung soll die Spannungszeitfläche betrachtet werden, welche durch die Über- und Unterspannungen entsteht. Konkret sollen hier die Flächen berücksichtigt werden, welche über den zulässigen statischen Spannungsbereich hinausgehen; in Anlehnung an die LV 124 wird dieser mit 9 – 16 V angenommen. Abbildung 18 veranschaulicht dies an einem verallgemeinerten Überspannungspuls. In Anlehnung an die Literatur entspricht dies dem in der Regelungstechnik geläufigen Integral Absolute Error (IAE) [11].

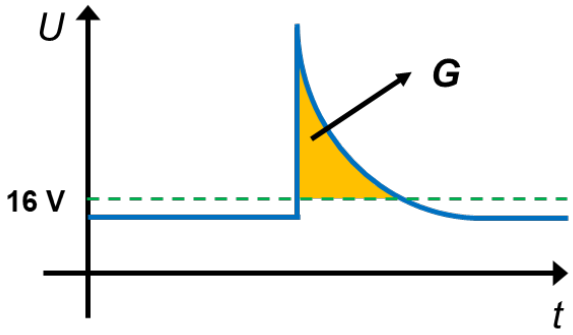


Abbildung 18: Berücksichtigung der Spannungszeitfläche zur Bewertung von Unter- und Überspannungspulsen

Als Ergänzung hierzu wird in diesem Zusammenhang ein weiteres Kriterium definiert, welches über das Quadrat der Über- bzw. Unterspannung integriert und damit einen Zusammenhang zur Fehlerenergie herstellt.

Die drei beschrieben Kriterien sind in den Funktionen *weight_function_evaluation.m*, *area_evaluation.m* und *energy_area_evaluation.m* implementiert (Unterordner *evaluation/criteria*).

4.2 Gesamtmetrik des Bordnetzes

Für die vielen simulierten Lastspannungsverläufe können mithilfe der Kriterien aus 4.1 einzelne Metriken berechnet werden. Diese müssen nun auf sinnvolle Art zu einer Gesamtmetrik zusammengesetzt werden, welche die Spannungsstabilität der untersuchten Topologie beschreibt. Da die Lasten unterschiedliche Sicherheitsanforderungen haben (ASIL D, QM, …), ist eine entsprechende Gewichtung sinnvoll; eine instabile Versorgung eines Komfortverbrauchers sollte nicht in gleichem Maße in die Gesamtbewertung eingehen wie die Störung eines sicherheitsrelevanten Steuergeräts. Darüber hinaus ist eine Gewichtung der untersuchten Fehler denkbar, wenn beispielsweise unterschiedliche Ausfallwahrscheinlichkeiten bekannt sind.

Um diese Aspekte zu berücksichtigen, wird die folgende gewichtete Summe zur Bildung einer Gesamtmetrik vorgeschlagen:

$$G_{\text{gesamt}} = \frac{1}{\sum K_{\text{Last},j} \cdot \sum K_{\text{Fehler},i}} \sum_{j}^{\text{Lasten}} \sum_{i}^{\text{Fehler}} G_{i,j} \cdot K_{\text{Last},j} \cdot K_{\text{Fehler},i} \tag{27}$$

Jede Einzelbewertung $G_{i,j}$ des Spannungsverlaufs der Last j bei Fehlerszenario i geht über die Faktoren $K_{\mathrm{Last},j}$ und $K_{\mathrm{Fehler},i}$ (jeweils zwischen 0 und 100 %) in die Gesamtsumme ein. Damit die Metrik unabhängig von der absoluten Anzahl der berücksichtigten Fehler und Lasten ist, wird anschließend zur Normierung jeweils durch die Summe aller Gewichtungsfaktoren geteilt.

5 Beispielhafte Anwendung

Die in den vorherigen Kapiteln entwickelten Methoden zur schnellen Vorselektion und Bewertung von spannungsstabilen Energiebordnetzen sollen abschließend an einem konkreten Beispiel aus dem Arbeitskreis untersucht werden. Abbildung 18 zeigt die hierfür verwendete 12 V-Ebene der automatisch generierten Topologie 3500. Diese stellt eine der vielen Topologievarianten dar, die auf Basis der allgemeinen im Arbeitskreis definierten einhüllenden Topologie vom Projektpartner Universität Kassel generiert wurde. Die durch den Workflow des Projektpartners Fraunhofer daraus erzeugte Topologiedefinition in Form einer Netzliste ist der Ausgangspunkt der folgenden Untersuchungen.

In der vorliegenden Topologie sind die vier PDUs mit mehreren redundanten Leitungen miteinander verbunden. Neben einer ringförmigen Struktur sind die PDUs 1 und 4 zusätzlich direkt miteinander verbunden. Die roten Leitungen kennzeichnen die (nicht schaltbaren) Backbone-Verbindungen. Alle grün markierten Ein- und Ausgänge der PDUs sind durch Sicherungen individuell abgesichert.

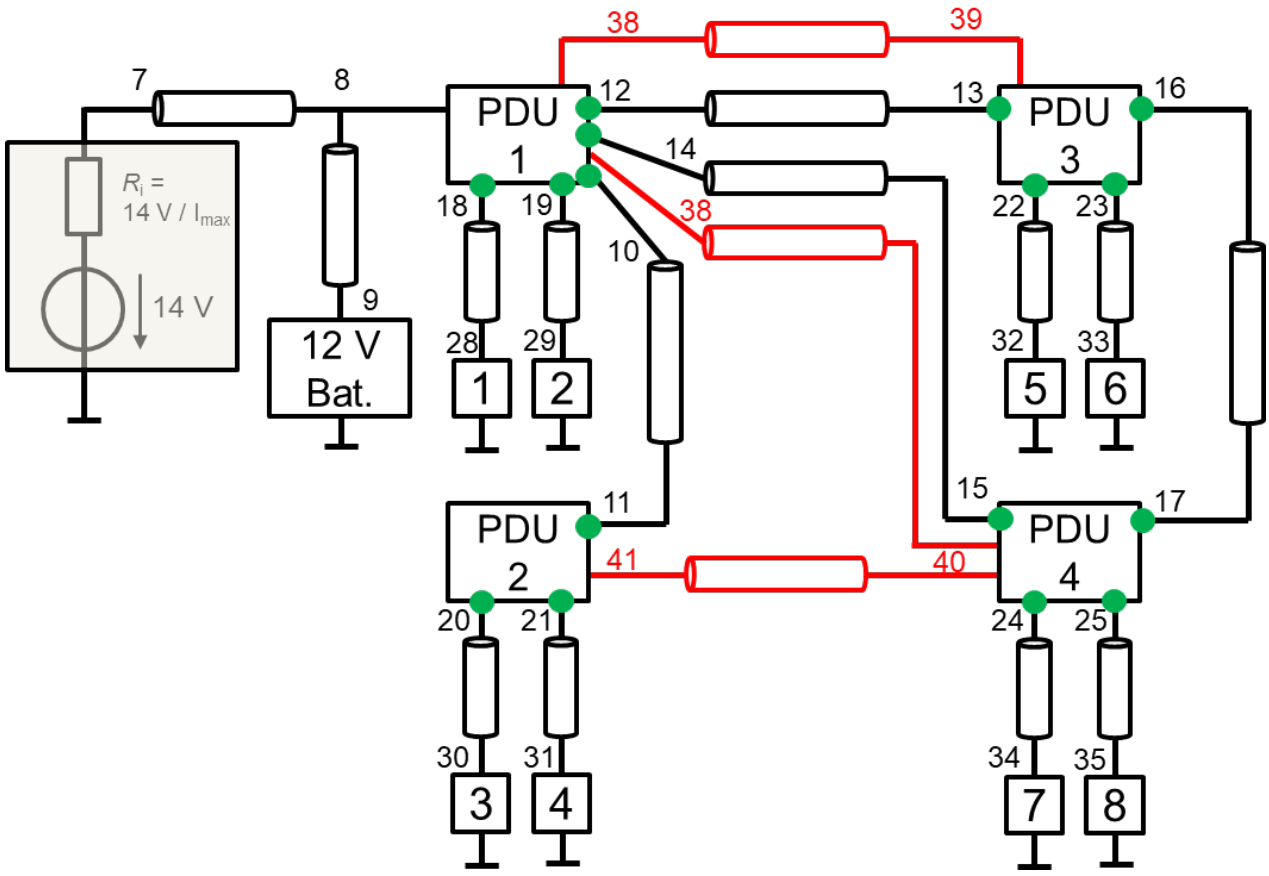


Abbildung 19: Schematische Darstellung der untersuchten Beispieltopologie (Topologie 3500, 12 V-Ebene). Rot: Backbone-Leitungen. Grün: Schaltbare / abgesicherte Verbindung

Entsprechend des entwickelten Verfahrens zur Vorselektion werden im ersten Schritt die Auswirkungen aller möglichen Leitungsfehler durch Rechteckpulse approximiert. Die Ergebnisse dieser Approximation sind im Folgenden für alle Lasten und alle Fehlerfälle dargestellt.

Abbildung 20 zeigt die approximierten Verläufe aller Kurzschlussfehler an den 23 Systemknoten. Analog sind in Abbildung 21 die aus den Leerlauffehler aller 17 Leitungen resultierenden Lastspannungen dargestellt.

Grundsätzlich ist zu erkennen, dass Kurzschlussfehler zu deutlich stärkeren Beeinträchtigungen der approximierten Spannungsverläufe führen. Die Überspannungspulse im Falle eines Leitungsbruchs erreichen Amplituden von weniger als 1 V über der Nominalspannung. Kritisch sind im Falle der Leerlauffehler lediglich die einzelnen Totalausfälle, die auftreten, wenn die unmittelbare Lastleitung unterbrochen wird. Kurzschlussfehler führen im Gegensatz dazu aufgrund der hohen Fehlerströme für größere zwischenzeitliche Spannungseinbrüche und durch das Abschalten dieser Ströme durch die Sicherung zu höheren Schaltpulsen.

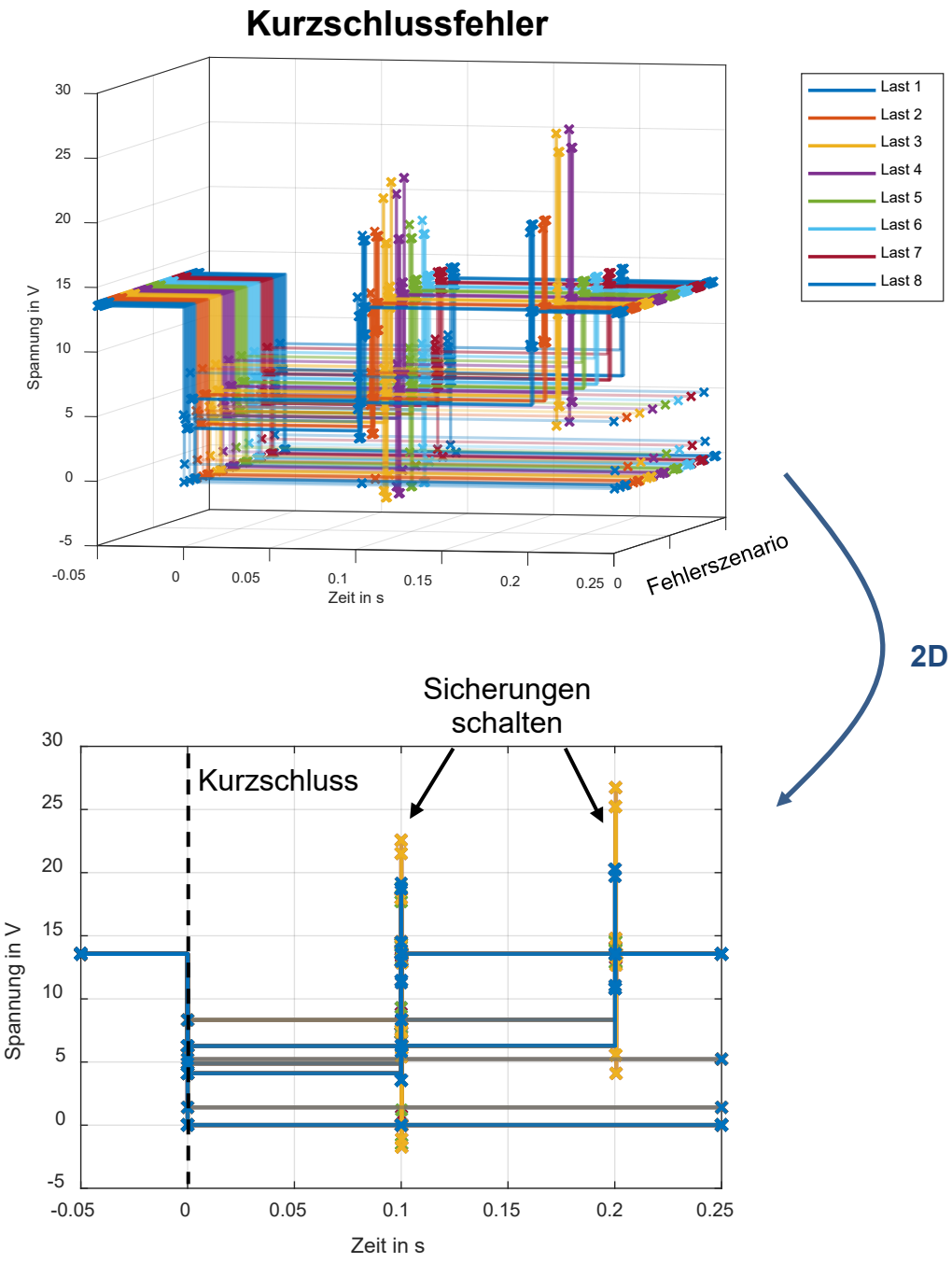


Abbildung 20: Approximierte Lastspannungen aller möglichen Kurzschlussfehler der untersuchten Topologie

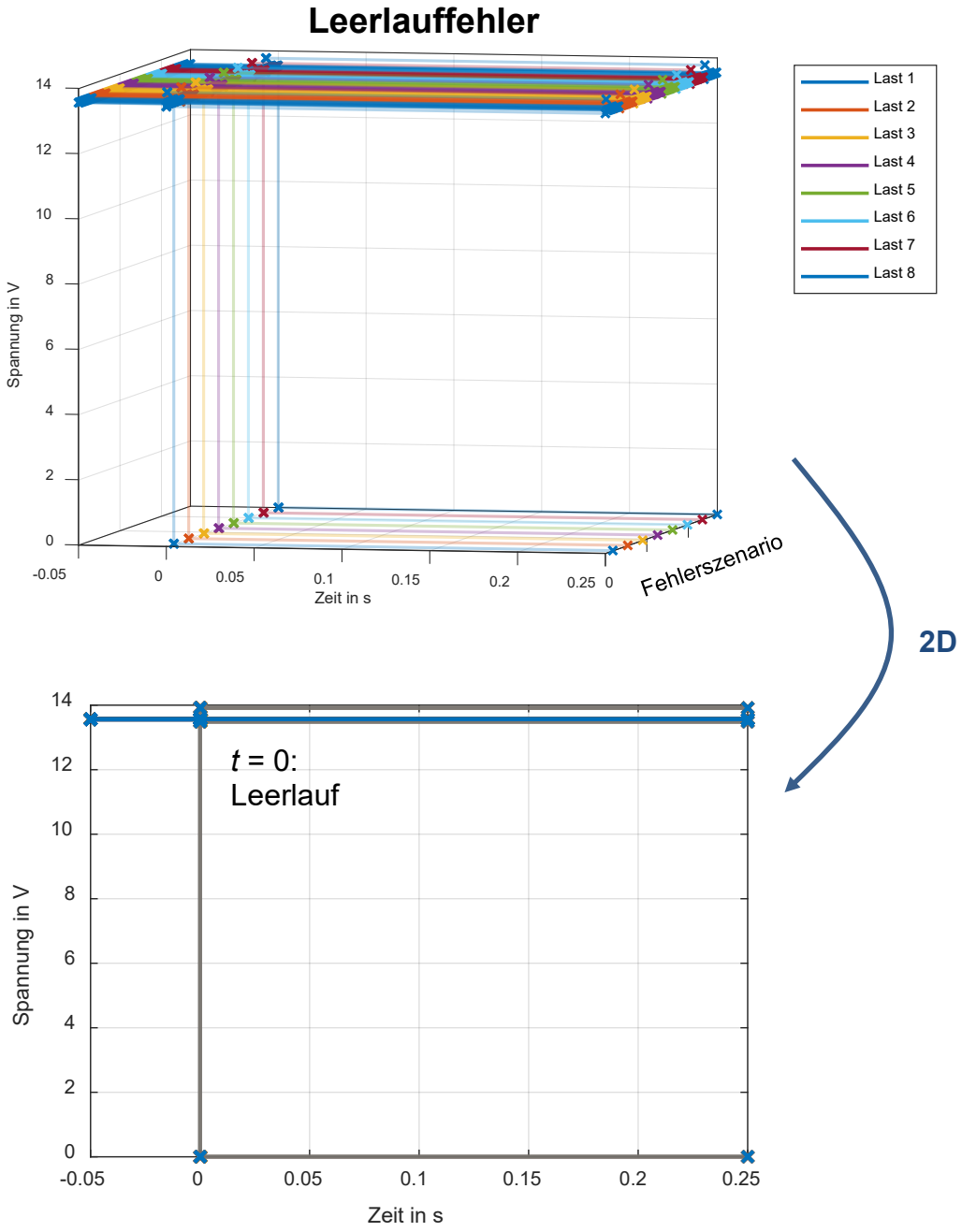


Abbildung 21: Approximierte Lastspannungen aller möglichen Leerlauffehler der untersuchten Topologie

Zur erneuten Validierung der Approximationsmethode wird ein ausgewähltes Fehlerszenario mit einer transienten LTspice-Simulation verglichen. Abbildung 22 zeigt die resultierenden Spannungen an Last 2 für den Schaltvorgang infolge eines Kurzschlussfehlers an Knoten 18. Wie zu erkennen ist, liefert die rechteckförmige Abschätzung einen deutlich größeren Spannungspeak als die genauere transiente Simulation. Dies entspricht der Forderung, mit der Approximation keine kritischen Fälle übersehen zu dürfen.

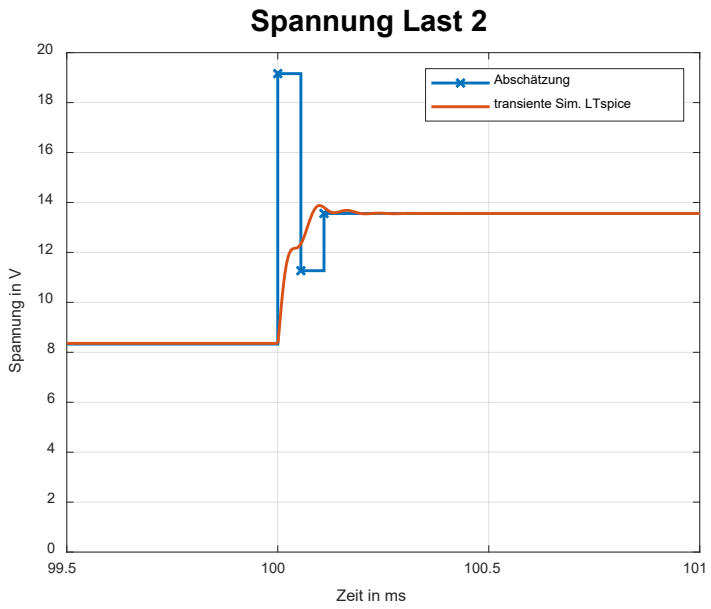


Abbildung 22: Vergleich der approximierten Spannung an Last 2 mit transienter LTspice-Simulation. Kurzschlussfehler an Knoten 18. Auslösen der Sicherung bei *t* = 100 ms

Im nächsten Schritt müssen auf Basis der Approximationsergebnisse diejenigen Szenarien ausgewählt werden, welche als potentiell kritisch angesehen werden und in der nächsten Stufe untersucht werden müssen. In diesem Beispiel wird hierfür die Last 5 als sicherheitsrelevant angenommen. Die hier angewendeten Kriterien zur Identifikation potentiell kritischer Szenarien orientieren sich an der LV 124 (siehe Abschnitt 4.1): Eine Überspannung über 27 V wird als kritisch angenommen, ebenso wie eine Unterspannung unter 6 V für mehr als 100 µs. Somit ergeben sich für Last 5 insgesamt elf Szenarien, die zu potentiell kritischen Verläufen führen (siehe Abbildung 23). Konkret sind dies Kurzschlüsse an den Knoten der Versorgungsleitungen der PDUs und den Quellen.

Diese Szenarien müssen nun in der nächsten Stufe der Vorselektion genauer untersucht werden, indem eine entsprechende transiente Schaltungssimulation der Fehlerfälle durchgeführt wird. Die daraus resultierenden simulierten Spannungsverläufe sind in Abbildung 24 dargestellt.

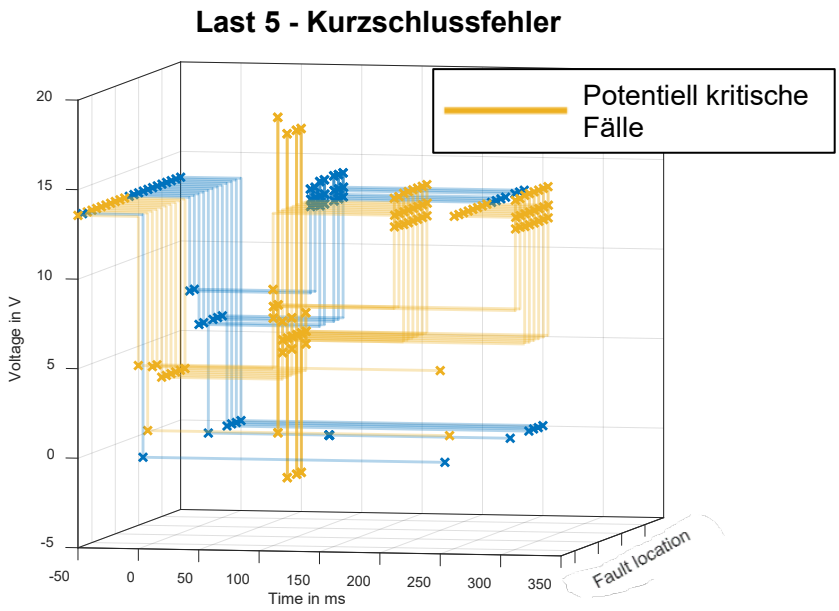


Abbildung 23: Approximierte Spannungsverläufe der Last 5 bei Kurzschlussfehlern. Gelb: Potentiell kritische Fälle

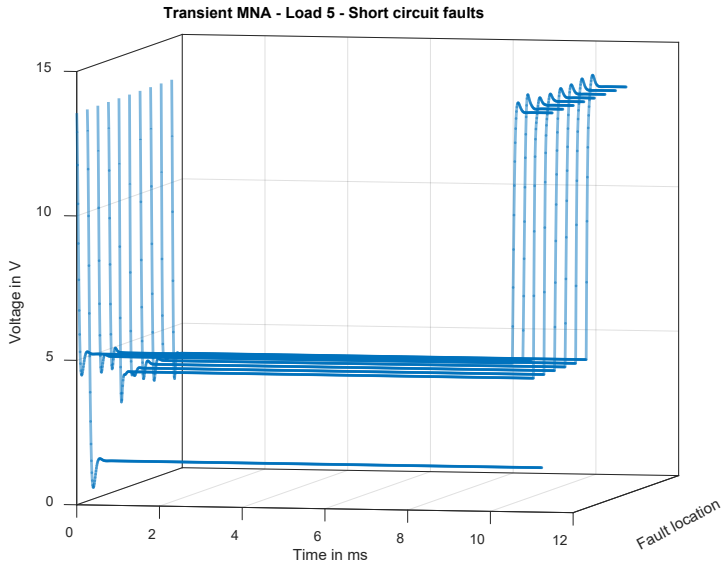


Abbildung 24: Spannungsverläufe der Last 5 der zweiten Stufe der Vorselektion (transiente Schaltungssimulation)

Nach Evaluation der Simulationsergebnisse der zweiten Vorselektionsstufe bleiben insgesamt 10 Fehlerszenarien übrig, die weiterhin als potentiell kritisch einzustufen sind. Für diese 10 Szenarien ist für eine abschließende Beurteilung eine genaue Modelica-Simulation notwendig. Das hierfür nötige JSON-Skript zur automatisierten Fehlersimulation wird von dem am Ende des Vorselektionsprozesses generiert (siehe 2.4).

Abbildung 25 fasst die durchlaufenen Stufen zusammen. Die anfänglich 40 Fehlerszenarien der untersuchten Topologie konnten durch die Approximation in der ersten Stufe auf 11 relevante Szenarien reduziert werden. Diese Abschätzung dauerte auf dem Testrechner (CPU: Intel Core i7-9700, 16 GB RAM) insgesamt etwa 2 s. Die transiente Schaltungssimulation

dauerte insgesamt 3,5 s. Die in Modelica zu untersuchenden Szenarien konnten somit von 40 auf 10 reduziert werden. Da eine OpenModelica-Simulation unter anderem aufgrund der notwendigen Kompilierung zum Teil deutlich über 10 s benötigt, ergibt sich in diesem Beispiel ein Performance-Gewinn der entwickelten Methode von etwa 75 %. Bereits bei einfachen Konfigurationen sind also erhebliche Einsparungen der Rechenzeit möglich. In realitätsnäheren und deutlich größeren Konfigurationen ist zu erwarten, dass das Einsparpotential noch deutlich höher ist.

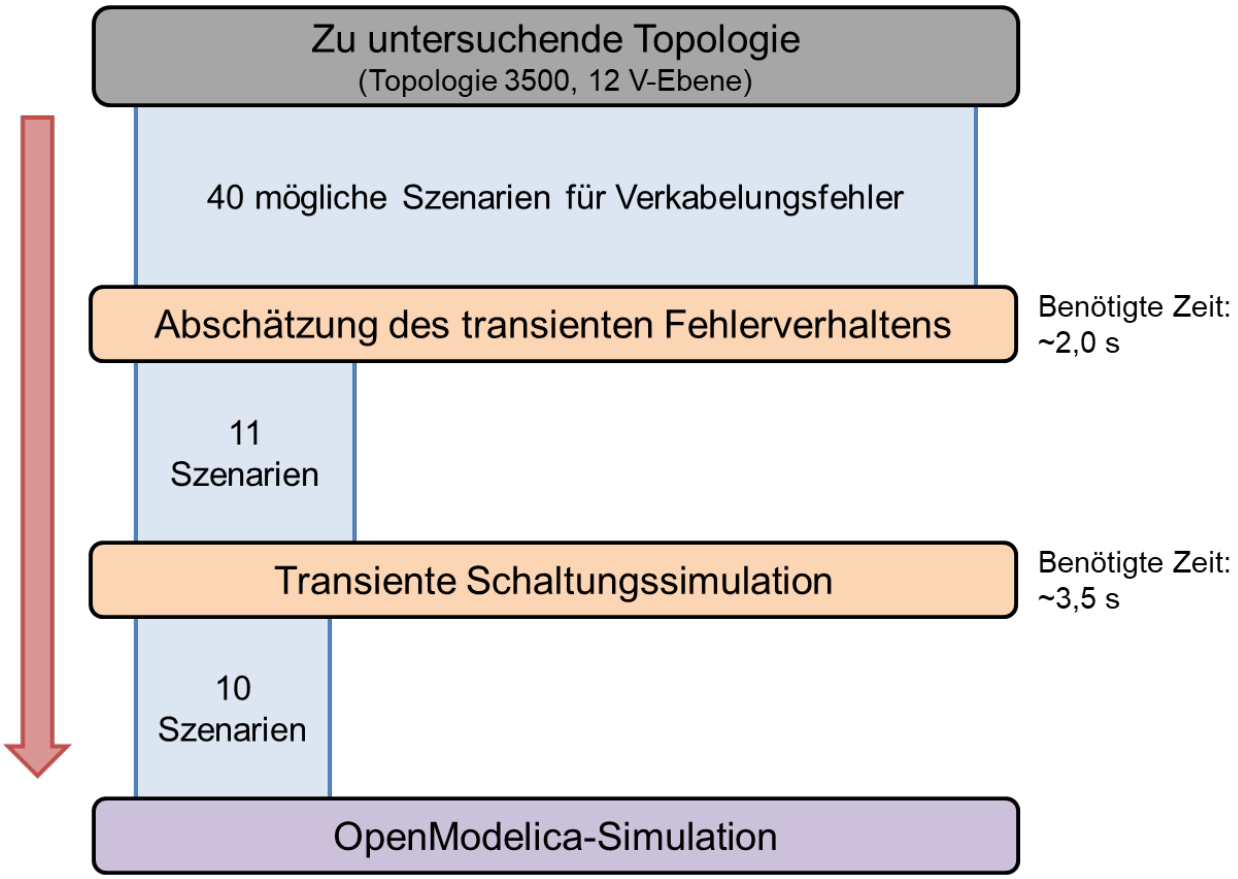


Abbildung 25: Anzahl analysierter Szenarien und benötigte Zeit in den einzelnen Stufen der Vorselektion

Die in Kapitel 4 beschriebene Bewertung der untersuchten Topologie anhand der simulierten Spannungsverläufe ist ebenfalls in den automatisierten Workflow integriert. Für dieses Beispiel wird zur Bildung der Gesamtmetrik ein Gewichtungsfaktor für sicherheitsrelevante Lasten (hier Last 5) von $K_{\mathrm{Last,SR}} = 1$ und für nicht-sicherheitsrelevante Lasten von $K_{\mathrm{Last,SR}} = 0{,}25$ gewählt. Für die einzelnen Fehlerszenarien wird eine gleichmäßige Gewichtung gewählt ($K_{\mathrm{Fehler}} = 1$), da keine Informationen über Ausfallwahrscheinlichkeiten vorliegen.

Die resultierende Gesamtbewertung der Topologie auf Basis der drei vorgestellten Bewertungsmethoden ist in Tabelle 2 dargestellt. Eine isolierte Interpretation dieser Ergebnisse ist jedoch nur bedingt möglich. Sie ermöglicht jedoch den Vergleich zwischen verschiedenen Topologien und ist damit eine Methode zur Auswahl möglichst hochverfügbarer Energiebordnetze.

Tabelle 2: Aus den Simulationsergebnissen resultierende Topologiebewertungen

Bewertungskriterium	Gesamtmetrik der Topologie
Gewichtungsfunktion	0.04
Spannungszeitfläche	85,4 mVs
Quadrierte Spannungszeitfläche („Energie“)	716,7 mV²s

Um einen solchen Topologievergleich exemplarisch durchzuführen, werden aus der Ausgangstopologie zwei neue Varianten erzeugt, indem bestimmte redundante Leitungen entfernt werden (siehe Abbildung 26). In Variante 1 werden die beiden Verbindungen zwischen PDU 1 und PDU 3 entfernt. PDU 3 wird somit nur über PDU 4 versorgt. In Variante 2 wird PDU 3 lediglich über die Backbone-Leitung von PDU 1 versorgt.

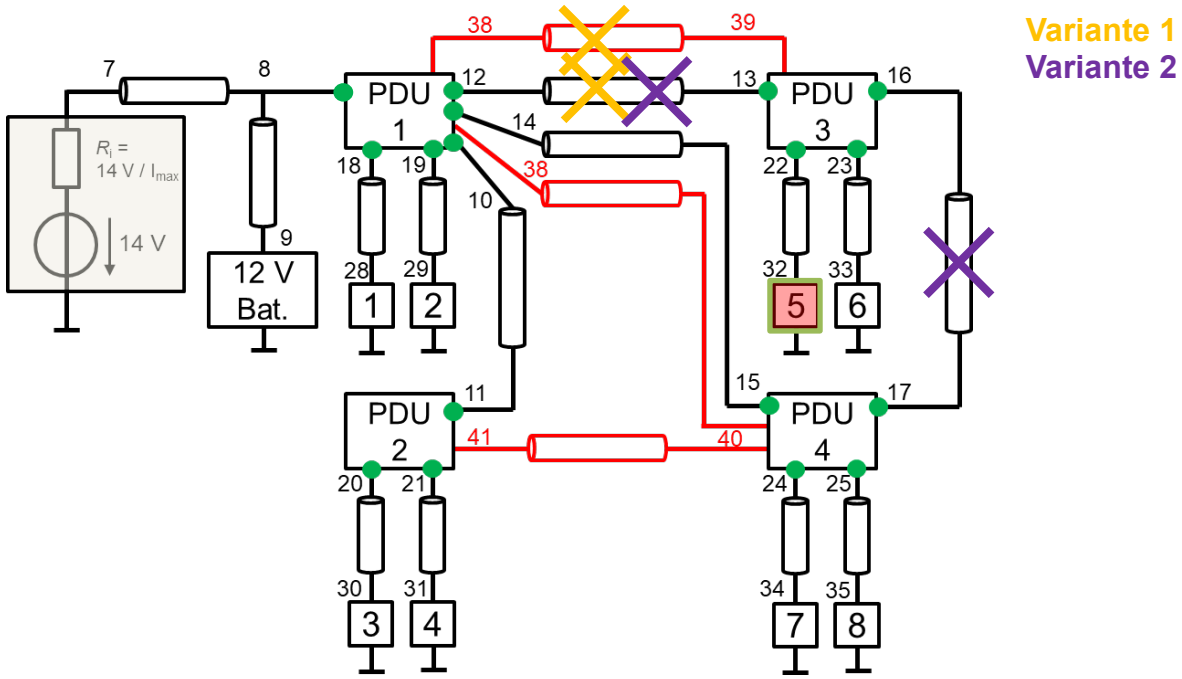


Abbildung 26: Variation der untersuchten Topologie durch Entfernung redundanter Leitungen

Die beiden Varianten werden analog zur Ausgangstopologie mit dem automatisierten Workflow simuliert und bewertet. Die resultierenden Metriken aller Topologien sind in Tabelle 3 im Vergleich dargestellt.

Tabelle 3: Gesamtmetriken der modifizierten Topologien im Vergleich

Verwendete Einzelmetrik	Ausgangsvariante G_{gesamt}	Variante 1 G_{gesamt}	Variante 2 G_{gesamt}
Gewichtungsfunktion	0,04	0,04	0,036
Spannungszeitfläche	85,4 mVs	85 mVs	75 mVs

„Energie“	716,7 mV²s	721,4 mV²s	623,78 mV²s

Auffällig ist, dass Variante 1 mit allen drei Bewertungsmethoden eine fast identische Gesamtbewertung wie die Ausgangstopologie besitzt. Variante 2 besitzt sogar eine um etwa 10 % bessere Bewertung. Dies ist zunächst kontraintuitiv, da diese Varianten jeweils weniger Redundanz aufweisen und damit eine unzuverlässigere Versorgung insbesondere der sicherheitsrelevanten Last 5 erwartet werden könnte. Eine genauere Untersuchung der Simulationsergebnisse zeigt die Ursache für dieses Verhalten; in der bisherigen Topologie sind die Sicherungsparameter der PDU-Versorgungsleitungen alle identisch parametriert. Dadurch löst die Hauptsicherung, welche über den Knoten 8 die Quellen mit dem restlichen Netz versorgt, im Falle eines Kurzschlusses entlang der Versorgungsleitungen immer zuerst aus. Infolge eines solchen Fehlers kommt es also immer zu einem Totalausfall des Systems; da durch zusätzliche Redundanz auch mehr potentielle Fehlerorte existieren, ist die redundante Ausgangstopologie daher sogar schlechter.

Eine solche Parametrierung ist allerdings nicht realistisch. Im Falle eines Fehlers sollte die Hauptsicherung als Letztes auslösen, damit ein Totalausfall möglichst vermieden werden kann. Daher wird die Auslösedauer dieser Sicherung entsprechend erhöht und die Untersuchung wiederholt. Die Ergebnisse der neuen Analyse sind in Tabelle 4 dargestellt. Die Ausgangstopologie besitzt nun eine signifikant bessere Bewertung als vorher. Wie ursprünglich zu erwarten, schneiden die weniger redundanten Varianten 1 und 2 nun tatsächlich schlechter ab. Variante 2, die weiterhin über die Backbone-Verbindung direkt mit PDU 1 verbunden ist besitzt nur eine geringfügig schlechtere Bewertung, während Variante 1 deutlich schlechter abschneidet. Dies ist plausibel, da Variante 1 nun nur noch über den längeren Weg über PDU 4 versorgt wird. Diese Beobachtung korreliert ebenfalls mit den in Tabelle 5 dargestellten logischen Ausfallwahrscheinlichkeiten der Topologien, welche von der Universität Kassel in Arbeitspaket 2 berechnet wurden.

Tabelle 4: Gesamtmetriken der modifizierten Topologien im Vergleich

Verwendete Einzelmetrik	**Ausgangsvariante** G_{gesamt}	**Variante 1** G_{gesamt}	**Variante 2** G_{gesamt}
Gewichtungsfunktion	0,024	0,032	0,025
Spannungszeitfläche	51,6 mVs	68,9 mVs	54 mVs
„Energie“	384,4 mV²s	547 mV²s	398,2 mV²s

Tabelle 5: Logische Ausfallwahrscheinlichkeiten der betrachteten Topologien (Universität Kassel)

	Ausgangstoplogie (Top.-Nr. 3500)	**Variante 1 (Top.-Nr. 2452)**	**Variante 2 (Top.-Nr. 2540)**
Wahrscheinlichkeiten des Ausfalls der Versorgung der 4 PDUs (1-2-3-4)	10^{-9} 10^{-9} 10^{-9} 10^{-9}	10^{-9} 10^{-9} $1.002 \cdot 10^{-6}$ 10^{-9}	10^{-9} 10^{-9} $4 \cdot 10^{-9}$ 10^{-9}
Summe	$\mathbf{4 \cdot 10^{-9}}$	$\mathbf{1.005 \cdot 10^{-6}}$	$\mathbf{8 \cdot 10^{-9}}$

6 Zusammenfassung

Im Rahmen des hier dargestellten Arbeitspakets wurde eine Methode entwickelt, um die Fehlertoleranz von Energiebordnetzarchitekturen schnell zu analysieren. Dies ermöglicht es, eine große Anzahl an Topologien effizient miteinander zu vergleichen und die vielversprechendsten auszuwählen. Die Topologiedefinitionen, welche vom Projektpartner Universität Kassel aus der definierten Grundkonfiguration generiert werden können, werden durch den am Fraunhofer IIS/EAS entwickelten Workflow in ein zur Simulation geeignetes Format konvertiert und können mit der hier vorgestellten Methodik untersucht werden.

Um die Dauer der notwendigen Fehlersimulationen zu reduzieren, wurde in der entwickelten Methode ein mehrstufiger Vorselektionsprozess implementiert. In der ersten Stufe werden zunächst alle möglichen Leitungsfehler (Kurzschluss und Leitungsbruch), die in der Topologie auftreten können, durch Lösen von einfachen linearen Gleichungssystemen und Auswertung analytischer Ausdrücke approximiert. Hierdurch können die im System resultierenden Über- und Unterspannungen mithilfe von Worstcase-Abschätzungen nach oben und unten abgeschätzt werden. Nach dieser ersten Abschätzung kann bereits ein Teil der Szenarien als unkritisch identifiziert werden und muss nicht weiter betrachtet werden. In der zweiten Stufe werden die Szenarien, die potentiell kritisch sein könnten, in einer schnellen transienten Schaltungssimulation näher untersucht. Lediglich die Szenarien, die nach dieser Vorselektion weiterhin als potentiell kritisch eingestuft werden, müssen anschließend mit sehr genauen, nichtlinearen Modellen final simuliert werden. Hierdurch ist es möglich, dass auch noch große Topologien mit einer Vielzahl an möglichen Fehlerorten und Fehlerarten unter sucht werden können. Die Gültigkeit der entwickelten Approximationen wurde in einer messtechnischen Validierung an einem beispielhaften Bordnetz im Laborprüfstand gezeigt.

Zur Auswertung der simulierten Bordnetztopologien wurde außerdem eine Bewertungsmethodik entwickelt und in den automatisierten Workflow integriert. Die einzelnen aus der Simulation resultierenden Lastspannungsverläufe werden dabei zunächst einzeln bewertet. Hierfür wurden in Anlehnung an existierende Literaturmethoden, drei konkrete Verfahren ausgewählt und erweitert. Die vielen Einzelbewertungen der jeweiligen Lasten und Fehlerfälle werden anschließend zu einer Gesamtmetrik des Energiebordnetzes zusammengefasst, welche die Fehlertoleranz der gesamten Bordnetztopologie widerspiegeln kann und Energiebordnetzauslegungen untereinander vergleichbar macht.

In einer abschließenden Untersuchung wurde der entwickelte, automatisierte Prozess beispielhaft erfolgreich anhand einer konkreten Topologie durchgeführt, die vom Projektpartner Fraunhofer zur Verfügung gestellt wurde und auf der Topologieerzeugung der Uni Kassel basiert. Auf Basis der berechneten Gesamtmetrik wurde ein exemplarischer Topologievergleich durchgeführt.

Literaturverzeichnis

[1] „The Modelica Association," [Online]. Available: https://www.modelica.org/. [Zugriff am 05 08 2022].

[2] FAT 305: Simulationsgestützte Methodik zum Entwurf intelligenter Energiesteuerung in zukünftigen Kfz-Bordnetzen, Berlin: Forschungsvereinigung Automobiltechnik E.V., 2018.

[3] FAT 334: Simulationsgestützte Analyse und Bewertung der Fehlertoleranz von Kfz-Bordnetzen, Berlin: Forschungsvereinigung Automobiltechnik E.V., 2020.

[4] „OpenModelica," [Online]. Available: https://www.openmodelica.org/. [Zugriff am 05 08 2022].

[5] J. Vlach und K. Singhal, Computer Methods for circuit analysis and design, New York: Van Nostrand-Reinhold, 1983.

[6] LV124, Elektrische und elektronische Komponenten in Kraftfahrzeugen bis 3,5t - Allgemeine Anforderungen, Prüfbedingungen und Prüfungen Teil 1 - Elektrische Anforderungen und Prüfungen 12 V Bordnetz.

[7] Volkswagen, VW80000 - Electric and Electronic Components in Motor Vehicles up to 3,5t - General Requirements, Test Conditions and Tests, 2013.

[8] L. Brabetz, M. Ayeb, G. Jilwan, P. Graebel und T. Kerner, „A New Approach to the Test, Assessment and Optimization of Robust Electrical Distribution Systems," in *SAE*, Detroit, 2012.

[9] J. Klötzl, Stabilität automobiler Leistungsbordnetze, München: Shaker Verlag, 2012.

[10] M. Magro, A. Mariscotti und P. Pinceti, „Definition of Power Quality Indices for DC Low Voltage Distribution Networks," in *IEEE Instrumentation and Measurement Technology Conference Proceedings*, 2006.

[11] P. Domanski, Control Performance Assessment: Theoretical Analyses and Industrial Practice, Springer, 2020.

7 Anhang

7.1 Quellcode

Im Folgenden ist der Quellcode der wichtigsten Matlab/Octave-Skripte zu finden.

7.1.1 topology_evaluation.m

```
clear all
close all
clc

%% evaulation parameters
netlist_path = 'examples\ak30_cir_release_with_load_wires_12V.cir';    % topology netlist path

sr_loads = [5];  % safety relevant loads
plot_results = 1;   % 1: plot results, 0: don't plot
% evaluation criteria
upper_limits = [27 1e-6];
lower_limits = [6, 100e-6];
% transient simulation parameters
simulation_time = 10e-3;
step_with = 2e-6;

%% add necessary paths
addpath(genpath('functions'))
rmpath('functions\_archiv\')
addpath(genpath('..\___Networkanalysis Ver6.2'))

%% preselection
disp(['***STAGE 1: APPROXIMATION***']);
tic
% approximate fault behavior
approximation_results = approximate_fault_behavior(netlist_path);
% plot calculated voltages of loads of interest
if(plot_results==1)
    plot_approx_load_voltages(approximation_results, sr_loads);
end

% evaluate voltage waveforms
[total_failures_short, total_failures_open, pot_critical_short, pot_critical_open] = evalu-
ate_approximation(approximation_results, sr_loads, upper_limits, lower_limits);
disp(['Fault scenarios evaluated: ' num2str(length(approximation_results.system_nodes) +
length(approximation_results.wire_IDs))]);
toc

%% transient MNA simulation where necessary
disp([newline '***STAGE 2: TRANSIENT MNA***']);
tic
transient_MNA_results = fault_behavior_transient_MNA (netlist_path, pot_critical_open, pot_crit-
ical_short, approximation_results, simulation_time, step_with);
% plot calculated voltages of loads of interest
if(plot_results==1)
    plot_transientMNA_load_voltages(transient_MNA_results, sr_loads);
end

% evaluate voltage waveforms
[total_failures_short, total_failures_open, pot_critical_short, pot_critical_open] = evalu-
ate_approximation(approximation_results, sr_loads, upper_limits, lower_limits);
disp(['Fault scenarios evaluated: ' num2str(size(transient_MNA_results.voltages_short, 2) +
size(transient_MNA_results.voltages_open,2))]);
toc
```

```
weight_metric = calculate_metric(approximation_results, transient_MNA_results, sr_loads, 1);
area_metric = calculate_metric(approximation_results, transient_MNA_results, sr_loads, 2);
energy_area_metric = calculate_metric(approximation_results, transient_MNA_results, sr_loads,
3);

disp([newline 'Weight metric: ' num2str(weight_metric)]);
disp([newline 'Area metric: ' num2str(area_metric/1e-3) ' mVs']);
disp([newline 'Energy metric: ' num2str(energy_area_metric/1e-3) ' mV²s']);

%% Modelica necessary?
disp([newline 'Remaining scenarios to investigate in Modelica: ' num2str(length(pot_criti-
cal_short) + length(pot_critical_open))]);

create_JSON_faultinjection('examples\ak30_cir_release_with_load_wires.txt', pot_critical_short,
pot_critical_open);
```

7.1.2 functions/approximation/approximate_fault_behavior.m

```
function results = approximate_fault_behavior(netlist_path)
    %% load and convert netlist
    netlist_raw = fileread(netlist_path);
    netlist_raw = strsplit(netlist_raw, {char(10) char(13)});
    netlist = cell(length(netlist_raw), 6);
    idx_delete = [];
    for i=1:length(netlist)
        new_entry = Netlist.splitLine(netlist_raw{i});
        if(~isempty(new_entry))
            netlist(i,1:length(new_entry)) = new_entry;
        else
            idx_delete = [idx_delete i];
        end
    end
    netlist(idx_delete, :) = [];


    %% extract supply system nodes, load nodes and wire numbers from netlist
    [results.system_nodes, results.load_nodes, results.load_IDs, results.wire_IDs] =
get_nodes_and_wire_info(netlist);

    %% evaluate open circuit faults
    [results.voltages_open, results.times_open] = open_circuit_routine(netlist, re-
sults.wire_IDs, results.load_nodes);

    %% evaluate short circuit faults
    [results.voltages_nominal, results.voltages_short, results.times_short] = short_circuit_rou-
tine(netlist, results.system_nodes, results.load_nodes);
end
```

7.1.3 functions/approximation/open_circuit_routine.m

```
%% Determine open circuit behavior for all possible fault locations
% voltages_open:mxn cell of approximated open circuit voltage arrays
%               (m: load numbers, n: fault locations)
% times_open:   corresponding mxn cell of time vectors
%               (m: load numbers, n: fault locations)
%
% netlist:      cell-array of netlist file
% fault_nodes:  nodes at which short faults are to be evaluated
% load_nodes:   nodes of loads
```

```
function [voltages_open, times_open] = open_circuit_routine(netlist, fault_wires, load_nodes)

    % calculate nominal state
    list_dc = generate_dc_netlist(netlist);
    [results_nominal, ~, ~] = networkanalysis(list_dc);
    [fuse_label, fuse_type, fuse_param, fuse_node] = get_fuse_infos(netlist);

    % determine wire currents and wire nodes
    [current_inductance_before, starting_nodes_before, ending_nodes_before] = get_wire_cur-
rents(results_nominal, list_dc);

    % determine system nodes for inductances and loads
    [inductance_incidence, load_incidence, inductance_values, load_values, load_inductances,
load_wires, wire_resistances] = map_ind_loads_to_system_nodes(netlist);


    voltages_open = cell(0,0);
    times_open = cell(0,0);

    for i=1:length(fault_wires)
        [maxVoltages, minVoltages, peakWidths, staticVoltages, triggered_fuse, fusing_time,
fault_netlist] = calculate_open_circuit(netlist, fault_wires(i), load_wires, load_values,
load_nodes, list_dc, results_nominal, ...
            inductance_incidence, current_inductance_before, inductance_values, start-
ing_nodes_before, ending_nodes_before, load_incidence, wire_resistances, load_inductances, ...
            fuse_label, fuse_type, fuse_param, fuse_node);

        if(isempty(peakWidths))
            peakWidths = zeros(length(load_nodes), 1);
        end
        % concatenate calculated voltages and times to one voltage array and
        % one time array representing the sample points for each load and fault
        % case
        voltages_open(:, i) = num2cell(maxVoltages);
        voltages_open(:, i) = num2cell([cell2mat(voltages_open(:,i)) minVoltages], 2);
        voltages_open(:, i) = num2cell([cell2mat(voltages_open(:,i)) staticVoltages], 2);

        times_open(:, i) = num2cell(zeros(length(load_nodes), 1));
        times_open(:, i) = num2cell([cell2mat(times_open(:,i)) peakWidths], 2);
        times_open(:, i) = num2cell([cell2mat(times_open(:,i)) 2*peakWidths], 2);

        while(~isnan(triggered_fuse))
            [maxVoltages, minVoltages, peakWidths, staticVoltages, next_triggered_fuse,
next_fusing_time, fault_netlist] = calculate_open_circuit(fault_netlist, triggered_fuse(1),
load_wires, load_values, load_nodes, list_dc, results_nominal, ...
                inductance_incidence, current_inductance_before, inductance_values, start-
ing_nodes_before, ending_nodes_before, load_incidence, wire_resistances, load_inductances, ...
                fuse_label, fuse_type, fuse_param, fuse_node);

            if(isempty(peakWidths))
                peakWidths = zeros(length(load_nodes), 1);
            end
            % concatenate calculated voltages and times to one voltage array and
            % one time array representing the sample points for each load and fault
            % case
            voltages_open(:, i) = num2cell([cell2mat(voltages_open(:,i)) maxVoltages], 2);
            voltages_open(:, i) = num2cell([cell2mat(voltages_open(:,i)) minVoltages], 2);
            voltages_open(:, i) = num2cell([cell2mat(voltages_open(:,i)) staticVoltages], 2);

            prev_times = cell2mat(times_open(:, 1));
            next_triggering_time = prev_times(:,end)+fusing_time;

            times_open(:, i) = num2cell([cell2mat(times_open(:,i)) zeros(length(load_nodes), 1)
+ next_triggering_time], 2);
```

```
            times_open(:, i) = num2cell([cell2mat(times_open(:,i)) peakWidths + next_trigger-
ing_time], 2);
            times_open(:, i) = num2cell([cell2mat(times_open(:,i)) 2*peakWidths + next_trigger-
ing_time], 2);

            triggered_fuse = next_triggered_fuse;
            fusing_time = next_fusing_time;
        end


    end

end
```

7.1.4 functions/approximation/calculate_open_circuit.m

```
function [maxVoltages, minVoltages, peakWidths, staticVoltages, triggered_fuse, fusing_time,
fault_netlist] = calculate_open_circuit(netlist, fault_wire, load_wires, load_values,
load_nodes, list_dc, results_nominal, inductance_incidence, current_inductance_before, induct-
ance_values, starting_nodes_before, ending_nodes_before, load_incidence, wire_resistances,
load_inductances, fuse_label, fuse_type, fuse_param, fuse_node)
        maxVoltages = [];
        minVoltages = [];
        staticVoltages = [];
        peakWidths = [];
        % inject fault and perform static simulation
        list_open = inject_open_circuit(netlist, fault_wire, 10e6);
        list_open_dc = generate_dc_netlist(list_open);
        [results, ~, ~] = networkanalysis(list_open_dc);
        fault_netlist = list_open_dc;

        % wire currents and wire nodes
        [current_inductance_after, starting_nodes_after, ending_nodes_after] = get_wire_cur-
rents(results, list_open_dc);

        % find relevant nodes for approximation
        relevant_nodes = unique(starting_nodes_after(load_wires));

        % Determine node with largest supply current sum in the system (for
        % worst case approximation)
        max_node = [];  % node with largest overall supply current flowing into node
        max_node_current = 0;  % largest overall supply current flowing into node
        for i=1:length(relevant_nodes)
            node_inductances = find(inductance_incidence(relevant_nodes(i),:));

            I_before = current_inductance_before(node_inductances);
            % determine sign (positive: into node; negative: from node)
            I_before(starting_nodes_after(node_inductances)==relevant_nodes(i)) = I_be-
fore(starting_nodes_after(node_inductances)==relevant_nodes(i))*-1;
            if(sum(I_before(I_before>0)) > max_node_current)
               max_node = relevant_nodes(i);
               max_node_current = sum(I_before(I_before>0));
            end
        end

        % variable to store peak width of iteration
        peakWidths_iteration = zeros(length(load_values), 1);

        % compare current into node with nominal
        worst_case = 1; % use worst case approximation for each load node
        for j=1:length(relevant_nodes)
            V_peak_max_load = [];
            V_peak_min_load = [];
```

```
            if(current_flow_into_node(list_open_dc, results, relevant_nodes(j)) ~= cur-
rent_flow_into_node(list_dc, results_nominal, relevant_nodes(j)))

                % für relevant_node: beiteiligte Induktivitäten (incidence_mat)
                relevant_inductances = find(inductance_incidence(relevant_nodes(j),:));

                switched_off_indunctance = find(abs(current_inductance_after(relevant_induct-
ances))<1e-4);
                if(~isempty(switched_off_indunctance))
                    temp = relevant_inductances~=relevant_induct-
ances(switched_off_indunctance)';
                    relevant_inductances = relevant_inductances(logical(prod(temp, 1)));
                end
                % If all node inductances have 0 current than all got switched
                % off --> no peak
                if(sum(current_inductance_after(relevant_inductances)) == 0)
                    continue;
                end

                % currents of relevant node inductances
                I_before = current_inductance_before(relevant_inductances);
                % determine sign (positive: into node; negative: from node)
                I_before(starting_nodes_after(relevant_inductances)==relevant_nodes(j)) = I_be-
fore(starting_nodes_after(relevant_inductances)==relevant_nodes(j))*-1;

                % approximate inductance of complete supply paths
                supply_inductances_values = approximate_supply_paths_inductances(rele-
vant_nodes(j), inductance_incidence, inductance_values, relevant_inductances, current_induct-
ance_before, starting_nodes_before, ending_nodes_before);

                % Discard load wires that supply a subnet by setting their previous current to 0
(worst case approx.: closest loads see whole current peak)
                if(worst_case)
                    direct_load_wires = sum(relevant_inductances == load_wires, 1);
                    I_before(and(~direct_load_wires, I_before<0)) = 0;
                end

                num_supply_path = sum(I_before>0);
                num_load_path = sum(I_before<0);

                % calculate equivalent parallel inductances for supply and load paths
                supply_indunctances_parallel = 1/(sum(1./supply_inductances_values));
                load_indunctances_parallel = 1/(sum((I_before<0).*(1./(inductance_values(rele-
vant_inductances)))));


                % Wenn nicht max_node: Worst case: I_before(größte Ind) = max_node_current
               if(worst_case && relevant_nodes(j)~=max_node)
                    supply_inductances = relevant_inductances(I_before>0);
                    if(~isempty(supply_inductances))
                        [~, idx_largest_inductance] = max(supply_inductances_values);
                        I_before(relevant_inductances == supply_inductances(idx_largest_induct-
ance)) = max_node_current;
                    end
               end

                % A*x = b
                % x: vector of supply and load currents during peak
                % [Iv1,peak Iv2,peak ... Il1,peak Il2,peak ...]
                if(num_supply_path ~= 0)
                    A=[];
                    b=[];

                    A(1, :) = repmat(supply_indunctances_parallel, 1, length(I_before)) .*
(I_before>0) + repmat(load_indunctances_parallel, 1, length(I_before)) .* (I_before<0);
```

```
                    A(2,:) = ones(1, length(I_before)).*sign(I_before);

                    b(1) = sum((repmat(supply_indunctances_parallel, 1, length(I_before)) .*
(I_before>0) + repmat(load_indunctances_parallel, 1, length(I_before)) .* (I_before<0)) .*
abs(I_before));
                    b(2) = 0;

                    if(num_supply_path>1)
                        idx = find(I_before>0);
                        for k=1:num_supply_path-1
                            A(end+1,idx(k)) = inductance_values(relevant_inductances(idx(k)));
                            A(end,idx(k+1)) = -inductance_values(relevant_induct-
ances(idx(k+1)));
                            b(size(A, 1)) = inductance_values(relevant_induct-
ances(idx(k)))*abs(I_before(idx(k))) - inductance_values(relevant_induct-
ances(idx(k+1)))*abs(I_before(idx(k+1)));
                        end
                    end
                    if(num_load_path>1)
                        idx = find(I_before<0);
                        for k=1:num_load_path-1
                            A(end+1,idx(k)) = inductance_values(relevant_inductances(idx(k)));
                            A(end,idx(k+1)) = -inductance_values(relevant_induct-
ances(idx(k+1)));

                            b(size(A, 1)) = inductance_values(relevant_induct-
ances(idx(k)))*abs(I_before(idx(k))) - inductance_values(relevant_induct-
ances(idx(k+1)))*abs(I_before(idx(k+1)));
                        end
                    end

                    % I_before und I_peak im LGS positiv
                    I_peak = A\b';
                else
                    I_peak = zeros(length(relevant_inductances), 1);
                end
                % Match inductance currents to loads
                node_currents = sum(inductance_incidence(:, relevant_inductances) .* I_peak',
2);
                I_peak_load = sum(load_incidence .* node_currents);


                % Indices of purely resistive and RC loads
                idx_R_loads = find(load_values(:,2)==0);
                idx_RC_loads = find(load_values(:,2)~=0);

                %% Determine voltage peaks at resistive loads
                V_peak_max_load = zeros(length(load_values), 1);
                V_peak_min_load = zeros(length(load_values), 1);
                temp = load_values(:, 1).*I_peak_load';
                V_peak_max_load(idx_R_loads) = temp(idx_R_loads);
                V_peak_min_load(idx_R_loads) = temp(idx_R_loads);


                %% Approximate voltage peak at RC loads
                % Worst Case Approximation: Assumtion of sum load currents
                I_peak_load(I_peak_load~=0) = sum(I_peak_load);

                % Approx. load supply inductance
                load_supply_nodes = starting_nodes_after(load_wires);
                for k=1:length(load_values)
                    % all inductances connected to node
                    node_inductances = [find(starting_nodes_after==load_supply_nodes(k)),
find(ending_nodes_after==load_supply_nodes(k))];
                    % delete inductances of load wire
```

```
                    for m=1:length(load_values)
                        node_inductances = node_inductances(node_inductances~=load_wires(m));
                    end

                    % worst-case approximation: assume minimum supply wire resistance
                    supply_inductance(k) = max(approximate_supply_paths_inductances(load_sup-
ply_nodes(k), inductance_incidence, inductance_values, node_inductances, current_inductance_be-
fore, starting_nodes_before, ending_nodes_before));
                    supply_resistance(k) = min(wire_resistances(node_inductances));
                end
                % equivalent circuit elements
                L = (load_inductances + supply_inductance)';
                R_load = load_values(:, 1);
                C = load_values(:, 2);
                R_wire = (wire_resistances(load_wires) + supply_resistance)';
%% RLC solution without wire resistances
                % Radicand of ODE solution
                q = -
2.*C(idx_RC_loads).*L(idx_RC_loads).*R_wire(idx_RC_loads).*R_load(idx_RC_loads) - ...
                        4.*C(idx_RC_loads).*L(idx_RC_loads).*R_load(idx_RC_loads).^2 +
C(idx_RC_loads).^2.*R_wire(idx_RC_loads).^2.*R_load(idx_RC_loads).^2 + L(idx_RC_loads).^2;
                % first positive and negative peak of dampened oscillation
                A =
sqrt(2*C(idx_RC_loads).*L(idx_RC_loads).*R_wire(idx_RC_loads).*R_load(idx_RC_loads) +
4*C(idx_RC_loads).*L(idx_RC_loads).*R_load(idx_RC_loads).^2 -
C(idx_RC_loads).^2.*R_wire(idx_RC_loads).^2.*R_load(idx_RC_loads).^2 - L(idx_RC_loads).^2) ./
(2*C(idx_RC_loads).*L(idx_RC_loads).*R_load(idx_RC_loads));
                B = (L(idx_RC_loads) +
C(idx_RC_loads).*R_wire(idx_RC_loads).*R_load(idx_RC_loads)) ./
(2*C(idx_RC_loads).*L(idx_RC_loads).*R_load(idx_RC_loads));
                t_max = (atan(A./B))./A;
                t_min = (atan(A./B)+pi)./A;
                V_peak_max_load(idx_RC_loads) =
(2*I_peak_load(idx_RC_loads)'.*L(idx_RC_loads).*R_load(idx_RC_loads))./(sqrt(-q)) ...
                                        .*exp(-
t_max.*(L(idx_RC_loads)+C(idx_RC_loads).*R_wire(idx_RC_loads).*R_load(idx_RC_loads))./(2*C(idx_R
C_loads).*L(idx_RC_loads).*R_load(idx_RC_loads)));
                V_peak_min_load(idx_RC_loads) = -
(2*I_peak_load(idx_RC_loads)'.*L(idx_RC_loads).*R_load(idx_RC_loads))./(sqrt(-q)) ...
                                        .*exp(-
t_min.*(L(idx_RC_loads)+C(idx_RC_loads).*R_wire(idx_RC_loads).*R_load(idx_RC_loads))./(2*C(idx_R
C_loads).*L(idx_RC_loads).*R_load(idx_RC_loads)));

                % add initial voltage
                for k=1:length(idx_RC_loads)
                    idx = find(double(string(results.labels(:,
3)))==load_nodes(idx_RC_loads(k)));
                    V_peak_max_load(idx_RC_loads(k)) = V_peak_max_load(idx_RC_loads(k)) + re-
sults.y(idx, 2);
                    V_peak_min_load(idx_RC_loads(k)) = V_peak_min_load(idx_RC_loads(k)) + re-
sults.y(idx, 2);
                end

                %% approximate peak widths by time constants
                % RC loads
                peakWidths_iteration(idx_RC_loads) =
(2*C(idx_RC_loads).*L(idx_RC_loads).*R_load(idx_RC_loads))./(L(idx_RC_loads)+C(idx_RC_loads).*R_
wire(idx_RC_loads).*R_load(idx_RC_loads));
                % R loads
                peakWidths_iteration(idx_R_loads) = L(idx_R_loads)./R_load(idx_R_loads);
            end
            if(size(maxVoltages, 2) == 1)
                if(~isempty(V_peak_max_load))
                   maxVoltages(:) = max(maxVoltages(:), V_peak_max_load);
                   minVoltages(:) = min(minVoltages(:), V_peak_min_load);
```

```
                    peakWidths(:) = peakWidths_iteration;
                end
            else
                maxVoltages = V_peak_max_load;
                minVoltages = V_peak_min_load;
                peakWidths = peakWidths_iteration;
            end
        end
        % extract static fault voltages
        for j=1:length(load_nodes)
            idx = find(double(string(results.labels(:, 3)))==load_nodes(j));
            staticVoltages(j) = results.y(idx, 2);

            idx = find(double(string(results_nominal.labels(:, 3)))==load_nodes(j));
            nominalVoltages(j) = results_nominal.y(idx, 2);
        end

        staticVoltages = staticVoltages';
        nominalVoltages = nominalVoltages';

        if(~isempty(maxVoltages))
%             maxVoltages = max(nominalVoltages, maxVoltages);
%             minVoltages = min(nominalVoltages, minVoltages);
        else
            maxVoltages = nominalVoltages;
            minVoltages = nominalVoltages;
        end

        [triggered_fuse, fusing_time] = evaluate_fuses(results, fuse_label, fuse_type,
fuse_param);
end
```

7.1.5 functions/approximation/short_circuit_routine.m

```
%% Determine short circuit behavior for all possible fault locations
% voltages_open:mxn cell of approximated short circuit voltage arrays
%               (m: load numbers, n: fault locations)
% times_open:   corresponding mxn cell of time vectors
%               (m: load numbers, n: fault locations)
%
% netlist:      cell-array of netlist file
% fault_nodes:  nodes at which short faults are to be evaluated
% load_nodes:   nodes of loads

function [voltages_nominal, voltages_short, times_short] = short_circuit_rou-
tine(netlist, fault_nodes, load_nodes)
    [fuse_label, fuse_type, fuse_param, fuse_node] = get_fuse_infos(netlist);
    voltages_nominal = [];
    minVoltages_preFuse = [];

    voltages_short = cell(0,0);
    times_short = cell(0,0);

    list_dc = generate_dc_netlist(netlist);
    [results_nominal, ~, ~] = networkanalysis(list_dc);

    % extract nominal voltages
    for j=1:length(load_nodes)
            idx = find(double(string(results_nominal.labels(:, 3)))==load_nodes(j));
            voltages_nominal(j) = results_nominal.y(idx, 2);
```

```
    end

    for i=1:length(fault_nodes)
        % inject fault and perform static simulation
        list_short = inject_short_circuit(netlist, fault_nodes(i), 10e-6);
        list_short_dc = generate_dc_netlist(list_short);
        [results, ~, ~] = networkanalysis(list_short_dc);

        % extract resulting load voltages
        for j=1:length(load_nodes)
            idx = find(double(string(results.labels(:, 3)))==load_nodes(j));
            minVoltages_preFuse(j, i) = results.y(idx, 2);
        end
        voltages_short(:, i) = num2cell(minVoltages_preFuse(:, i));
        times_short (:, i) = num2cell(zeros(length(load_nodes), 1));

        %% open circuit routine for triggered fuses; TODO: recursive checking
        [triggered_fuse, fusing_time] = evaluate_fuses(results, fuse_label, fuse_type,
fuse_param);

        if(~isnan(triggered_fuse))
            [voltages_fuse, times_fuse] = open_circuit_routine(list_short, trig-
gered_fuse(1), load_nodes);
            voltages_short(:,i) = num2cell([cell2mat(voltages_short(:,i))
cell2mat(voltages_fuse)], 2);
            times_short (:,i) = num2cell([cell2mat(times_short (:,i))
cell2mat(times_fuse) + fusing_time+150e-6], 2);
        end
    end
end
```

7.1.6 functions/transient/fault_behavior_transient_MNA.m

```
function transient_MNA_results = fault_behavior_transient_MNA (netlist_path, pot_crit-
ical_open, pot_critical_short, approximation_results, simulation_time, step_width)
% Perform transient MNA time domain simulation for potentially critical fault
% scenarios
%
% netlist_path: Path of topology netlist
% pot_critical_open: wire IDs of potentially critical open fault locations
% pot_critical_short: node IDs of potentially critical short fault locations
% approximation_results: results struct of previous preselection (incl system IDs)
%
% transient_MNA_results: struct with calculated voltages

%% load and convert netlist
netlist_raw = fileread(netlist_path);
netlist_raw = strsplit(netlist_raw, {char(10) char(13)});
netlist = cell(length(netlist_raw), 6);
for i=1:length(netlist)
    new_entry = Netlist.splitLine(netlist_raw{i});
    netlist(i,1:length(new_entry)) = new_entry;
end
% change simulation command to perform transient simulation
netlist(end, 3) = {-0.5e-3};
netlist(end, 5) = {simulation_time};
netlist(end, 7) = {step_width};
```

```
%% calculate nominal DC values for initial state
[results_nominal, ~, ~] = networkanalysis(netlist);
initial_values = results_nominal.y(:,end);

%% short circuit simulation
transient_MNA_results.voltages_short = cell(0,0);
for i=1:length(pot_critical_short)
    list_short = inject_transient_short_circuit(netlist, pot_critical_short(i), 10e-6,
2e-6);
    [results, ~, ~] = networkanalysis(list_short, [initial_values; 0]); % adding 0 to
initial values: initial current of short circuit switch
    % convert results to transient_MNA_results cell
    [~, result_idx_loads] = find(double(string(results.labels(:, 3)))' == approxima-
tion_results.load_nodes);
    transient_MNA_results.voltages_short(:, i) = num2cell([results.y(result_idx_loads,
:)], 2);
    transient_MNA_results.times_short(:, i) = num2cell([repmat(results.x, length(re-
sult_idx_loads), 1)], 2);
end

%% open circuit simulation
transient_MNA_results.voltages_open = cell(0,0);
for i=1:length(pot_critical_open)
    list_open = inject_transient_open_circuit(netlist, pot_critical_open(i), 100e6,
2e-6);
    [results, ~, ~] = networkanalysis(list_open, initial_values);
    % convert results to transient_MNA_results cell
    [~, result_idx_loads] = find(double(string(results.labels(:, 3)))' == approxima-
tion_results.load_nodes);
    transient_MNA_results.voltages_open(:, i) = num2cell([results.y(result_idx_loads,
:)], 2);
    transient_MNA_results.times_open(:, i) = num2cell([repmat(results.x, length(re-
sult_idx_loads), 1)], 2);
end

% IDs
transient_MNA_results.load_IDs = approximation_results.load_IDs;
transient_MNA_results.evaluated_nodes_short = pot_critical_short;
transient_MNA_results.evaluated_wires_open = pot_critical_open;

end
```

Bisher in der FAT-Schriftenreihe erschienen (ab 2017)

Nr.	Titel
292	Innenhochdruckumformen laserstrahlgelöteter Tailored Hybrid Tubes aus Stahl-Aluminium-Mischverbindun für den automobilen Leichtbau, 2017
293	Filterung an Stelle von Schirmung für Hochvolt-Komponenten in Elektrofahrzeugen, 2017
294	Schwingfestigkeitsbewertung von Nahtenden MSG-geschweißter Feinbleche aus Stahl unter kombinierter Beanspruchung, 2017
295	Wechselwirkungen zwischen zyklisch-mechanischen Beanspruchungen und Korrosion: Bewertung der Schädigungsäquivalenz von Kollektiv- und Signalformen unter mechanisch-korrosiven Beanspruchungs-bedingungen, 2017
296	Auswirkungen des teil- und hochautomatisierten Fahrens auf die Kapazität der Fernstraßeninfrastruktur, 2017
297	Analyse zum Stand und Aufzeigen von Handlungsfeldern beim vernetzten und automatisierten Fahren von Nutzfahrzeugen, 2017
298	Bestimmung des Luftwiderstandsbeiwertes von realen Nutzfahrzeugen im Fahrversuch und Vergleich verschiedener Verfahren zur numerischen Simulation, 2017
299	Unfallvermeidung durch Reibwertprognosen, 2017
300	Thermisches Rollwiderstandsmodell für Nutzfahrzeugreifen zur Prognose fahrprofilspezifischer Energieverbräuche, 2017
301	The Contribution of Brake Wear Emissions to Particulate Matter in Ambient Air, 2017
302	Design Paradigms for Multi-Layer Time Coherency in ADAS and Automated Driving (MULTIC), 2017
303	Experimentelle Untersuchung des Einflusses der Oberflächenbeschaffenheit von Scheiben auf die Kondensatbildung, 2017
304	Der Rollwiderstand von Nutzfahrzeugreifen unter realen Umgebungsbedingungen, 2018
305	Simulationsgestützte Methodik zum Entwurf intelligenter Energiesteuerung in zukünftigen Kfz-Bordnetzen, 2018
306	Einfluss der Kantenbearbeitung auf die Festigkeitseigenschaften von Stahl-Feinblechen unter quasistatisch und schwingender Beanspruchung, 2018
307	Fahrerspezifische Aspekte beim hochautomatisierten Fahren, 2018
308	Der Rollwiderstand von Nutzfahrzeugreifen unter zeitvarianten Betriebsbedingungen, 2018
309	Bewertung der Ermüdungsfestigkeit von Schraubverbindungen mit gefurchtem Gewinde, 2018
310	Konzept zur Auslegungsmethodik zur Verhinderung des selbsttätigen Losdrehens bei Bauteilsystemen im Leichtbau, 2018
311	Experimentelle und numerische Identifikation der Schraubenkopfverschiebung als Eingangsgröße für eine Bewertung des selbsttätigen Losdrehens von Schraubenverbindungen, 2018
312	Analyse der Randbedingungen und Voraussetzungen für einen automatisierten Betrieb von Nutzfahrzeuge im innerbetrieblichen Verkehr, 2018
313	Charakterisierung und Modellierung des anisotropen Versagensverhaltens von Aluminiumwerkstoffen für die Crashsimulation, 2018

314 Definition einer „Äquivalenten Kontakttemperatur“ als Bezugsgröße zur Bewertung der ergonomischen Qualität von kontaktbasierten Klimatisierungssystemen in Fahrzeugen, 2018

315 Anforderungen und Chancen für Wirtschaftsverkehre in der Stadt mit automatisiert fahrenden E-Fahrzeugen (Fokus Deutschland), 2018

316 MULTIC-Tooling, 2019

317 EPHoS: Evaluation of Programming - Models for Heterogeneous Systems, 2019

318 Air Quality Modelling on the Contribution of Brake Wear Emissions to Particulate Matter Concentrations Using a High-Resolution Brake Use Inventory, 2019

319 Dehnratenabhängiges Verformungs- und Versagensverhalten von dünnen Blechen unter Scherbelastung, 2019

320 Bionischer LAM-Stahlleichtbau für den Automobilbau – BioLAS, 2019

321 Wirkung von Systemen der aktiven, passiven und integralen Sicherheit bei Straßenverkehrsunfällen mit schweren Güterkraftfahrzeugen, 2019

322 Unfallvermeidung durch Reibwertprognosen - Umsetzung und Anwendung, 2019

323 Transitionen bei Level-3-Automation: Einfluss der Verkehrsumgebung auf die Bewältigungsleistung des Fahrers während Realfahrten, 2019

324 Methodische Aspekte und aktuelle inhaltliche Schwerpunkte bei der Konzeption experimenteller Studien zum hochautomatisierten Fahren, 2020

325 Der Einfluss von Wärmeverlusten auf den Rollwiderstand von Reifen, 2020

326 Lebensdauerberechnung hybrider Verbindungen, 2020

327 Entwicklung der Verletzungsschwere bei Verkehrsunfällen in Deutschland im Kontext verschiedener AIS-Revisionen, 2020

328 Entwicklung einer Methodik zur Korrektur von EES-Werten, 2020

329 Untersuchung zu den Einsatzmöglichkeiten der Graphen- und Heuristikbasierten Topologieoptimierung zur Entwicklung von 3D-Rahmenstrukturen in Crashlastfällen, 2020

330 Analyse der Einflussfaktoren auf die Abweichung zwischen CFD und Fahrversuch bei der Bestimmung des Luftwiderstands von Nutzfahrzeugen, 2020

331 Effiziente Charakterisierung und Modellierung des anisotropen Versagensverhaltens von LFT für Crashsimulation, 2020

332 Charakterisierung und Modellierung des Versagensverhaltens von Komponenten aus duktilem Gusseisen für die Crashsimulation, 2020

333 Charakterisierung und Meta-Modellierung von ungleichartigen Punktschweißverbindungen für die Crashsimulation, 2020

334 Simulationsgestützte Analyse und Bewertung der Fehlertoleranz von Kfz-Bordnetzen, 2020

335 Absicherung des autonomen Fahrens gegen EMV-bedingte Fehlfunktion, 2020

336 Auswirkung von instationären Anströmeffekten auf die Fahrzeugaerodynamik, 2020

337 Analyse von neuen Zell-Technologien und deren Auswirkungen auf das Gesamtsystem Batteriepack, 2020

338 Modellierung der Einflüsse von Mikrodefekten auf das Versagensverhalten von Al-Druckguss-komponenten mit stochastischem Aspekt für die Crashsimulation, 2020

339 Stochastisches Bruchverhalten von Glas, 2020

340 Schnelle, breitbandige Datenübertragung zwischen Truck und Trailer als Voraussetzung für das hochautomatisierte Fahren von Lastzügen, 2021

341 Wasserstoffkompatibilität von Aluminium-Legierungen für Brennstoffzellenfahrzeuge, 2021

342 Anforderungen an eine elektrische Lade- und Wasserstoffinfrastruktur für gewerbliche Nutzfahrzeuge mit dem Zeithorizont 2030, 2021

343 Objective assessment of database quality for use in the automotive research and development process, 2021

344 Review of non-exhaust particle emissions from road vehicles, 2021

345 Ganzheitliche Betrachtung von Rollwiderstandsverlusten an einem schweren Sattelzug unter realen Umgebungsbedingungen, 2021

346 Studie zur Abschätzung der Anwendungspotentiale, Risiken und notwendigen Forschungsbedarfe bei der Verwendung von Glashohlkugeln in Kombination mit thermoplastischem Schaumspritzguss, 2021

347 Typgenehmigungsanforderungen an Level-3-Autobahnsysteme - Hintergrundbetrachtungen zu technischen Anforderungen für eine automatisierte Fahrfunktion, 2021

348 Einfluss der Kantenbearbeitung von Aluminiumblechen auf das Restumformvermögen sowie die Festigkeitseigenschaften unter quasistatischer und schwingender Beanspruchung, 2021

349 Verstärkung dünner formgehärteter Bauteile mittels FVK-Verrippungen, 2021

350 HMI Anforderungen für den automatisierten Individualverkehr unter Berücksichtigung von Leistungsmöglichkeiten und -grenzen älterer Nutzer, 2021

351 Compatibility of polymers for fuel cell automobiles, 2021

352 Entwicklung einer gewichtsoptimierten Batteriegehäusestruktur für Volumenfahrzeuge, 2021

353 Charakterisierung und Modellierung des Deformations- und Versagensverhaltens von nicht-faserverstärkten Thermoplasten unter mehrachsiger Crashbelastung, 2021

354 Untersuchung zum thermischen Komfort im Pkw für den Grenzbereich des Luftzugempfindens, 2021

355 Anforderungen an die Güte, Verfügbarkeit und Vorausschau einer Reibwertschätzung aus Funktionssicht, 2021

356 Entwicklung einer standardisierten Prüfanordnung zur Bewertung der Übernahmeleistung beim automatisierten Fahren, 2022

357 Vorstudie zu Verkehrsemissionen - Räumlich und zeitlich aufgelöste Daten durch Schwarmmessungen, 2022

358 Produktivitätssteigerung und Kostensenkung der laser-additiven Fertigung für den Automobilbau, 2022

359 Analyse der Einflussfaktoren auf die Abweichung zwischen CFD und Fahrversuch bei der Bestimmung des Luftwiderstands von Nutzfahrzeugen mit Fokus auf den Ventilationswiderstand von Nfz-Rädern, 2022

360 Werkstoffmodelle und Kennwertermittlung für die industrielle Anwendung der Umform- und Crash-Simulation unter Berücksichtigung der thermischen Behandlungen beim Lackieren im Prozess bei hochfesten Werkstoffen, 2022

361 Compatibility of polymers for fuel cell automobiles, 2022

362 Ermüdung kurzfaserverstärkter thermoplastischer Polymerkwerkstoffe, 2022

363 Market research and definition of procedure to comparison of comfort measuring systems for a vehicle cabin, 2022

364 Methodische Ansätze zur Auswahl von Bordnetzstrukturen mit erhöhten Zuverlässigkeitsanforderungen, 2022

Impressum

Herausgeber	FAT Forschungsvereinigung Automobiltechnik e.V. Behrenstraße 35 10117 Berlin Telefon +49 30 897842-0 Fax +49 30 897842-600 www.vda-fat.de
ISSN	2192-7863

Verband der Automobilindustrie e.V. (VDA)
Behrenstraße 35, 10117 Berlin
www.vda.de
Twitter @VDA_online

Forschungsvereinigung Automobiltechnik e.V. (FAT)
Behrenstraße 35, 10117 Berlin
www.vda.de/fat

VDA | Verband der Automobilindustrie

FAT | Forschungsvereinigung Automobiltechnik