Markus Mayerwieser

A Generic Simulation Framework for Next Generation NFC-Transceivers

Master's Thesis

Graz University of Technology

Signal Processing and Speech Communication Laboratory (SPSC)
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Gernot Kubin

Supervisor: Dr. Klaus Witrisal
Mentors: Dr. Ulrich Mühlmann / Dr. Stefan Mendel

Gratkorn, December 2015

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Appreciation

The completion of this thesis, and therefore my studies of Telematik at TU Graz would have never been possible without some very dear persons in my life. First, i want to thank my parents Irmgard and Wolfgang. Thanks for giving me the opportunity and support to get my education. Also thanks for motivating me, always believing in me and challenging me to give my best (looking at you dad ;)). I also want to thank my brother Thomas who always had an open ear and some advising words and my grandfather Werner for passing me his stubbornness. During the time of the thesis, the most affected person was of course my lovely girlfriend Ramona. She showed a lot of patience. She is, and always will be my anchor in life.

My time at university would have been way harder without my dear friends Gabriel Haas, Timotheus Hell and Markus Blauensteiner. Thanks you guys for being reliable friends and colleagues. It may be odd to thank my bachelor thesis mentor Berhard Geiger here in the master thesis, but anyway: Thank you Bernhard for being one of the nicest guys in the world and inspiring me to get into the field of digital signal processing. An important person to mention is of course my university mentor Klaus Witrisal, who helped me bringing all the things into a thesis form. Working on the thesis and writing it all down are two different things, so thank you Klaus for your guidance.

Last but not least i want to thank NXP for the great chance of doing my thesis as a company project. I want to thank Alexander Maili for believing in me and take me on board, also big thanks to Michael Michelitsch for recommending me to the right company. I want to thank Ulrich Muehlmann, who mentored me in a very patient and productive way. I think in this world exists NO problem where Ulrich does not have an idea about how to overcome it. My last thanks are for Stefan Mendel who was my mentor as well. His guidance brought structure to my work and I am pretty sure it would not be finished yet without him. I am very glad i was able to work with such brilliant people. Now I look forward to my further work in NXP with my dear mentors that are now colleagues.

Abstract

The market for Near Field Communication (NFC) devices grows rapidly. Different applications require different features and consequently the NFC solutions become more complex. Research & development for the devices therefore is an economically and technologically demanding matter. Pre-silicon verification & validation is a good way to avoid costs for bug-fixing and to reduce the time to market and increase the reliability of devices.

This thesis addresses the need for a holistic approach to do pre-silicon simulations for NFC devices. A framework is introduced which is capable of performing functional simulations on system level. The system includes a stimuli generator for different types of stimuli and the flow of this signal from the transmitter over the air-interface to the receiver. Evaluation functions for the output are supported to provide closed-loop testing. The system model reflects the behaviour of the real world system in an appropriate grade of detail.

The crucial part of the framework is the contact-less interface between transmitter and receiver. This interface is modelled as an inductive coupling of the device antennas. The antennas are loaded with their corresponding transmitter and receiver circuits. Differential equations describe the inductive coupling and the transmitter and receiver circuits which are described as LC resonator circuits. The system of differential equations for the couplings and load circuits is represented as a state-space model. State-space models enable easy and fast time-domain simulation.

The outcome at this thesis is a generic framework to model NFC devices in different operations modes: Card mode $\leftrightarrow$ Reader mode, Reader mode $\leftrightarrow$ Card mode and NFC $\leftrightarrow$ NFC. The model generation needs just a netlist as input.

To illustrate the usage of this framework, exemplary simulation results are provided. Different application scenarios are analyzed and optimised using the automatically generated models and the framework inherent stimuli generation.

Contents

List of Figures

1 Introduction

1.1 Introduction to Near Field Communication

In modern pervasive computing applications, device communication works mainly wireless. Devices get smaller and capable of more complex functions. Battery capacity and energy consumption are important topics. First approaches for energy efficient, wireless identification applications go by the name of Radio Frequency Identification (RFID). A transponder device (also called tag, proximity integrated circuit card or PICC) is inductively coupled and passively powered by the electromagnetic field of a reading device (also called proximity coupling device or PCD). As the name part 'Radio Frequency' says, the field is generated at the frequency of radio bands. The reader device provides the radio frequency field, the tag is capable of sending its identification back to the reader. For most relevant applications the data transmission is achieved by load modulation. The transponder changes its load corresponding to the transmit data. A binary '0' is represented by a changed load circuit condition (modulation), a binary '1' by the original circuit condition (unmodulated). The reader senses the change in the load of the Tag. This modulated signal contains an encoded bit-stream which can be interpreted by the reader. Inductive coupling and the idea of load modulation are also the fundamentals of Near Field Communication (NFC).

In comparison to RFID, NFC-Devices are more advanced in terms of hardware features and communication protocols. Applications are not limited to identification, where a reader gets the information stored in a tag/card. In NFC, data can be exchanged in both directions. Bidirectional communication in different operation modes is supported. The physical parameters and protocols for the communication are standardized to make devices

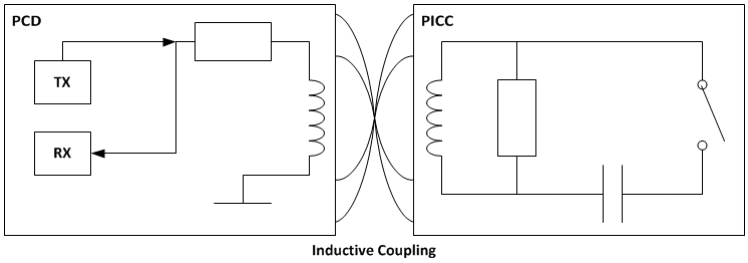

Figure 1.1: ISO 14443: Proximity Card Standard

compatible. NFC-Devices operate as communication initiators (masters) and as communication targets (slaves) and can change the direction of the data flow. Similar to RFID the devices are inductively coupled and operate at 13.56MHz carrier frequency. The communication range is limited to 20cm (an operation distance of 10cm is common). Data rates for NFC-Devices are at 106$kbit/s$ to 848$kbit/s$ using Amplitude Shift Keying (ASK). The load modulation can work in a passive (circuit change by switching loads) and an active (source driven) way. The encoding for an NFC device in reader mode for sending data includes (modified) Miller, Manchester and Non-Return-To-Zero(NRZ). NFC devices in card mode use Manchester encoding with subcarrier, Binary Phase Shift Keying (BPSK) and normal Manchester encoding to send data. For more detailed information about RFID and NFC basics please refer to (Finkelzeller, 2008), (ISO/IEC, 2008) and (ISO/IEC, 2013).

1.2 Applications and Operation Modes

NFC-Devices offer a wide range of applications. In general NFC is used for identification and data transmission between terminal points. In the automotive field, NFC is used for wireless entry solutions. The NFC-Device in the key authenticates and identifies itself to the car which grants access to the doors or even engine. In infrastructure applications, NFC-Devices also operate as authentication and identification devices to gain the right to enter

a building. Contactless payment is also driving the NFC market. Payment information and of course device/user authentication has to be done on both communication ends. Besides for payment, NFC is used in mobile phones for data transmission and natural pairing of accessories (headphones, ...) via tapping. Here NFC acts as an enabler for other wireless standards such as Bluetooth where pairing is less intuitive. New up-and-coming approaches in R & D aim to combine more than one application into one device. A mobile phone can provide payment information, infrastructure access control and keyless entry for a car. Summarized, the behaviour of an NFC-Device is dedicated to its supported and used standards and the current operation mode. As details on the standards are not important to understand the simulation framework approaches in this thesis, please refer to Finkelzeller, 2008 for further information. NFC devices can operate in different modes. Peer-to-peer mode, card mode and reader mode are possible options for the device. The operation modes of NFC devices are essential to understand the framework structure and applications, therefore the next subsections will give a short explanation.

1.2.1 Peer-to-Peer Mode: NFC Active

The sending device operates in reader mode, the receiving device operates in card emulation mode. The initiator starts the communication and generates the field. After the transmission the initiator switches off its field and the target takes the role as reader by switching its field on. This ping-pong game of RF-Field generation continues until the end of the bidirectional communication (see Fig. 1.2). For further information, see **ISO15693**

ISO18092 also contains the option of a peer-to-peer passive mode, where data are bidirectionally exchanged, but the initiator always acts as reader and therefore provides the RF-field.

1.2.2 Card Mode

In Card mode (see Fig. 1.3) the NFC-Device acts as an RFID tag. No communication initialization is done by the device. It just responds to reader

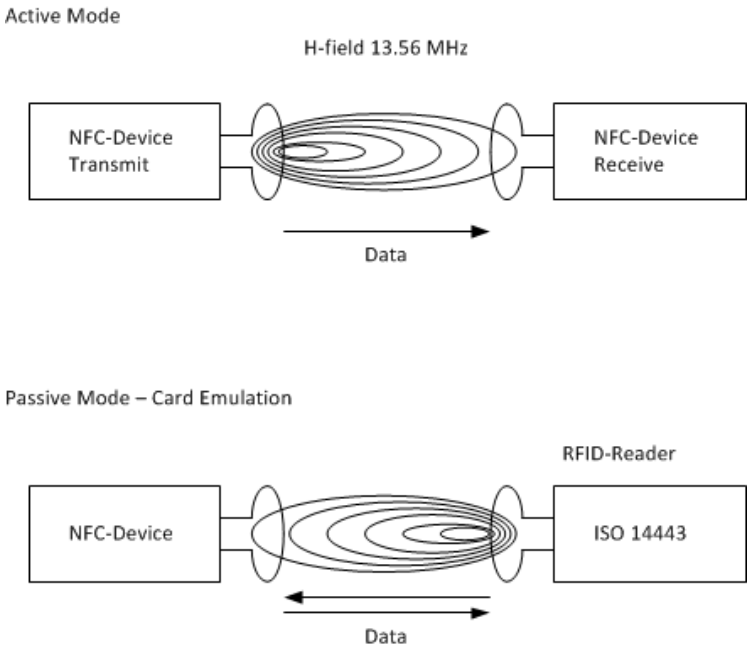

Figure 1.2: Peer-to-Peer: NFC active

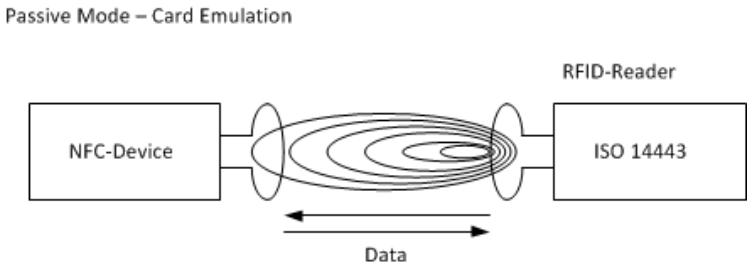

Figure 1.3: NFC Device in Card Mode

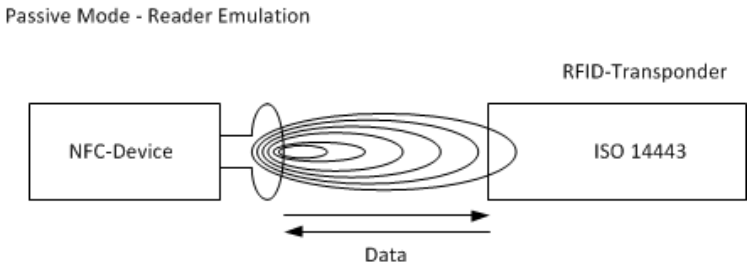

Figure 1.4: NFC Device in Reader Mode

commands. Reader devices can be NFC-Devices in Reader Mode or RFID readers.

1.2.3 Reader Mode

In Reader mode (see Fig. 1.4) the NFC-Device initiates communication with other devices and works as a reader device. The other devices can be passive cards, or NFC-Devices in card emulation mode.

For further information, please refer to ISO/IEC, 2008 and ISO/IEC, 2013.

1.3 Motivation

NFC includes many different applications and use-cases. The variety of operation modes, modulation types and encodings bring a big set of features that have to be taken into consideration when it comes to research & development (R&D) as well as verification & validation (V&V). Increasing requirements for NFC system complexity and time to market demands create the need for pre-silicon simulation. Concepts have to be tested fast and easily before producing a physical device. Simulation stimuli have to be created, simulation models have to be generated, the results have to be evaluated. There are software tools to perform parts of this procedure, but only a combination of these tools within a complex and inflexible tool-chain includes all features.

These requirements imply the need for a holistic framework. A modularly structured framework allows for rapid concept engineering and concept verification. Time and frequency domain simulation results yield information about different aspects of the system. The pre-silicon simulated data can be used at different stages of the development. For example, test stimuli for just digital verification can be generated. The framework includes all parts of the signal-chain and is able to perform a simulation from signal generation to output evaluation. Still, single modules of the framework can be used as inputs to other software or even laboratory hardware. Evaluation modules can be used to evaluate results from external sources. Hence, the framework can be used for concept engineering, general R & D as well as for V & V purposes. The applications include the optimization of the analogue block and the digital block. Antenna designs can be tested and matching circuits can be optimized. For existing ICs (integrated circuits) the optimal operation settings can be found. Silicon failure scenarios can be reproduced by modelling the effects and potential workarounds may be identified. The framework (or parts of it) can be used throughout all stages of an NFC development project. Tests can include the whole system or just parts of it.

This thesis is about setting up an extendible framework that includes:

- Basic stimuli generation functions
- Matching circuit modelling

- Matching circuit simulation
- Simulation result evaluation and graphic representation
- Comparison with spice simulations

2 Framework

The wide variety of use-cases and applications for NFC raises the need for a holistic framework which can be used for the simulation and evaluation of all upcoming use-cases and application. To build a framework that is able to fulfil all future use-cases and applications, the requirements for the use-cases and applications have to get more specific. This chapter specifies the use-cases, applications and structure for the framework. The requirements and features for the framework are defined due to this use-cases, applications as well as from structural needs (how to build modular and extendible) and convenience reasons (how to integrate already existing information).

2.1 Basic Framework Functions

The framework needs to be constructed in a modular way to provide options for extension. The overall framework connects the sub-blocks, therefore sub-blocks can be changed to fit to new test scenarios. The stimuli generation, model generation as well as the simulation and evaluation should be present just within one dynamic exchangeable signal chain. The sub-blocks are designed to work as stand-alone modules.

Stimuli generation has to include different types of signal encodings and data rates as well as different (integer) oversampling rates. These parameters of the input signal define the communication type and the used operation mode. For the important use-cases of the ISO-standards different signal shapings need to be supported to reflect the ISO-scenarios. This block can also be used to create a stimulus file for spice simulators or similar.

The model generation of the coupling system (the air interface including matching RX/TX) needs to work automatically for different circuit topologies and different component values within a circuit. Coupling factors are to be seen as circuit parameters just like the component values. Therefore the variation in geometric coupling conditions is modelled using these coupling factors.

The simulation has to support non time-varying and time-varying behaviour (as it is used for passive load modulation). Multiple Sources need to be supported for active load modulation, due to the fact that there is one source to create the RF-Field, another source modulates this RF-field in an active way. Simulations in time and in frequency domain need to be supported to cover and analyse all model characteristics.

Evaluation functions provide access to the simulated data and prepare it for investigations. functions from this block can also be used to read simulation data from spice simulators and evaluate it or compare it with other simulation data.

With the summary of the basic framework functions and a set of showcase applications, we can define the requirements list:

- Modular structure
- Stimuli generation
- Automatized Coupling Model generation
- Simulation in time & frequency domain
- Output evaluation
- Parameter optimization
- Extendibility for different test scenarios
- Blocks individually applicable
- Time and Frequency Domain Simulations
- Multiple Input/Multiple Output System (MIMO)
- Generalization for:
  - Topology of the circuits
  - Parameter changes of individual components
- Antenna Coupling
- Time-varying behaviour (modulation)

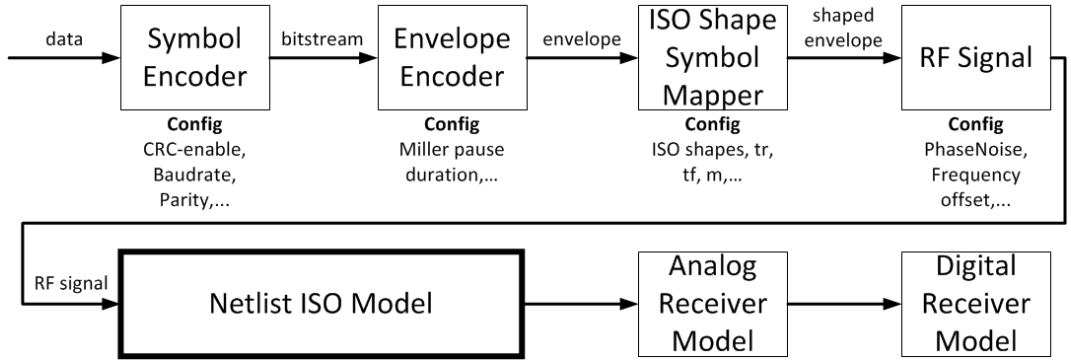

Figure 2.1: Framework Toplevel Structure

- Transient behaviour (circuit condition should be kept consistent for transient simulations)

2.2 Structure

Figure 2.1 depicts the top-level structure of the framework. This top-level description shows the connections between the blocks and the responsibility of each block. The interfaces of the blocks are defined in terms of their content. This already shows what the input, the function and the output of each block has to be.

2.3 Showcase Applications

Now that the modularity of the framework is given, more detailed information about its features is needed. To get an idea of what features are needed, standard applications are taken as a reference. The following showcase applications help to identify functional needs for the framework. These real-life applications are yet unsolved (or unsatisfactorily solved) issues in research & development of NFC systems.

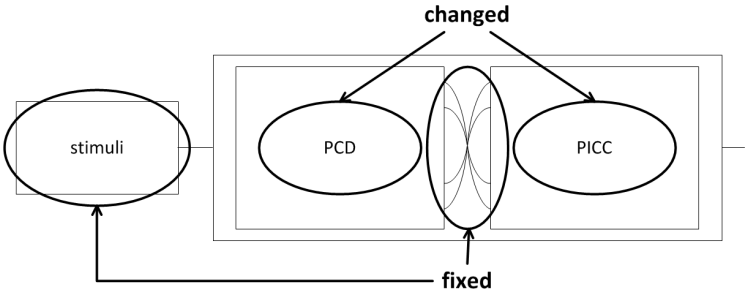

Figure 2.2: Matching Parameter Tests

2.3.1 Matching Parameter Optimisation

The matching circuits consist of resistors, capacitors and inductors. The combination of these components are resonator circuits that define the behaviour of the circuits and therefore the behaviour of the inductive coupling between the antennas. Each component value can be seen as a parameter that changes this resonance behaviour. Hence, the framework needs to support configuration files to perform the simulation for different component parameters. Given a fixed circuit and a fixed coupling (represented by coupling factors k), the optimal component values can be identified (as shown in Fig. 2.2).

2.3.2 De-Tuning Influence Analysis

In Fig. 2.3 we can see the De-Tuning Impact Analysis use-case. The components in the matching circuit define the tuning of the coupling system, this tuning is done for a certain coupling scenario (and therefore a certain value k). For this fixed circuit (with parametrization) the coupling factors k can be varied. The variation in coupling factors represents a variation in the position of transmitter and receiver antennas and therefore another coupling scenario. The variation of k leads to a de-tuning of the matching network, hence a change of the system behaviour. The target of this framework use-case is to find a correspondence between the voltages and currents in the

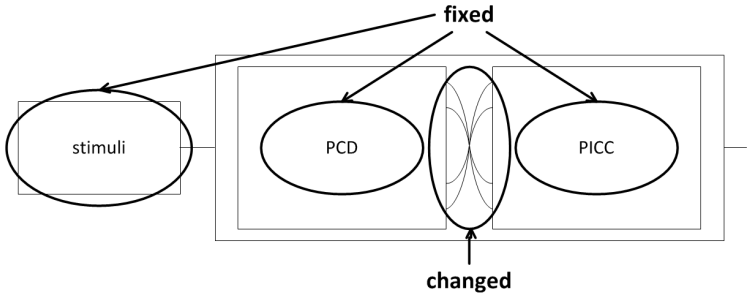

Figure 2.3: De-Tuning Influence Analysis

system and the coupling parameters k. The values for k can be obtained by antenna simulation tools or measurements in laboratory setups.

2.3.3 ISO Wave-Shape Verification

ISO14443 defines the allowed reader wave-shapes in terms of rise/fall times, overshoot, ... These shapes are define at the air interface. To be able to communicate with all ISO compliant readers, the card mode receiver needs to be able to decode all of these shapes. For a fixed circuit with fixed coupling the stimulus is modified according to the conditions reflected in the ISO-shape and then simulated (see Fig. 2.4). This is called the ISO-setup.

If the transmitter sends a not perfectly formed signal (of course within some limitations) the receiver still has to be able to decode it error free. A database of standardized envelope shapes is used to determine the card receiver performance.

2.4 Framework Functions

Given the structure of the framework and a set of showcase applications which brought up some needed features and functions, the internal block

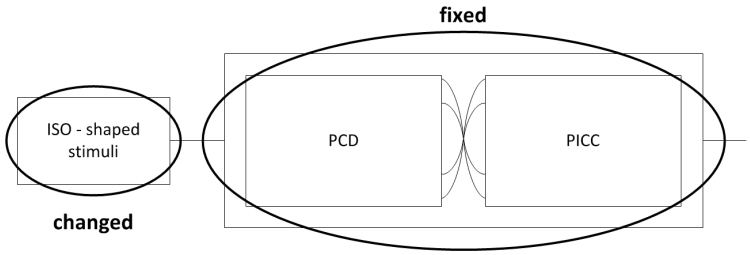

Figure 2.4: Time-Domain Shape Verification

functionality can be identified and mapped to the right blocks. The following subsections will cover the internal block functionality of the framework.

2.4.1 Envelope Generation

A given data word (given as a hex value) is converted to a bit stream. This bit stream is an encoded version of the data word. Encoding types include Non-Return-to-Zero (NRZ) encoding and Manchester encoding. The baudrate of the signal is considered, as well as some protocol related bits (CRC, Parity, ...). We call the encoded bit-stream the baseband envelope.

2.4.2 RF-Carrier Modulation

The baseband envelope gets up-sampled by a desired oversampling rate. Then the up-sampled envelope is modulated with the RF-carrier of $13.56MHz$. The modulation happens by a sample-wise multiplication of the sample values, where the carrier and the envelope are scaled by a factor that is related to the modulation depth. A modulation depth of 10% means that 10% of the amplitude of the envelope signal is used and 90% of the RF-signal's amplitude. The resulting RF-signal can be used as a stimulus to the time-domain state-space model.

2.4.3 Symbol ISO-Shape Mapping

For the application of wave-shape verification, the ISO compliant wave-shapes get applied here. A database for standardized ISO-shapes is provided as a Matlab data container. A wave-shaping tool is used to perform the linear filtering for the envelopes according to the timing definitions of the ISO compliant wave-shapes. The filtered signals are saved in Matlab container structures for further use. A perfect data envelope is used to find the general signal form. The edges in the perfect envelope are replaced by the shaped patterns from the Matlab container structure. To do this, a specific shape gets copied to the position in the bit-stream, where the corresponding edge is found. Therefore the shaped RF-signal is like the original RF-Signal with parts of it substituted by the shape. This block is optional in the signal chain, but necessary for ISO-shape verification.

2.4.4 Coupling System

The core of the simulation framework always remains the Air-Interface model, noting that the topology, circuit parametrizations and coupling conditions can vary. The main aspect for the framework generalization is the automated generation of models for different air-interface circuits. All other features are built around this air-interface model. The given modular framework structure enables the further implementation of extensions. New standards (including new modulations/encodings) and new signal shapes can be added.

The output of the automated model generation is a model with symbolic variables. Parametrizing this model means that the symbolic variables get substituted by the numeric values of the circuit components that are included in the model. For a maximum in flexibility and a minimum of computation time, generated models are saved as non-parametrized (symbolic) models for simulations with changing parameters (parametrization has to be done before simulation) and a fully-parametrized model that includes the standard component values from the netlist. The non-parametrized model provides the most flexible way to save a model, hence the values can be changed without the need of generating the model from scratch. The

fully-parametrized model provides the minimum computation time since a fully simulate-able model can be loaded and used.

2.4.5 Simulation Module

The RF-carrier signal is used as stimulus to the model for time-domain simulations. As mentioned before, the RF-carrier can be ISO-shaped for some of the use-cases. The simulation module also supports block processed simulations. This kind of simulation is needed for passive load modulation scenarios (time varying behaviour of the system) or simulations where the feedback of the system is used for dynamic adaptation of the transmitter wave-shape to be ISO compliant.

Each simulation result gets saved in a container structure. This container holds not just the simulation results, but also the stimulus signals and relevant model data. With that approach results can be fully evaluated at a later time.

2.4.6 Evaluation

Using the container structure from the output of the simulation block, the signal can be plotted for each state variables and each other component of the circuit. This is possible due to the fact that all components in the circuit can be represented as a linear combination of state variable. The equations for the calculation of the linear combination are available from the model generation and are present in the result container.

2.5 Core Modelling Problem: Coupling System

The generation of the air-interface model is the core block of the whole framework, not only in terms of importance but also in terms of implementation effort. The air-interface is given as a circuit schematic of the matching

network. This network includes the transmitter and the receiver antennas and the whole matching circuits on both connection sides.

The following subsections show which approaches are already researched to model this air-interface. Spice simulators provide a way to simulate circuits like this, also Matlab tools are available. One thing that all approaches have in common is to describe the system as a set of differential equations. This section will show the pros and cons of the available approaches and derive the used modelling approach from this.

2.5.1 Systems Description as Differential Equations

The circuits include resistors R, inductors L and capacitors C alongside voltage V and current I sources. Even if the description of current and voltage values for resistors is easily given by Ohm's Law $V_R = R \cdot I_R$, it is not so straight forward for inductors and capacitors in an alternating voltage/current circuit. Both components' behaviour can be explained with differential equations: $I_C = -C \cdot \dot{V}_C$, $V_L = L \cdot \dot{I}_L$.

The common way of describing electrical circuits is to form a system of differential equations. This system can be used for simulations where the transient behaviour of L and C components is taken into account. There are sophisticated algorithms to form systems of differential equations from component equations. The system of differential equations forms a 'state-space model'. Therefore finding the differential equations of the system is the key to represent the system as a model that can be simulated in time-domain.

2.5.2 The Modified Node Analysis Algorithm

A widely used algorithm to find the differential equations for a circuit is called Modified Node Analysis (MNA). A circuit is defined by the connection points of the included components. These connections are called nodes. The system of differential equations is formed with respect to all node voltages, node currents and sources in the circuit. If a system is solved by

hand, all the circuit equations are needed (Kirchoff's Equations). The special feature of the MNA algorithm is its simple model construction without the Kirchoff Equations. Instead of using all given component and circuit equations at once, the system is divided into sub parts. Each sub part can be determined by the knowledge of the node connections and a set of rules. These rules define where to put the component variables into the matrix without the explicit knowledge of Kirchoff's equations. When looking at a component, there are some simple things to check that define where to put it, which sign to use (direction) and if the value has to be inverted. Kirchoff's equations are given implicitly by applying these matrix building rules.

The general form of an MNA system is given by:

$$A \cdot x = z \tag{2.1}$$

$$x = A^{-1} \cdot z \tag{2.2}$$

where A is a matrix of $(m+n)$x$(m+n)$ where m is the number of independent sources and n is the number of nodes in the circuit. x is a vector with a length of $(n+m)$. Its contents are the unknown node voltages and currents of the circuit. z is a vector of length $(n+m)$ (same as x) and contains the known parts. Independent voltage and current sources make up the known parts.

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix}$$

G is a matrix of nxn and filled with the connections of the circuit elements. For capacitors and inductors, the complex impedance representation is used: $Z_C = \frac{1}{sC} = \frac{1}{j\omega C}$ and $Z_L = sL = j\omega L$.

B is a matrix of nxm and contains all connections of the sources. If a voltage or current source is connected between to nodes, the matrix B has an entry of '1' or '-1' (dependent on the direction).

C is a matrix of mxn and is the transpose of the matrix B: $C = B^T$

D is a matrix of mxm and is filled with all zeros if only independent sources are present in the circuit.

The matrices G, B, C and D are filled with values and variables according to a set of rules that determine which variable to put where with which sign. The rules determine sub-matrices of the matrix A in such a way, that the differential equations for the system are implicitly defined. No differential equation has to be written explicitly, just the matrix filling rules have to be applied. This is a very fast and efficient way to build up the differential equations for the system.

For a closer look at the rules to fill the Matrices, please refer to
Wing, 2008.

The MNA Algorithm (or a slightly modified version of it) is used by most spice simulators. There is also a Matlab toolbox available with an MNA solver. The advantages and disadvantages are discussed in the next two subsections.

Spice Simulator

Spice simulators use detailed equations for the components that also include physical behaviour like temperature dependence of components. Spice simulators give the most accurate results for all kinds of simulation. It is possible to model inductive antenna coupling. Simulations in time and frequency domain are possible. Most spice simulators use the discussed MNA algorithm as a part of their circuit representation. Therefore, they use the approach of representing the circuit state at each node. This yields approximated results for the desired component voltages and currents (at capacitors and inductors), but with a very high numerical accuracy. Other high-order differential equation models exist as basis for the spice simulation. Due to their complexity they are not desired and also not discussed in this thesis.

The representation of the circuit is essential for the effectiveness of the modelling algorithm. For spice simulators, netlists (or the corresponding

schematics) serve as input for the spice simulation. The output evaluation is limited to plots of currents at components and voltages at nodes. Some mathematical operations are directly applicable in the tool. The component and circuit describing equations are hidden in a machine (only) processable form. Some spice simulators don't provide equations. This fact makes more complex or protocol related output evaluation impossible. Outputs can be written to text files and evaluated in other tools (like Matlab).

Stimuli generation is just possible on a very basic level. Current and voltage sources with transient functions and frequency sweeps can be generated. Communication protocol relevant stimuli can't be created. No data encoding, ISO-shaping or transmission package organisation can be done within the spice simulator. To use input with protocol relevant data, it has to be generated in an external tool (like Matlab) and imported as a source file.

Due to the option of voltage controlled switches, time-varying behaviour can be simulated. A switch changes the parametrization of the circuit. The voltage that controls the switch has to be controlled via an input file (just like other protocol relevant inputs). Simulations with multiple sources are considered in spice simulators, also inductive antenna coupling can be modelled and simulated.

An integration into a holistic simulation framework is not reasonable. The simulator could be started and used via a Matlab framework, the options and features would still be very limited compared to the manual use of the spice simulator.

Spice simulator results are sophisticated and can be used to verify the functionality of other simulators.

MNA Solver Matlab Toolbox: SCAM

The MNA solver SCAM is a Matlab Tool that operates on the same set of system equations as spice simulators with MNA. A detailed description and the sourcecode can be found in Cheever, 2014. The system of differential equations is formed with the MNA algorithm. The resulting system is given a kind of state-space model in s-domain. Just like the spice simulator, the results are approximations for the component voltages and currents, but less

accurate than the results of the spice simulator, because no more physical behaviour is taken into account. This system can be converted to a transfer function for simulation purposes.

The input for the MNA solver is also a netlist like for the spice simulator. From the netlist, the model creation is very fast and easy. The circuit equations are implicitly given by the system due to this approach.

As the MNA solver is a Matlab tool, it is easy to integrate into a framework. The resulting system is not given in any of Matlab's standard system representations, therefore a simulation with integrated Matlab functions is not directly possible. A conversion of the system to a transfer function has to be applied. This conversion generally creates an single input system. For simulation scenarios with active load modulation (card modulates the load with a source), a single input system is not sufficient. Time-varying simulations are not considered in a straight forward manner. Inductive coupling is also not considered in this model creation process, as it would not be straightforward with this approach.

2.5.3 Discussion of Related Work and Motivation for Proposed Solution

The lack of any protocol-related stimuli generation and output evaluation makes it necessary to embed a spice simulator into a tool-chain and not use it alone. A full integration of the spice simulator into a framework is just not possible due to the lack of interfaces of the spice software. Data exchange can just happen via files. Protocol-related stimuli have to be created in an external tool (Matlab, ...) and loaded as a source control file for the current and voltage sources in the spice circuit. The results can be plotted directly in spice but have to be saved and transferred to another external tool for further evaluation.

The MNA solver can be integrated very easily and supports the need for the desired stimuli generation. The form in which the system is represented is not directly suitable for the Matlab simulators, additional effort for system conversion or advanced solvers are needed. The state variables are given

Table 2.1: Comparison: Spice Simulator vs. MNA Solver: SCAM

Spice Simulator	**MNA Solver: SCAM**
+ most generic	- not generic
+ straight forward circuit description	- needs a circuit description from Spice
+ simulation in time and frequency domain	- no efficient way for simulation without model conversion
+ parameter variation	+ parameter variation
+ time-varying behaviour	- no time-varying behavior
- no protocol level stimuli generation	+ stimuli generation in Matlab
- not integrateable in holistic framework	+ integrateable in holistic framework
- insufficient output evaluation	+ output evaluation in Matlab
+ antenna coupling considered	- antenna coupling not considered

for each node of the circuit and not on the energy preserving L and C components. For output evaluation, additional effort is needed to calculate the equations for the desired variables (which are not necessarily the nodes).

The circuit representation as netlist is common and a good option to use. The desired approach for the modelling needs to have the smooth integrability of the MNA Matlab toolbox. It is a big advantage for the framework if the stimuli generation, the model generation, the simulation and the output evaluation can be done within one piece of software. The desired simulation approach is oriented towards the accuracy and simulation speed of spice simulators.

The general idea of using differential equations to describe the circuit topology and condition is the way to go. As the MNA algorithm based tools just give approximated values for voltages and currents at components, they lack accuracy or the simulation speed suffers severely. The solution to this problem is to form the model with respect to other variables than the node voltages and node currents. A state-space model formed from node and mesh equations (Kirchoff's laws) with respect to the energy storing variables in the circuit (voltages for capacitors and currents for inductors)

is a proper solution. The accuracy for this state variables is very good, still the modelling can be done in Matlab (which makes the integration into a holistic framework rather easy).

The model generation effort is higher for the Kirchoff's laws approach than for the MNA approach. All circuit describing equations and component equations have to be found and used to define the dynamic system equations. The additional effort pays off in terms of easier simulation options due to Matlab integrated and optimized simulator functions. The output evaluation has to be done with the circuit equations for each of the approaches. The Kirchoff's laws approach has a point because the equations are given explicitly from the model generation and not implicitly in the model.

2.5.4 Conclusion of Related Work and Proposed Solution

In summary, the chosen approach is a Matlab based framework for stimuli generation, model generation, simulation and output evaluation. The chosen approach combines the positive features of spice simulators and the MNA solver and extends the features where both other options lack the needed functionality. The model generation is done via Kirchoff's Laws which provides equations that are formed to a state-space model. This trades higher model generation effort for better generalization. Due to the fact that Matlab does not provide a netlist to state-space model function, this is one of the main tasks in this thesis. State-space models can be simulated very efficiently by using Matlab built-in simulation functions (lsim, ...). The lack of antenna coupling in the MNA solver is a big draw back that can also be fixed by using the Kirchoff's laws approach.

3 Coupling System: Model Design

The systems of differential equations are organised in a structure called a state-space model. General state-space models are used to model linear systems of first order. This chapter shows the fundamental theory for state-space systems and discusses which equations are used to form the dynamic system equations and how they are used as a model representation.

3.1 Finding the Dynamic System Equations

The selected way of modelling includes the step of finding the equations for the dynamic system. Voltage mesh equations and current node equations represent the basic description of the circuit topology. The dynamic behaviour comes into play by using the component equations for the resistors R, capacitors C and inductors L. For the NFC circuits the inductors are the device antennas, so the component equations for inductors include the inductive coupling part for the wireless NFC system.

3.1.1 Circuit Equations

Kirchoff's voltage law (KVL) states that the sum of voltages in a mesh of the circuit has to be zero. Meshes are defined as the smallest possible closed loops (cycles) within a circuit (see Fig.3.1).

$$0 = -U_x + U_y + U_z \tag{3.1}$$

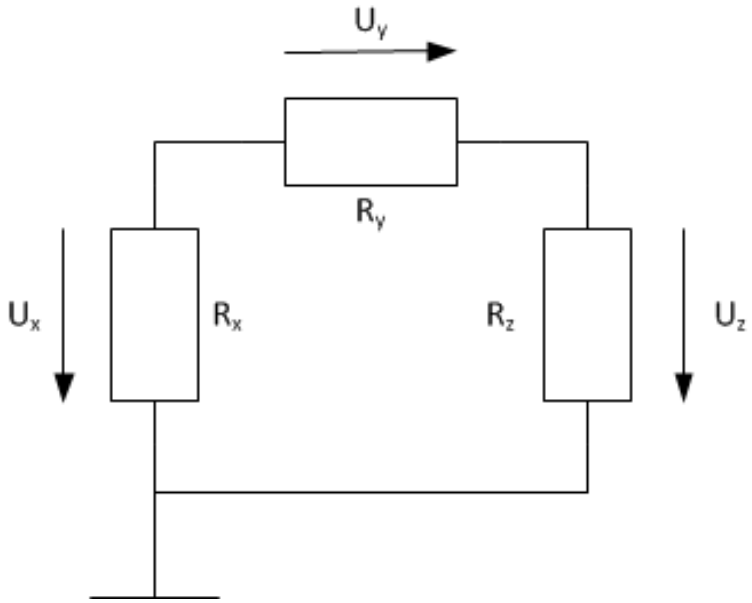

Figure 3.1: Kirchoff's Voltage Law: Mesh

Kirchoff's current law (KCL) is about nodes of currents. The sum of all incoming and outgoing currents at each node of the circuit is zero (see Fig. 3.2).

$$0 = -I_x + I_y - I_z \tag{3.2}$$

The component equations for resistors are defined by Ohm's law:

$$U_R = I_R \cdot R \tag{3.3}$$

Capacitors are energy storage elements in the circuit. The current at the capacitor is proportional to the capacity value multiplied with the change of voltage at the capacitor. In other words, the capacitor will cause current flow corresponding to its charging state. The negative sign at the capacity value changes the direction of the current flow. This is necessary because

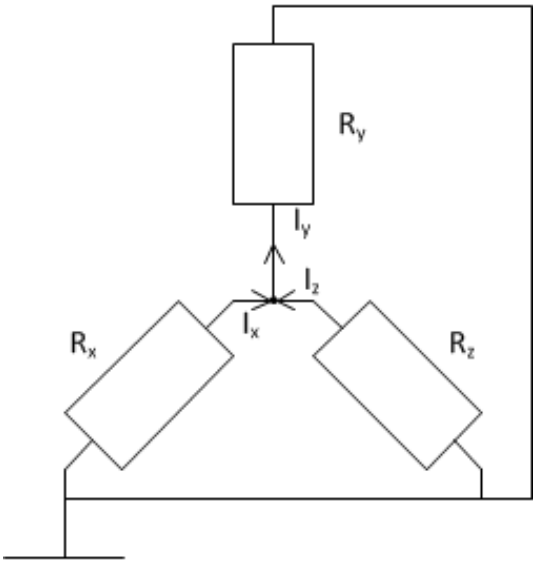

Figure 3.2: Kirchoff's Current Law: Node

capacitors drive their current in the opposite direction as the voltage, just like a voltage source element.

$$I_C = -C \cdot \frac{\partial U_C}{\partial t} \tag{3.4}$$

3.1.2 Coupling Equations

The coupling between the inductors is a major task in the modelling of an NFC matching circuit. The two parts of the component equations for inductors are called the self-inductance and the mutual inductance.

Inductors (also called coils) get inducted by electromagnetic fields. This fields create a current in the coil, the energy is stored in the magnetic field. The voltage over the inductor is defined by the inductive value times the change of induced current. This equation represents the self-inductance of the inductor:

$$U_L = L \cdot \frac{\partial I_L}{\partial t} \tag{3.5}$$

The effect of the magnetic field and the inducted current is not limited to just one inductor. Figure 3.3 shows the coupling situation for two inductors. The principle is extendible for a arbitrary number of coupled inductors. Each inductor has an influence on each other, which depends on the geometric conditions of the circuit. The influence between two inductors x and y is called mutual inductance M_{xy}:

$$M_{xy} = k_{xy} \cdot \sqrt{L_x \cdot L_y} \tag{3.6}$$

The coupling factor k_{xy} represent the geometric conditions of the circuit. Its value come from measurements and antenna simulations. For the further work with coupling factors, it is assumed that $M_{xy} = M_{yx}$ and therefore just the M_{xy} factors are needed.

The component equation for inductors has to include the self-inductance and the mutual inductance. Its parameters are:

$$M = \begin{pmatrix} L_x & M_{xy} \\ M_{yx} & L_y \end{pmatrix} \tag{3.7}$$

where the main diagonal contains the values for the inductive components. The M_{xy} entries of (3.7) correspond to the coupling of inductors L_x and L_y and can be calculated with equation (3.6).

The relation between the inductor currents and the resulting voltages can be calculated from this combined self-inductance and mutual inductance equations in M to get the needed coupling equations:

$$\begin{pmatrix} U_{Lx} \\ U_{Ly} \end{pmatrix} = \underbrace{\begin{pmatrix} L_x & M_{xy} \\ M_{xy} & L_y \end{pmatrix}}_{\mathrm{M}} \cdot \begin{pmatrix} Ip_{Lx} \\ Ip_{Ly} \end{pmatrix} \tag{3.8}$$

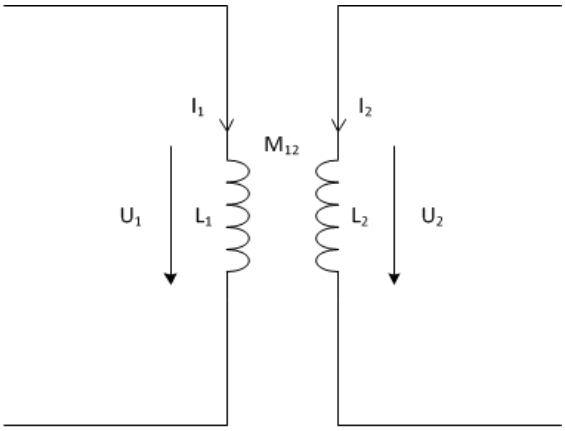

Figure 3.3: Inductive Coupling

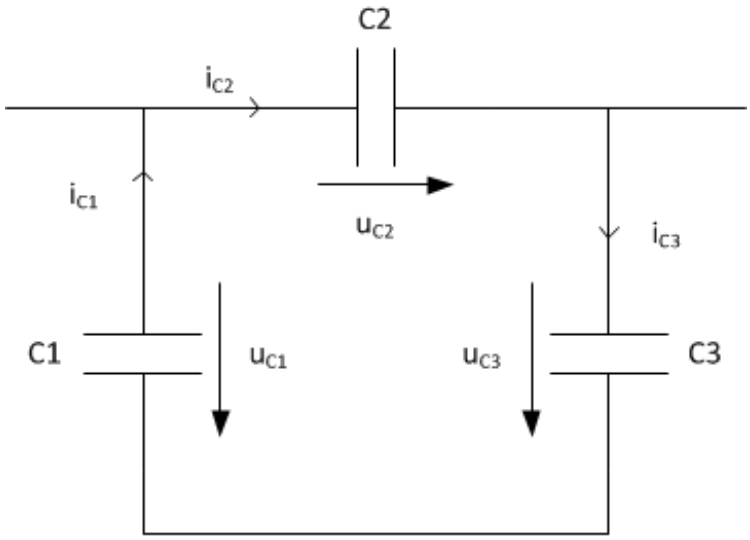

Figure 3.4: Mesh of Capacitors

3.1.3 Meshes of Capacitors and Nodes of Inductors

The state-space representation requires n linearly independent state variables. The equations have to be changed due to the fact that loops of capacitors and nodes of inductors result in linear dependent state variables. Dependent components have to be expressed via other components, so the corresponding linearly dependent state variable can be left out of the state vector.

For loops of capacitors an example is given in (3.4).

The equation for the mesh is:

$$\text{Mesh: } 0 = U_{C1} - U_{C2} - U_{C3} \tag{3.9}$$

The state variable U_{C2} is not part of the state vector (it is free to choose which linearly dependent variable to replace). A combination of the other state variables is used to express this voltage via the mesh equations in 3.9.

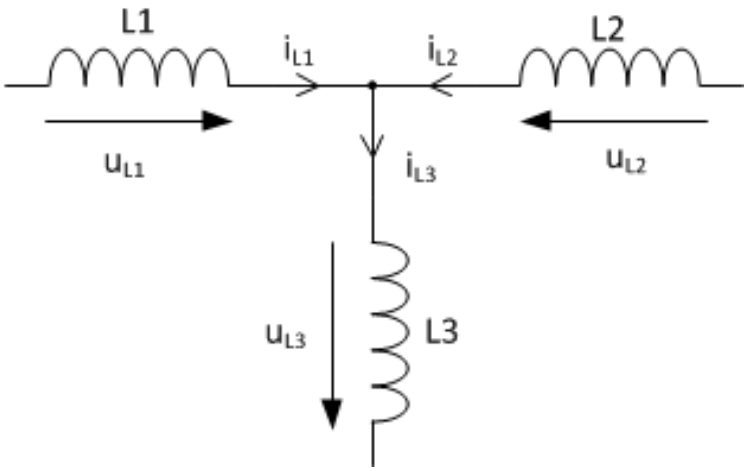

Figure 3.5: Node of Inductors

The current for the omitted capacitor is expressed via the equation in (3.10) instead of its component equation:

$$I_{C2} = C2 \cdot \frac{\partial U_{C1}}{\partial t} - C2 \cdot \frac{\partial U_{C3}}{\partial t} \tag{3.10}$$

For nodes of inductors an example is shown in Fig. 3.5:

Equation 3.11 shows the node equation with just inductors present. i_{L3} is the state variable to get rid of. The combination of other state variables leads to the representation as shown in (3.12).

$$\text{Mesh: } 0 = I_{L1} + I_{L2} - I_{L3} \tag{3.11}$$

$$U_{L3} = L3 \cdot \frac{\partial I_{L1}}{\partial t} + L3 \cdot \frac{\partial I_{L2}}{\partial t} \tag{3.12}$$

3.2 General Form State-Space Models

The mesh, node, component and coupling equations are used to form a system of differential equations to describe the dynamic behaviour of the air-interface circuit. The system of differential equations is given as a state-space model. The general form of a linear time invariant (LTI) state-space model is given by (3.13) and (3.14).

$$\frac{\partial x(t)}{\partial t} = A \cdot x(t) + B \cdot u(t) \tag{3.13}$$

$$y(t) = C \cdot x(t) + D \cdot u(t) \tag{3.14}$$

- $x(t)$ ...State vector: nx1, for n is the number of linearly independent state variables
- $u(t)$ ... Input vector: nx1
- $y(t)$ ...Output vector: nxo, for o is the number of outputs
- A ...State matrix: nxn
- B ....Input matrix: nxi, for i is the number of inputs
- C ...Output matrix: oxn
- D ...Feed-through matrix: oxi

For further information on state-space systems, please refer to
N. Dourdoumas, 2003.

3.2.1 State vector

The state vector $x(t)$ consists of state variables. Each state variable describes the current state of a part of the dynamic circuit. This saved values are used to determine the trajectory of the change of each state variable. In a software based system, we could say there is an 'update rule' for each state variable that is influenced by the input, the system behaviour and the current value stored in the state variable.

The chosen state variables are the relevant parameters of the energy storing elements in the circuit. For inductors, the relevant parameter is the

corresponding current. For capacitors, the relevant parameter is the corresponding voltage. The utilisation of these state variables is one of the main advantages over the MNA approach. In MNA, there were also state variables used which do not represent energy storage elements, this is not the case in our approach.

Some topologies can lead to linearly dependent voltages at capacitors or linear dependent currents. These constellations have to be determined and reduced. Linear dependencies would lead to non-minimal order systems. To have a unique solution, the system has to be of minimal order.

$$x(t) = \left(I_{L_1} \; I_{L_2} \; \dots \; I_{L_i} \; U_{C_1} \; U_{C_2} \; \dots \; U_{C_j} \right)^T \tag{3.15}$$

where i is the number of linear independent inductors and j is the number of linear independent capacitors.

The state vector represented with the derivatives of the inductor currents and the capacitor voltages is:

$$\frac{\partial x(t)}{\partial t} = \left(\frac{\partial I_{L_1}}{\partial t} \; \frac{\partial I_{L_2}}{\partial t} \; \dots \; \frac{\partial I_{L_i}}{\partial t} \; \frac{\partial U_{C_1}}{\partial t} \; \frac{\partial U_{C_2}}{\partial t} \; \dots \; \frac{\partial U_{C_j}}{\partial t} \right)^T \tag{3.16}$$

3.2.2 Input vector $u(t)$

$u(t)$ is the Input vector of the system. It contains all the linearly independent sources that drive the circuit and influence the output. The input vector contains the known part of the system and influences the state of the dynamic system according to the input matrix. Another name for the input vector is control vector which reflects its function. Our test systems will have less sources than the size of the input vector would allow. The input vector is filled with '0' for the undriven parts. The maximum number of independent sources is given by the number of state variables. In other words: If there are more sources than energy storing elements to describe the system state, then the behaviour will be described with linear combinations and is therefore linearly dependent.

The input vector is:

$$u(t) = (\ \cdots\ \cdots\ U_{Source_i}\ \cdots\ \cdots\ I_{Source_j}\ \cdots\ )^T \tag{3.17}$$

where i is the number of linear independent voltages sources and j is the number of linear independent current sources. The position of the entries in the vector is related to the circuit topology, entries where no source drives the circuit are 'o'.

3.2.3 Output vector $y(t)$

In the Output vector $y(t)$ the progression of the output of the system can be found. For a single output $y(t)$ is a scalar function of time. For multiple outputs y(t) is a vector with one scalar value per output as a function of time.

3.2.4 State matrix A

In each time step, the state vector $x(t)$ is updated according to the state-matrix A and the influence of the driving sources $u(t)$. The state matrix describes the behaviour of the dynamic system when it gets driven by sources. The rows of the state matrix correspond to the differential equations that describe the circuit (with respect to the state vector). Equation 3.18 shows an example for a coupling system with just two inductors and undefined number of capacitors. The dimensionality is given by the number of state variables, the entries correspond to the circuit connectivity.

$$A = \begin{pmatrix} \cdots & \frac{1}{R_1}\cdots & C_1 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ M_{12} & L2 & 0 & \cdots \\ L_1 & M_{12} & 0 & \cdots \end{pmatrix} \tag{3.18}$$

3.2.5 Input matrix B

The Input-matrix B defines the influence of the Input vector to the state vector update in each time step. The entries in B describe the contribution of the sources to the dynamic system. In other words, the input matrix scales and connects the driving sources to the dynamic system. As the size of this matrix is related to the number of inputs, it is not a matrix but a vector for a system with just one input.

3.2.6 Output matrix C

The output matrix C builds the linear combinations of the state trajectories in $x(t)$. A general approach is to take C as a unity matrix with a size according to the length of $x(t)$. With this structure, the output contains the trajectories of all state variables. The output evaluation can happen after the simulation by building the linear combination of the rows of the output vector. The size of this matrix is nxn according to the number of state variables. The size of this matrix is related to the number of outputs in the system, it is not a matrix but a vector for a system with just one output.

$$C = \begin{pmatrix} 1 & 0 & ... & ... \\ 0 & 1 & ... & ... \\ ... & ... & ... & ... \\ ... & ... & ... & 1 \end{pmatrix} \tag{3.19}$$

3.2.7 Feed-through matrix D

In general state-space models, there are sources that directly pass to the output without contributing to the dynamic system behaviour. The feed-through matrix D connects these sources to the output. The size of the feed-through matrix is related to the number of inputs and outputs in the system.

3.3 Dynamic Equations to State-Space Model

The calculation of the general form state-space model has to be done via an intermediate step. With the state vector determined (3.17 and 3.16) and the set of coupling equation (3.8), KVL and KCL equations the MATLAB symbolic extension MuPad can be used to obtain the representation:

$$0 = H\frac{\partial x(t)}{\partial t} + K \cdot x(t) + L \cdot u(t) \tag{3.20}$$

where H (nxn) consists of the components related to $\frac{\partial x(t)}{\partial t}$ state vector (dynamic behaviour), K (nxn) contains the $x(t)$ related the equation parts (static behaviour), L (nx1) corresponds to the sources in the system and therefore includes all driving voltages and currents.
This form is related to the general form of an LTI system as presented in (3.13) and (3.14). Multiplied by the inverse of the matrix H, the general form can be obtained:

$$A = H^{-1} \cdot K \tag{3.21}$$

$$B = H^{-1} \cdot L \tag{3.22}$$

where the inverse of H exists, if H is a nxn matrix with full rank. This can be assumed due to the fact that the state vector consists of one independent state variable per energy saving component and linear dependencies of capacitors and inductors are already removed. For details on this approach, see Muehlmann, 2013.

3.4 A Small Example: ISO-Setup Modelling

The ISO-setup (Fig. 3.6) is a simulation and measurement setup to verify the ISO compliance of the card device. For this purpose, the ISO wave-shapes

are used to shape the ideal RF signal and test the communication quality with this non-ideal wave-shapes. As an example of the application of this modelling approach, the so called ISO-setup (as shown in Fig. 3.6 is used.

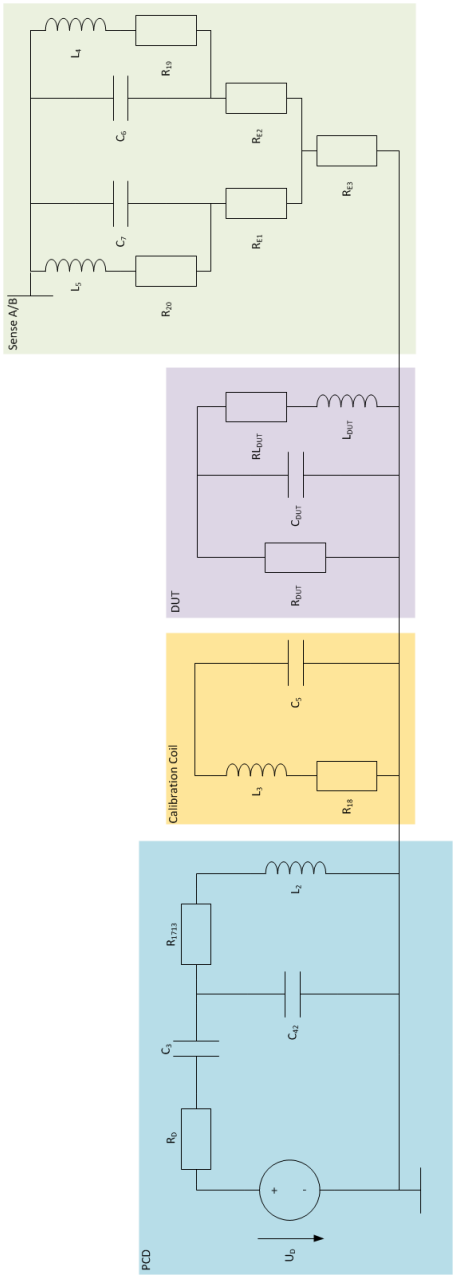

Figure 3.6: ISO-setup

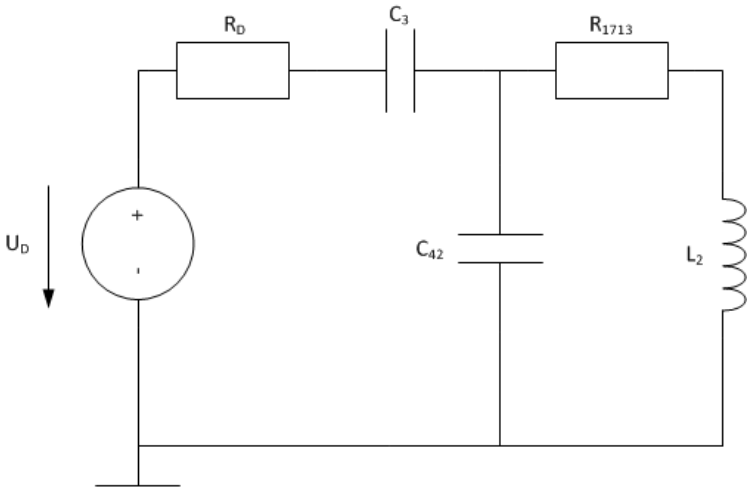

Figure 3.7: ISO-setup PCD part

3.4.1 Circuit to Equations

For the following calculations, the currents and voltage labels are corresponding to the component names.In a first step we derive the KCL (3.2) and KVL (3.1) equations:

PCD Circuit Part (Fig. 3.7)

$$\begin{aligned} \text{Node 1: } 0 &= -I_{C_3} + I_{L_2} + I_{C_{42}} \\ &= -C_3 \cdot \dot{U_{c_3}} + I_{L_2} + C_{42} \cdot \dot{U_{C_{42}}} \end{aligned} \tag{3.23}$$

$$\begin{aligned} \text{Mesh 1: } 0 &= -U_D + U_{R_D} + U_{C_3} + U_{C_{42}} \\ &= -U_D + I_{C_3} \cdot R_D + U_{C_3} \end{aligned} \tag{3.24}$$

$$\begin{aligned} \text{Mesh 2: } 0 &= -U_{C_{42}} + U_{R_{1713}} + U_{L_2} \\ &= -U_{C_{42}} + I_{L_2} \cdot R_{1713} + U_{L_2} \end{aligned} \tag{3.25}$$

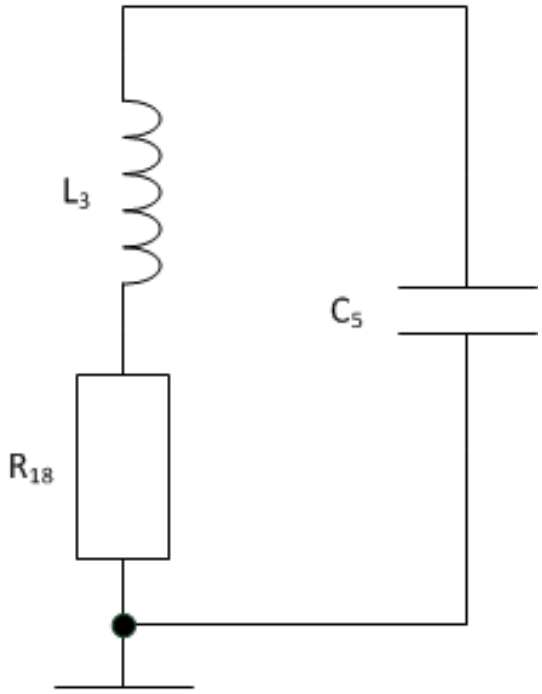

Figure 3.8: ISO-setup Calibration Coil part

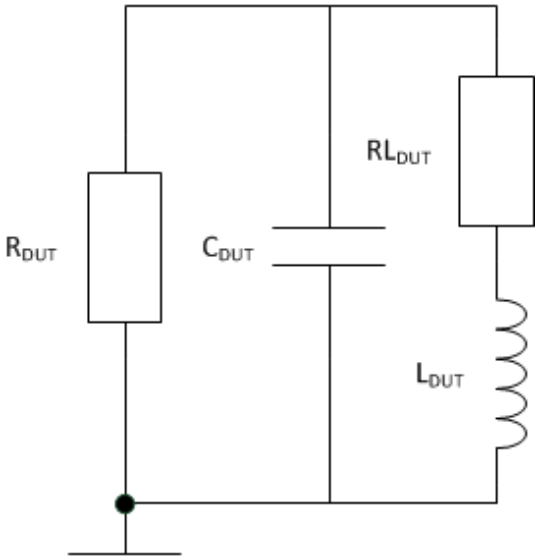

Figure 3.9: ISO-setup DUT part

Calibration Coil Circuit Part (Fig. 3.8)

$$\begin{aligned} \text{Mesh 1: } 0 &= U_{R_{18}} + U_{L_3} + U_{C_5} \\ &= I_{L_3} \cdot R_{18} + U_{L_3} + U_{C_5} \end{aligned} \tag{3.26}$$

$$\begin{aligned} \text{Node 1: } 0 &= -I_{L_3} + I_{C_5} \\ &= -I_{L_3} + C_5 \cdot \dot{U_{C_5}} \end{aligned} \tag{3.27}$$

DUT Circuit Part (Fig. 3.9)

$$\begin{aligned} \text{Mesh 1: } 0 &= U_{L_{DUT}} + U_{L_{DUT}} + U_{R_{L_{DUT}}} \\ &= U_{L_{DUT}} + U_{L_{DUT}} + I_{DUT} \cdot R_{L_{DUT}} \end{aligned} \tag{3.28}$$

$$\begin{aligned} \text{Node 1: } 0 &= -I_{L_{DUT}} + I_{C_{DUT}} + I_{R_{DUT}} \\ &= -I_{L_{DUT}} + C_{DUT} \cdot \dot{U_{C_{DUT}}} + \frac{U_{C_{DUT}}}{R_{DUT}} \end{aligned} \tag{3.29}$$

Sense A/B Coil Circuit Part (Fig. 3.10)

$$\begin{aligned} \text{Mesh 1: } 0 &= U_{L_4} + U_{C_6} + U_{R_{19}} \\ &= U_{L_4} + U_{C_6} + I_{L_4} \cdot R_{19} \end{aligned} \tag{3.30}$$

$$\begin{aligned} \text{Mesh 2: } 0 &= U_{L_5} + U_{C_7} + U_{R_{20}} \\ &= U_{L_5} + U_{C_7} + I_{L_5} \cdot R_{20} \end{aligned} \tag{3.31}$$

$$\text{Mesh 3: } 0 = -U_{C_7} - U_{R_{E1}} + -U_{C_6} - U_{R_{E2}} \tag{3.32}$$

$$\begin{aligned} \text{Node 1: } 0 &= -I_{R_{E3}} + I_{R_{E1}} + I_{R_{E1}} \\ &= -I_{R_{E3}} + I_{L_4} + I_{C_6} + I_{L_5} + I_{C_7} \\ &= -I_{R_{E3}} + I_{L_4} + C_6 \cdot \dot{U_{C_6}} + I_{L_5} + C_7 \cdot \dot{U_{C_7}} \end{aligned} \tag{3.33}$$

3.4.2 Coupling Equations

As there are 5 inductors present in the ISO-setup, the coupling Matrix is 5x5 in dimensions. The general form given in (3.8) becomes:

$$\begin{pmatrix} U_{DUT} \\ U_{L_2} \\ U_{L_3} \\ U_{L_4} \\ U_{L_5} \end{pmatrix} = \begin{pmatrix} L_1 & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{12} & L_2 & M_{23} & M_{24} & M_{25} \\ M_{13} & M_{23} & L_3 & M_{34} & M_{35} \\ M_{14} & M_{24} & M_{34} & L_4 & M_{45} \\ M_{15} & M_{25} & M_{35} & M_{45} & L_5 \end{pmatrix} \cdot \begin{pmatrix} Ip_{DUT} \\ Ip_{L_2} \\ Ip_{L_3} \\ Ip_{L_4} \\ Ip_{L_5} \end{pmatrix} \tag{3.34}$$

State vector

To bring all the KVL equations, KCL equations and the coupling together in one system matrix formulation, the state variables have to be determined first. The currents through inductors and the voltages over capacitors are the appropriate variables for this. For the ISO-setup the state vector from (3.17) is:

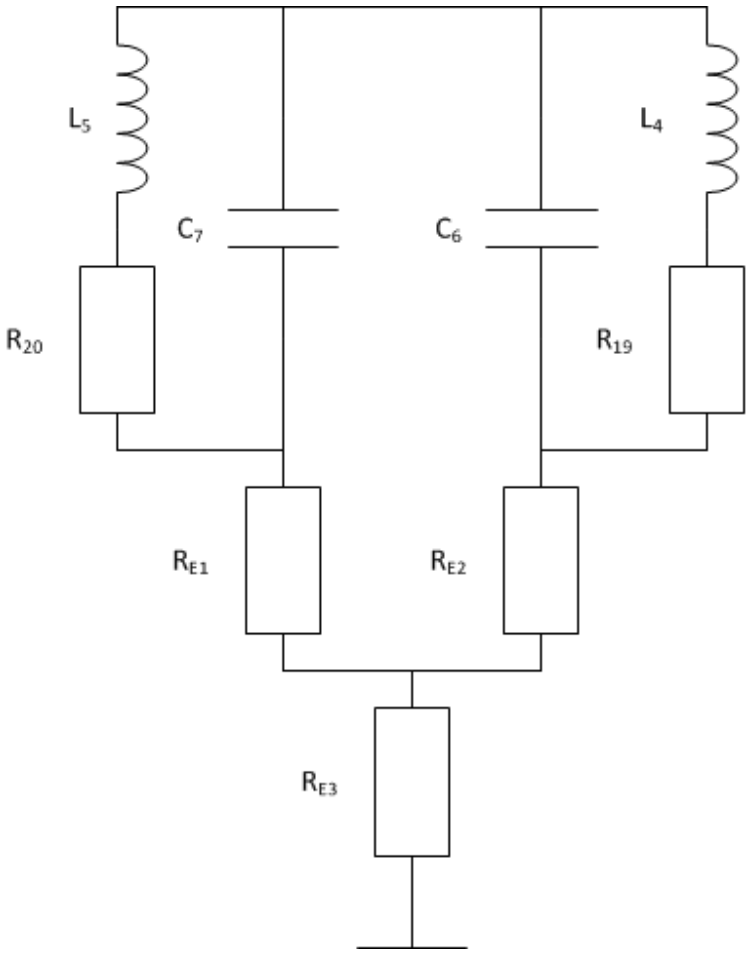

Figure 3.10: ISO-setup Sensecoil A/B part

$$x(t) = \left(I_{DUT}\ I_{L_2}\ I_{L_3}\ I_{L_4}\ I_{L_5}\ u_{C_{DUT}}\ u_{C_3}\ u_{C_{42}}\ u_{C_5}\ u_{C_6}\ u_{C_7} \right)^T \tag{3.35}$$

and according to (3.16) the derivative becomes:

$$\frac{\partial x(t)}{\partial t} = \left(Ip_{DUT}\ Ip_{L_2}\ Ip_{L_3}\ Ip_{L_4}\ Ip_{L_5}\ up_{C_{DUT}}\ up_{C_3}\ up_{C_{42}}\ up_{C_5}\ up_{C_6}\ up_{C_7} \right)^T \tag{3.36}$$

State-Space Model

Using the coupling, KVL and KCL equations and the symbolic calculation Toolbox (MuPad) for MATLAB (used due to convenience reasons), the system matrices are found to be:

$$
H = \begin{pmatrix}
M_{14} & M_{24} & M_{34} & L_4 & M_{45} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (R_{E2})*(C_6+(C_6*(R_{E1}))/(R_{E3}))+C_6*(R_{E1}) & 0 \\
M_{15} & M_{25} & M_{35} & M_{45} & L_5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C_6-(C_6*(R_{E1}))/(R_{E3}) & -C_7 \\
M_{13} & M_{23} & L_3 & M_{34} & M_{35} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -C_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & C_3*R_D & 0 & 0 & 0 & 0 \\
M_{12} & L_2 & M_{23} & M_{24} & M_{25} & 0 & C_3 & -C_2-C_4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & C_{DUT}*R_{DUT} & 0 & 0 & 0 & 0 \\
L_1 & M_{12} & M_{13} & M_{14} & M_{15} & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{3.37}
$$

$$
K = \begin{pmatrix}
0 & 0 & 0 & 0 & -R_{19} & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & R_{E1}+(R_{E2})*((R_{E1})/(R_{E3})+1) & 0 & 0 & 0 & 0 & 0 & 0 & -(R_{E2})/(R_{E3})-1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -R_{20} & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -(R_{E1})/(R_{E3})-1 & -1 & 0 & 0 & 0 & 0 & 1/(R_{E3}) & 0 & 0 \\
0 & 0 & 0 & -R_{18} & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -R_{13}-R_{17} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -R_{DUT} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-R_{DUT} & -R_{LDUT} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0)
\end{pmatrix}
\tag{3.38}
$$

$$L = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -U_D \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.39}$$

The operation of inverting H and creating the matrices A and B from the already calculated H, K and L is performed according to (3.21) and (3.22).

Matrices are not shown here as it is straight forward and brings no further information here.

3.5 Additional Modelling Steps

The following features represent additional steps in the modelling process, needed for special circuit topologies and applications.

3.5.1 Continuous-Time to Discrete-Time Conversion

The discussed system is given in continuous-time representation. For some applications a discrete-time model is sufficient in accuracy and beneficial in simulation time. In particular for block processing this approach is beneficial.

The discretization of the system is a reasonable step:

$$x[n+1] = A_{disc} \cdot x[n] + B_{disc} \cdot u[n] \tag{3.40}$$

$$y[n] = C_{disc} \cdot x[n] + D_{disc} \cdot u[n] \tag{3.41}$$

To bring the system in the discrete form a sampling rate has to be defined. For our NFC systems the sampling rate f_s can be defined as multiples of the carrier frequency $13.56MHz$ (oversampling).

The discrete state matrix A_{disc} can be determined as the exponential matrix of A, sampled at the discrete points defined by the time step $\tau_s = \frac{1}{f_s}$.

$$A_{disc} = e^{(A \cdot \tau_s)} \tag{3.42}$$

B_{disc} can be calculated as:

$$B_{disc} = \left(\int_{\tau=0}^{\tau_s} e^{(A \cdot \tau)} d\tau \right) \cdot B$$

which is equal to the form:

$$B_{disc} = A^{-1} \cdot (A_{disc} - I) \cdot B \tag{3.43}$$

for the matrix A needs to be invertible.

The output matrix C and the feed-through-matrix D show the same structure in both continuous and discrete time:

$$C_{disc} = C \tag{3.44}$$

$$D_{disc} = D \tag{3.45}$$

3.5.2 Time-Varying System Modelling

For application with a communication from PICC to PCD, the passive load modulation needs to be modelled. As mentioned before, this is a linear time-varying function where the PICC circuit switches between two different circuit conditions.

The mathematical definition of a time-varying state-space model is given in equations (3.46 and 3.47). In comparison to the definition of the time-invariant state-space system in (3.13 and3.14), the state matrix, the input matrix, the output matrix and the feed-through matrix are time-varying.

Time-varying behaviour is modelled as one system description per PICC condition. This means, there is one state-space system model for the PCD with the 'unmodulated' PICC circuit and one state-space system model for the PCD with the 'modulated' PICC circuit. The terms 'unmodulated' and 'modulated' refer to the condition of the envelope for this circuit states.

The basic structure of the state matrices is similar, just some entries have different values. The state variables are the same. During simulation, the values of the state vector are preserved, therefore the system matrix can be changed without the loss of the current and voltage situations in the circuit. Figure 3.11 shows a sketch of the time-varying simulation approach.

$$\frac{\partial x(t)}{\partial t} = A(t) \cdot x(t) + B(t) \cdot u(t) \tag{3.46}$$

$$y(t) = C(t) \cdot x(t) + D(t) \cdot u(t) \tag{3.47}$$

The transient behaviour of the system at the precise moment of the switching between the two models is not taken into account. The missing transition modelling does effect the model accuracy, but only in a neglectable way. The accuracy of the model is still good enough for all desired applications of the framework.

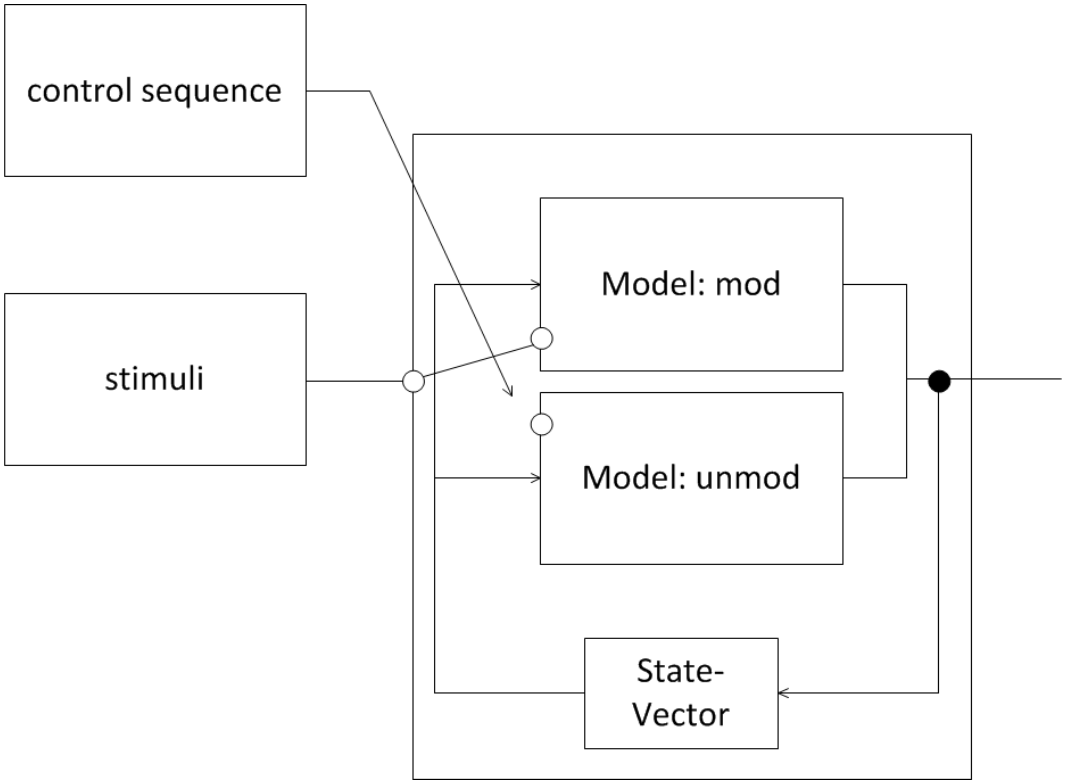

Figure 3.11: LTV Simulation Approach

4 Coupling System: Automated Model Generation

4.1 From Circuit to State-Space Model

A major task for the framework is the automated generation of the air-interface model. The manual generation of a linear system from its circuit schematic is straightforward, because it is intuitive for hand calculation. A schematic, however, is not useful when the model is created automatically, therefore the circuit description is done via netlists. Finding the node and mesh equations of the circuit is automated and takes special rules for topologies into account. The calculation of the equations is performed using the circuit, coupling and component equations. This way, a state-space representation can be found with just a circuit description in netlist form. Figure 4.1 shows an overview of the automated model generation on block level.

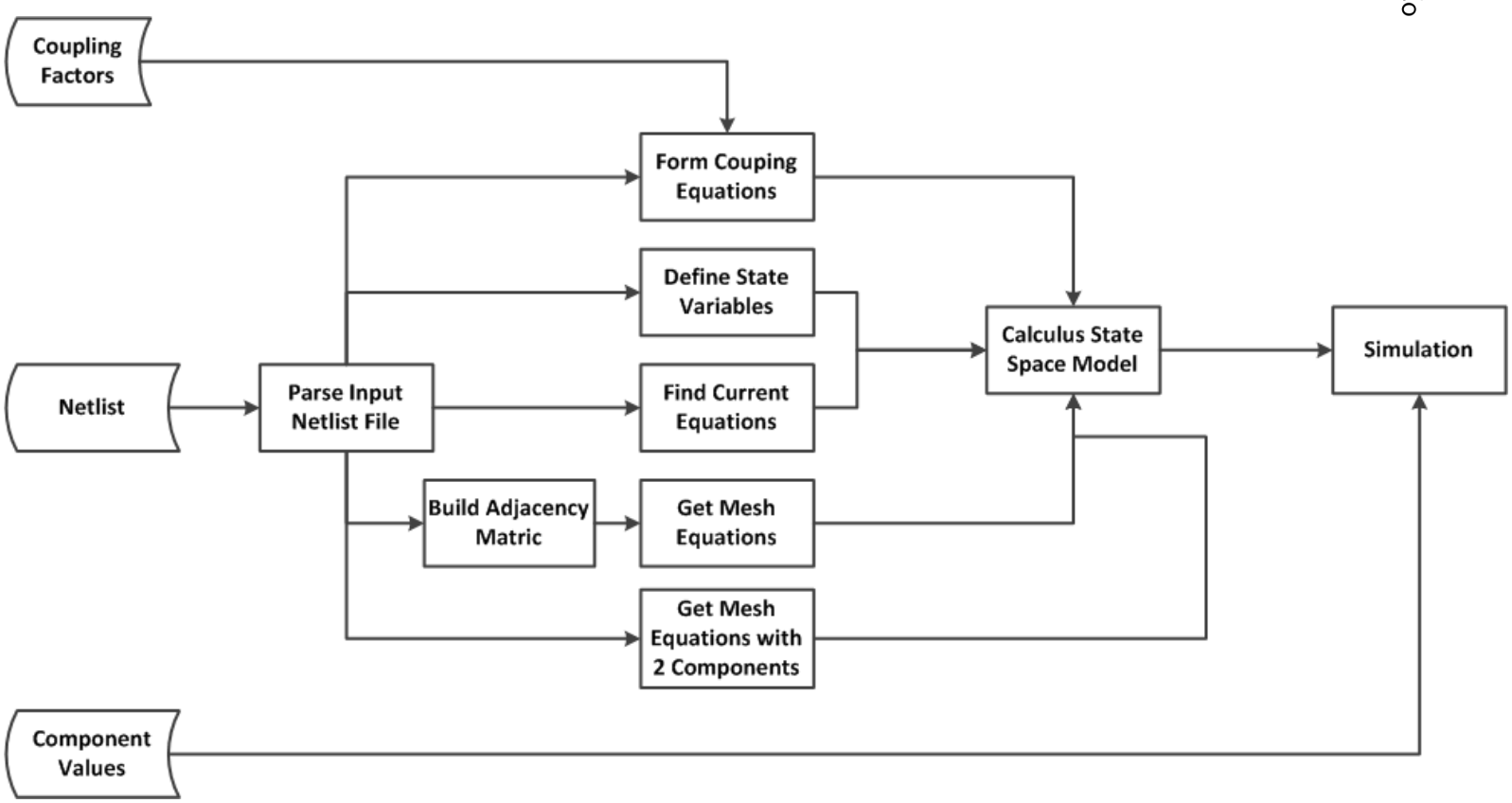

Figure 4.1: Model Generation Modul on Blocklevel

4.1.1 Inputs

The following inputs are required for the framework:

- Netlist(s)
- Component value configuration files
- Point-of-operation characteristics
- Stimuli Parameters
- ISO-shape database
- Operation mode configuration flags
- Configuration flags
  - Linear-time-varying simulation
  - Limiter optimisation
  - Parameter value configuration file inclusion
  - Shaping enable
  - Write stimuli to file enable
  - Read spice data from file enable

Netlists are a common way to describe circuit topologies. Each component is listed by name, a value and the two nodes it is connected to. By this minimal set of informations, a circuit with all the components' connections can be described. Coupling informations can also be represented in netlists. The coupling is treated like a component. The two coupled inductors represent the nodes of the connection, the value corresponds to the coupling factor between the inductors.

Configuration files can include different component values and coupling factors. If a configuration file (or multiple configuration files) is provided, the component values and coupling factors from the netlist get overwritten by the values from the configuration file.

To optimize a circuit for a certain point-of-operation, the combination of resistor and capacitor values is stored in a data container. The optimization algorithm determines the correct point-of-operation and takes the values for these components from the data container.

The stimuli generation is can be parametrized by a set of configuration flags, data rate, oversampling rate, data word (in hex) and communication type are included in the flag options.

The ISO-shapes are used while stimuli generation (if the shape enable configuration flag is set), to generate an input signal with special conditions and therefore ISO use-case significance. A database contains all the predefined shapes in container form.

4.2 Circuit Topology Description: Netlists

A netlist consists of one entry per component. Additionally, there is one coupling entry for each coupled pair of inductors. A netlist entry consists of the name, value and the connection nodes of the component. The name has to be unique and also defines the type of the component (V, I, R, L, C, k).

Each component comes with a value without unit. The unit is given implicitly by the component type. For sources the value field is set to '1' in the parser, hence the sources are driven by the simulation part of the framework. Therefore the driving input voltage/current is scaled with '1', additional scaling is implemented and can be used for a test scenario where the input current has to be scaled due to the voltage on a configuration pin in the circuit.

The connection in the circuit is defined by two node fields per entry. All nodes in the circuit get a number. The ground node is defined to be '0'. The component is connected between the nodes given in the entry. For coupling entries the two fields don't contain the connected nodes, but the two connected inductors which are described by the entry.

The direction of voltages and currents is implicitly given by the order of the connected nodes. Rotating a component (exchange the two connection fields in the entry) flips the sign of the variable in the equations, but the 'real' direction is still given by the circuit behaviour itself. Therefore it is not important in which direction a component is presented in the netlist, but the relations of the components to each other and the voltage to current relation on each component has to be well defined. Voltage sources and capacitors are

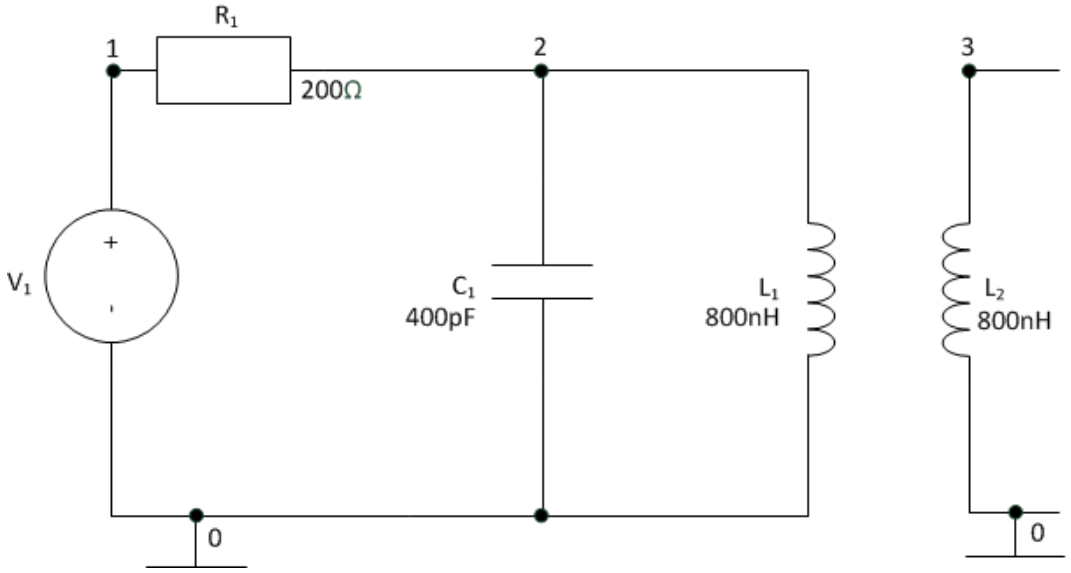

Figure 4.2: Example Circuit for Netlist Description

defined to have voltage and current into opposite direction. For inductors, resistors and current sources, the voltage and current directions are the same.

Figure 4.2 shows an example circuit part, the corresponding netlist is shown in the following Table 4.1:

4.3 Kirchoff's Laws: Finding Current Nodes

With the circuit topology given as a netlist, the investigation of the circuit equations can start. First of all, the equations for the currents at the circuit nodes are calculated. A node in the circuit is defined as a uniquely named port which connects components or sources. The KCL equation for each node can be found by adding the currents for all components or sources connected to the particular node. If there are just two components connected to a node, the KCL equation implies equality of the currents. One equation

Name	Node 1	Node 2	Value
V_1	1	0	1
R_1	1	2	200
C_1	2	0	$400e^{-12}$
L_1	2	0	$800e^{-12}$
L_2	3	0	$800e^{-12}$
K_{12}	L_1	L_2	0,05
...	...	...	...

Table 4.1: Example Netlist

per node can be found, the ground node has to be taken into account as well. Each node is unique in the circuit (if the netlist is not corrupted), no replications of an equation can occur.

4.4 Kirchoff's Laws: Finding Voltage Meshes

For a human it is easy to find the meshes within a circuit. For a machine, the problem has to be defined first. A mesh is a closed connection from one node of a circuit to itself. More precisely, it is the shortest cycle in the circuit that connects a node to itself. The connection of nodes can be seen as a graph. From graphic theory there are algorithms available to find shortest paths between nodes. Finding a mesh is a special case of finding a shortest path. Floyd Warshall's algorithm is chosen to find the shortest paths. Shortest paths from one node to itself are omitted in the general implementation of the algorithm, as it would just give trivial solutions. To find meshes and not trivial solutions, the algorithm has to be modified.

First, the general implementation has to be explained. An adjacency matrix has to be created as a description of the circuit graph. The adjacency matrix is of size (nxn) where n is the number of nodes in the circuit (including the ground node). For each two nodes that are directly connected, the matrix has a '1' entry. If no direct connection exists, a '0' fills the place in the matrix. The graph is undirected, so the adjacency matrix is symmetrical.

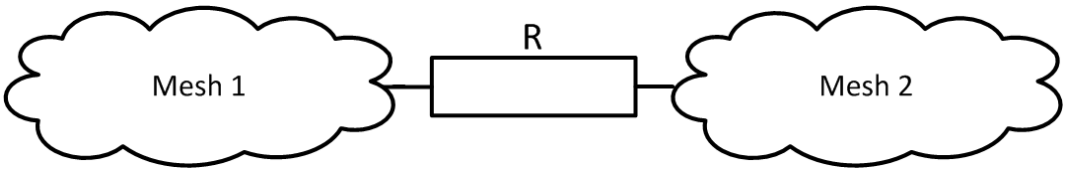

Figure 4.3: A 'bridge' component between two meshes

The Floyd Warshall's algorithm needs to have some restrictions on the length of the paths, so no meshes with less than three components can be found. Mesh equations with just two components are found separately by just comparing the netlist entries. These equations are called 'parallel equations' since they represent parallel circuit connections only. Two components which are parallel to each other share the same two connection nodes, no matter in which order. The order just influences the implicit current/voltage direction which is still consistent with the other components.

To use the shortest cycles as mesh equations is an efficient way to build up the mesh equations, but there are special cases which might leave a mesh equation out of the shortest cycles. If a component is a 'bridge' between two close cycles (as shown in Fig. 4.3) there will be no equation for this component, since its two nodes are both already part of a mesh. For this case, a deep-first-search (DFS) has to be applied for just that component. This kind of mesh finding is not very efficient and just practicable for finding a mesh for this special case.

4.4.1 Floyd Warshall's Algorithm

The Algorithm contains two essential parts: The shortest path optimisation and the path reconstruction. For the path optimisation part, an adjacency matrix has to be defined. This adjacency matrix is the input for the Floyd Warshall's algorithm and represents the initial connections of the distance matrix. This distance matrix is initialized with the connection weight for each direct connection, an infinite value for paths to be optimized and '0' are used for connections not to be optimized. In the general implementation

the unoptimized values are the diagonal elements of the matrix, in other words the connections of a node to itself.

For each step, the algorithm tries to replace the distance between the the two nodes with a shorter connection, which is a combination of two other connections. If the connection via the other two nodes is shorter, the value in the connection matrix gets replaced, otherwise the old value remains. The algorithm checks all possible connections and all possible replacements for this connection.

Here the pseudo code for the general implementation of the algorithm is given. The variables k, i and j iterate over all possible node connections and connection replacements. The function 'dist' represents the distance between the two nodes that are given as parameters.

```
for k from 1 to #Nodes
 for i from 1 to #Nodes
  for j from 1 to #Nodes
   if dist[i][j] > dist[i][k] + dist[k][j]
    dist[i][j] = dist[i][k] + dist[k][j]
   end if
  end for
 end for
end for
```

The runtime of this algorithm is $\mathcal{O}(n^3)$ for n is the number of nodes. For the general Floyd Warshall's algorithm the path reconstruction can be done after the path optimisation is done. Just the distance matrix is needed to find the nodes which are included in the shortest paths. As this approach is not applicable after the modifications, it is not explained any further.

4.4.2 Modifications for the Algorithm

To find the meshes in a circuit, the smallest cycles without repeated paths have to be found. With some modifications in the initial values of the connection matrix and some additional constraints, the Floyd Warshall's algorithm provides circuit meshes.

The initial values for the distance of each note to itself is normally set to '0'. For this approach an infinite distance has to be used to tell the algorithm to find a better path. The additional constrains contain a restriction in the length of a path to itself. A path from a node to itself has to be longer than two hops, otherwise it is the trivial solution. A length of two hops is the result of finding a 'path' over one component and going back over the same or a parallel component.

For connections between different nodes, paths of a minimal length of 2 are sufficient. For connections of a node to itself, paths of a minimal length of 3 is needed to avoid the trivial solution.

The modified pseudo code includes the distinction between a pair of different nodes and a 'pair' that consists of just one node to itself:

```
for k from 1 to #Nodes
 for i from 1 to #Nodes
  for j from 1 to #Nodes
   if i == j
    for l from 1 to #Nodes
     if dist[i][j] > dist[i][k] + dist[k][l] + dist[l][j]
      dist[i][j] = dist[i][k] + dist[k][l] + dist[l][j]
     end if
    end for
   else
        if dist[i][j] > dist[i][k] + dist[k][j]
         dist[i][j] = dist[i][k] + dist[k][j]
        end if
   end if
  end for
 end for
end for
```

Due to the modified update rule for the algorithm, the path reconstruction has to be done in a different way than for the general implementation. The connections have to be saved additionally in a connection matrix. In this matrix all the included nodes are saved explicitly for each connection. Therefore reconstruction of the path is not needed. The analyzed runtime of

the algorithm remains the same. The additional check $(i == j)$ will cause number of node times a loop over all Nodes. Therefore a term $n \cdot n$ is added to $\mathcal{O}(n^3)$. $\mathcal{O}(n^3 + n^2) = \mathcal{O}(n^3)$.

4.4.3 Extensions for the Mesh Algorithm: DFS

If a component is a 'bridge' between two meshes, its nodes are both part of shortest cycles and there is no mesh with that component included. The component equations have to be checked if all components are present in the mesh equations. If not present, a DFS search for a path through the connection graph of the circuit is performed, to find a mesh that connects to the component's nodes. The path through the connection graph is the resulting mesh equation. The DFS is implemented as a recursive algorithm with a start node as current node and an end node:

```
if endNode == current Node
 add current node to path
 return path
else
 add current node to path
 get neighbors of(current node)
 forall neighbors
   recursive function call dfs
 select shortest path of all returned dfs calls
```

To use the DFS algorithm just as a support for the minimum cycles approach is the best way to go. To find all mesh equations with the DFS approach would lead to a brute force like algorithm with many redundant equations and a long runtime.

For more information on DFS or Floyd Warshall's Algorithm, have a look at T. Cormen, 1990, Franz Aurenhammer, 1999 and Burfield, 2013.

4.5 Equations for Components & Couplings

Equations for resistors and capacitors are defined as (3.3) and (3.4). The equations are created directly from the netlist. The name of the component (which also defines the type) defines the equation that is used.

Inductors are represented with one equation per component. The coupling of the inductors is modelled into these equations (see (3.8)), so the length of each equation is related to the overall number of inductors in the circuit. The coupling factors can be parsed from the netlist and directly converted to M_{xy} parameters for the equations as in (3.6).

4.6 Find and Replace Meshes of Capacitors and Nodes of Inductors

Even if all equations are available at this point, the order of the system is not necessarily minimal. Certain circuit topologies (meshes of capacitors, nodes of inductors) lead to linearly dependent equations. The linearly dependent equations get solved here by replacing the depended state variables according to (3.17) and (3.16). To find a mesh of capacitors, all mesh equations get checked if just capacitors are included in the equation. For nodes of inductors all node equations get checked if they consist of inductors only. The old representation of the dependent variables gets deleted from the list of equations because these equations bring redundancy into the system.

4.7 Solving Equations for State-Space Model

The whole set of equations has to be used to create a state-space model of order n where n is the number of linearly independent state variables. The equations that are best for this approach are the component equations for all capacitors and inductors. These equations still contain variables which are not expressible in the state-space representation. All these variables have to be replaced by different equations. Variables that have to be replaced are:

- Currents of voltage sources
- Voltages of current sources
- Currents of capacitors
- Voltages of inductors
- Currents & Voltages of resistors

The variables can be replaced by reformulated node and mesh equations until just state variables are used in the equation. This substitution process takes care that each variable gets replaced properly in all other equations and is used just once for this purpose.

The state vectors are defined as shown in (3.17) and (3.16). The Matlab extension MuPad is used to express the n equations as a state-space system with respect to the state vector. Hence the system is already of minimal order, the inversion of the matrix H and the calculation of A and B can be calculated as shown in (3.21) and (3.22).

The state-space model is saved in two ways: A parametrized version where the component values from the netlist get evaluated to have a pure numerical system model and a symbolic model where the parameters can be added later using configuration files or optimisation algorithms.

5 Simulation & Evaluation

5.1 Simulation

The simulation module contains all functionality that is needed to perform simulations for the state-space model using the created time-domain stimulus. Figure 5.1 shows the signal flow within the simulation module.

The simulation module includes simulation functions for continuous time and discrete time state-space system representations. The generation of the model always is done as continuous time model, the conversion from continuous time to discrete time model is done in this module.

State-space models can be seen as multiple input, multiple output systems (MIMO) and therefore the simulation yields the results for all outputs at once.

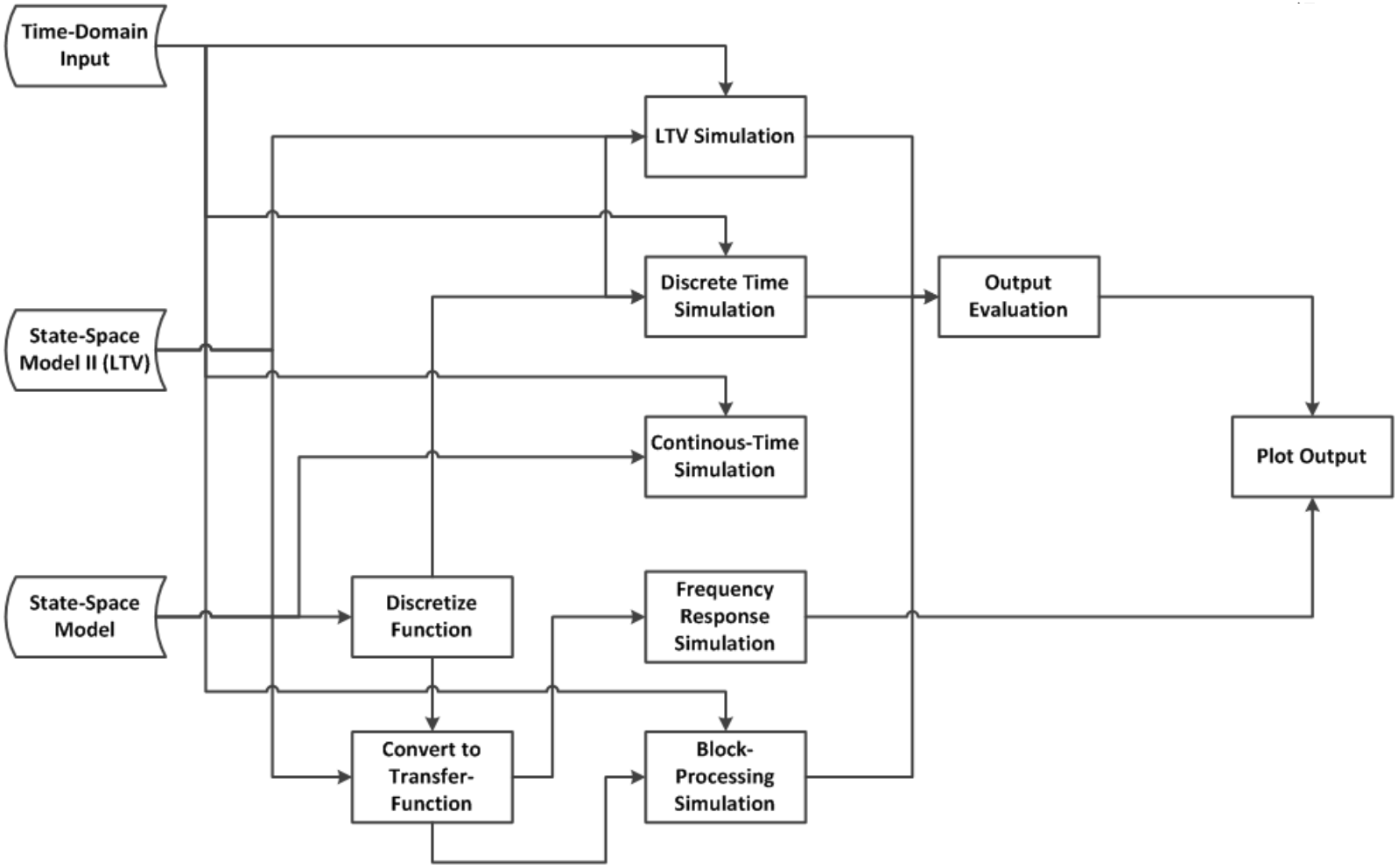

Figure 5.1: Simulation Modul on Blocklevel

5.1.1 Evaluation

All simulation outputs get saved in a container structure. Due to the structure of the chosen state-space model the whole circuit state is saved in the state variables which are voltages at capacitors and currents over inductors. To obtain the voltage or current trajectories for other components, the linear combinations of the state variables have to be found. Each other component's trajectory can be obtained as a linear combination of state variables. All equations that are used to create the model are saved with the model. These equations are used to obtain the linear combinations.

The result of this equation solving is a n dimensional vector that contains the linear combinations for the desired output variable. By multiplying this vector with the output vector, the trajectory of the desired output variable is obtained.

The result of this equation solving is a n dimensional vector that is multiplied with the output trajectory container to form the linear combination for the desired output component variable.

6 Example Applications & Results

6.1 Comparison of Spice Model vs. Matlab Model (Framework generated Model)

This example illustrates the comparison between the automatically generated Matlab model and the spice model. A comparison between the automatically generated model and the analytic model is not needed due to the fact that the spice model is the reference for the circuit behaviour.

6.1.1 Circuit

As matching scenario the ISO-setup is used. This circuit represents the reference for ISO wave-shape verification tests and is an important benchmark for most NFC systems. The ISO-setup has fixed coupling conditions. The wave-shapes of the envelope are varied for the purpose of wave-shape verification and therefore ISO compliance. Reader envelopes with wave-shaping and card envelopes with wave-shaping can be tested. For this example, the communication direction is from PCD to PICC.

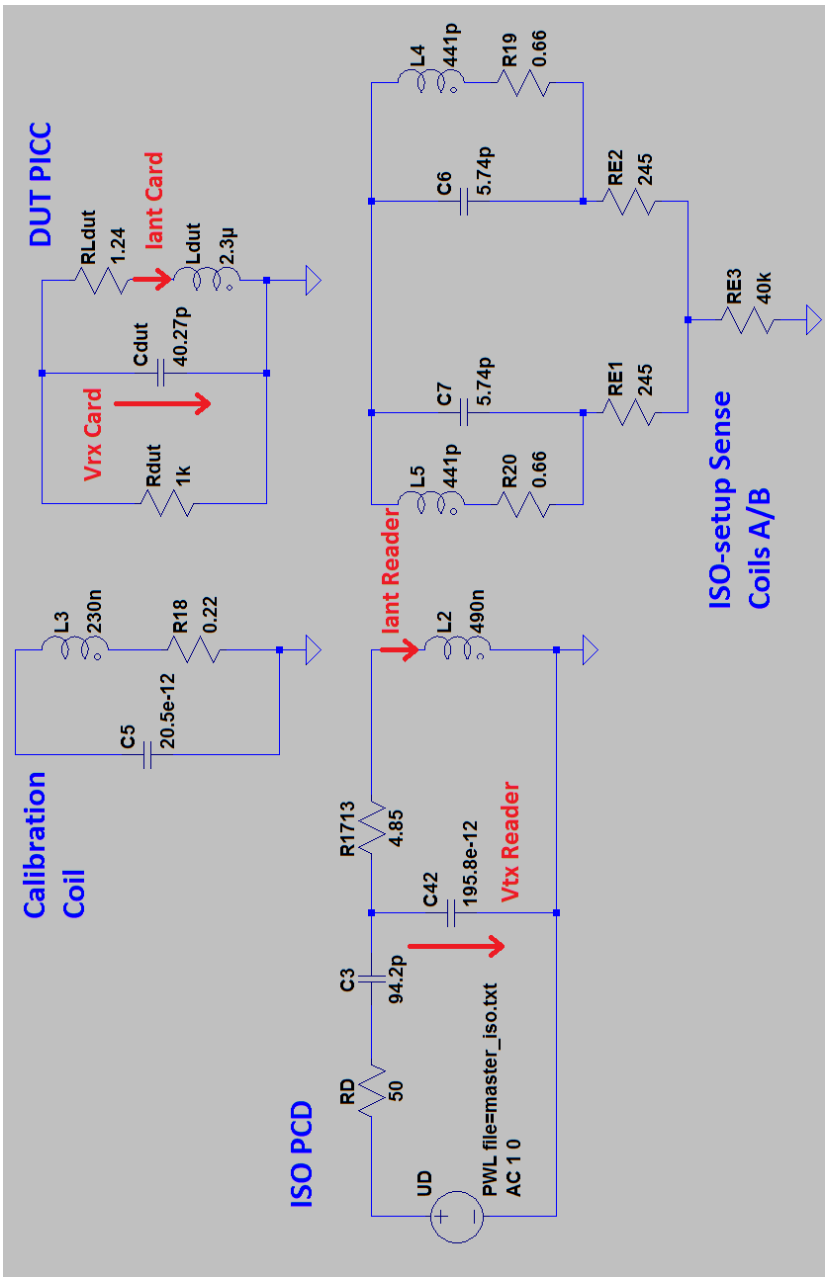

Figure 6.1: Spice schematic of ISO-setup

The spice schematic is shown in Fig. 6.1. The netlist is directly taken from the spice simulator and put into the automated model generation part of the framework.

6.1.2 Frequency Response

As a first comparison, the frequency response of the system is shown. Due to the fact that the ISO-setup is a matching network with many coupled parts, the comparison is shown for the PCD antenna to represent the behaviour.

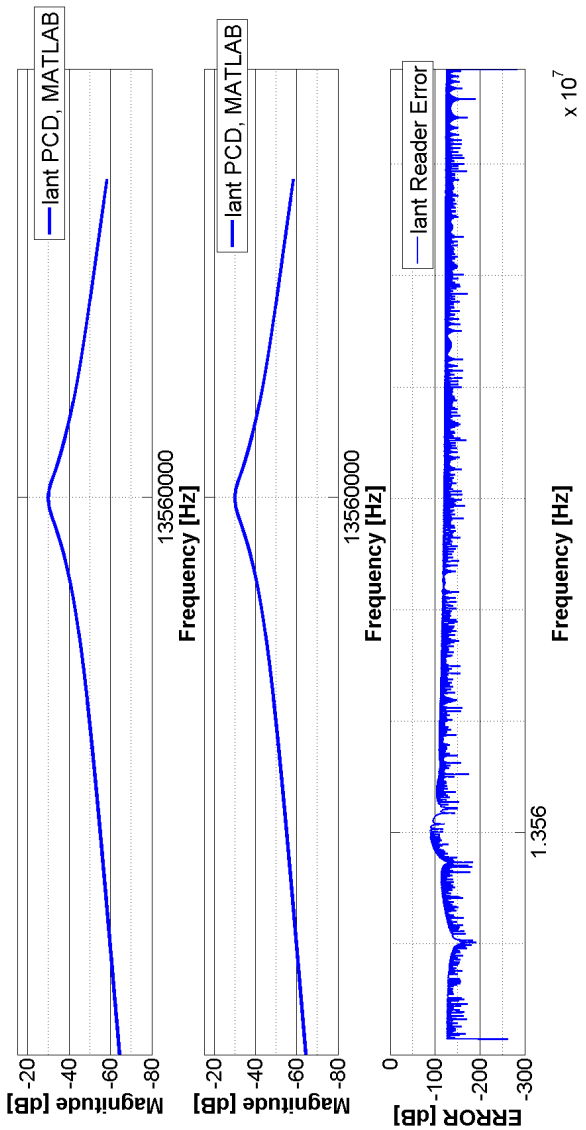

Figure 6.2: Frequency response for ISO-setup, PCD antenna

Figure 6.2 shows that the frequency response matches well for the tested frequency range of 1MHz to 60MHz. The error is calculated as the difference of the frequency responses of the Matlab and the spice model. The highest error is reached at around the tuning frequency of 13.56MHz and is below -80dB. This shows that the frequency behaviour of the two models is very similar.

6.1.3 Stimulus Generation

A 'Type A' reader envelope (106k bitrate, Modified-Miller encoding, 32x oversampling, dataword: '26') is used as stimulus for the time-domain simulation. The spice model gets the same envelope as the Matlab framework. To achieve this, the envelope is generated in Matlab and saved as a text file that is then used as a source control file in the spice simulation(also called 'PWL file').

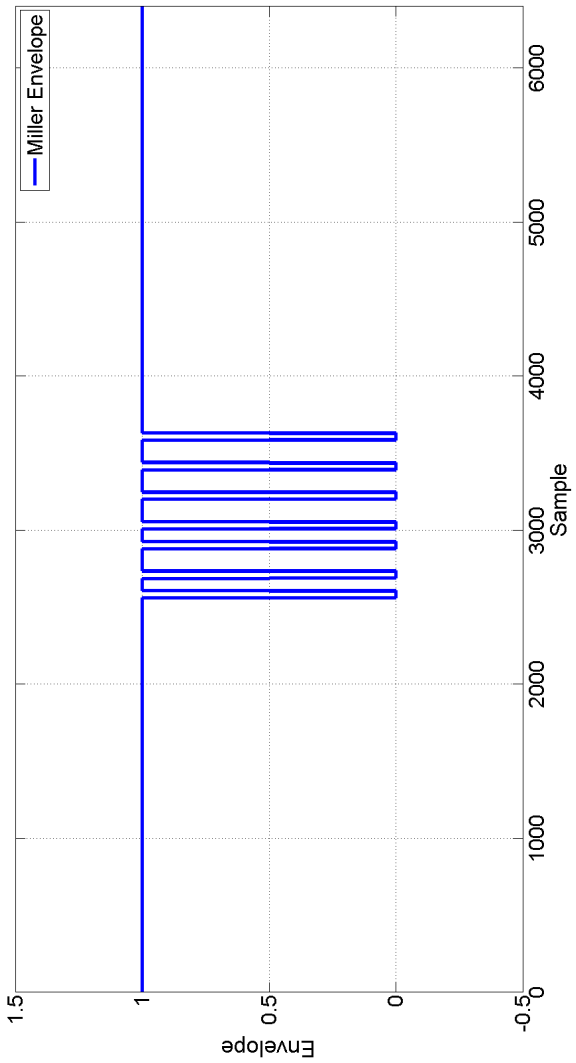

Figure 6.3: Type A Miller envelope for ISO-setup

6.1.4 Time-Domain Simulation

When characterising a coupling scenario from PCD to PICC, the relevant time-domain signals are: The PCD antenna current Iant Reader , the PICC antenna current Iant Card, the RX voltage at the PICC Vrx Card and the PCD TX voltage Vtx Reader. The antenna current Iant Reader at the PCD is directly related to the field-strength and therefore to the power of the transmission. The antenna current Iant Card at the PICC is purely inducted over the coupling and is also related to the received field-strength/power. The voltage Vrx Card at the card shows the signal that is used for reception in the card. The voltage Vtx Reader shows the signal at the transmission pin of the PCD circuit. The time-domain results for the given 'Type A' envelope is shown in Figures: 6.4, 6.5, 6.6 and 6.7.

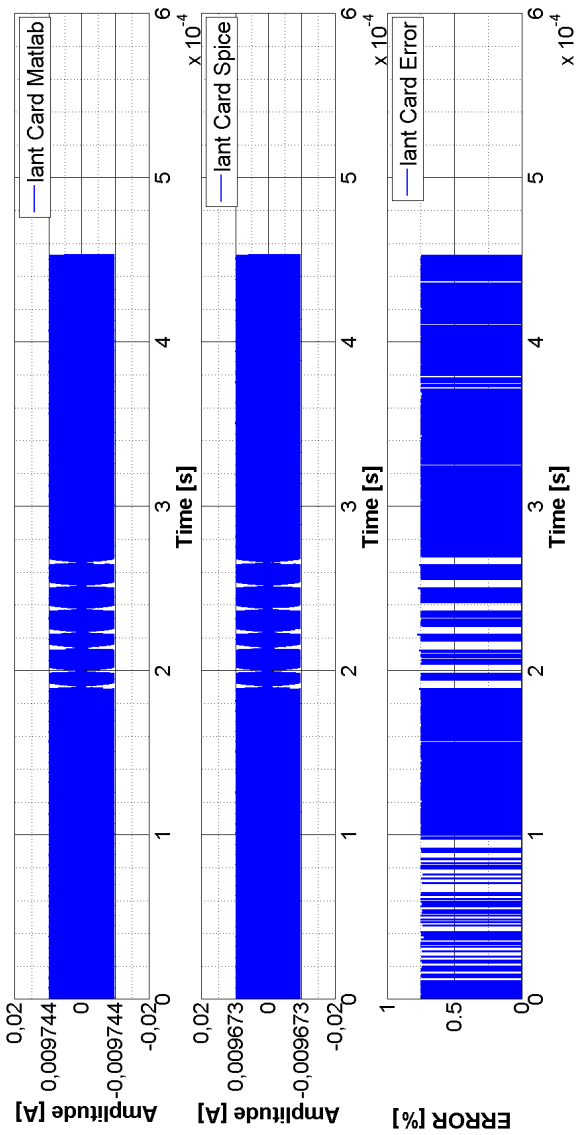

Figure 6.4: Antenna current of PICC

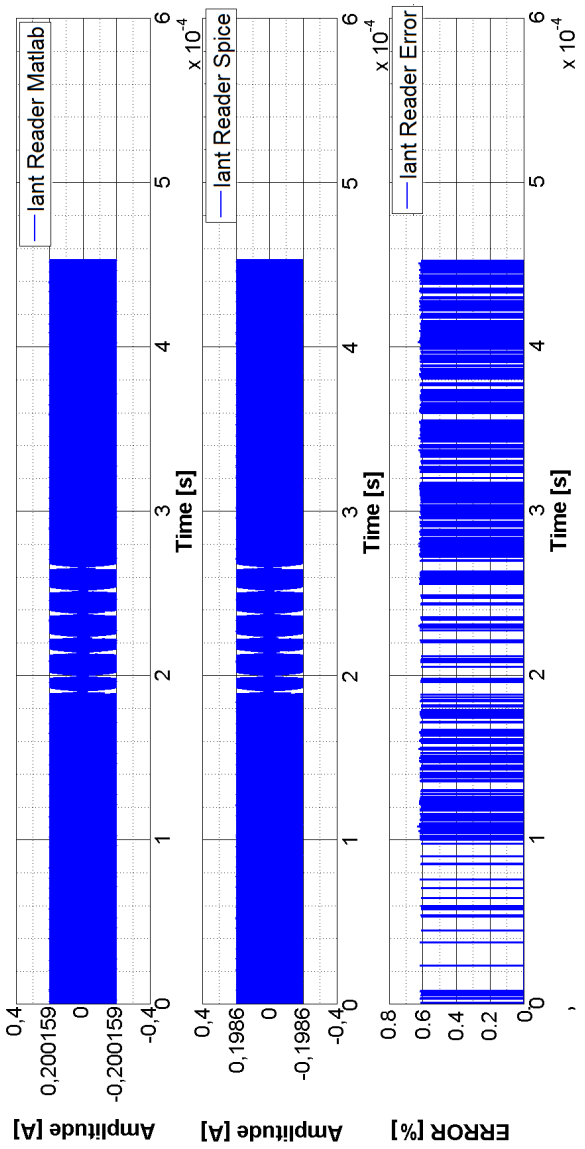

Figure 6.5: Antenna current of PCD

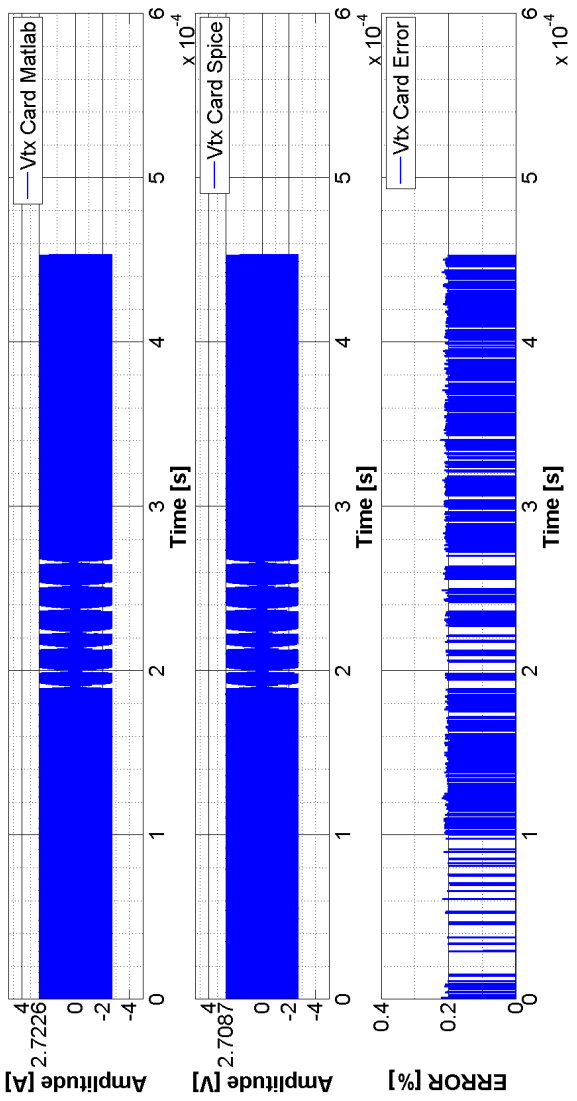

Figure 6.6: RX Voltage PICC

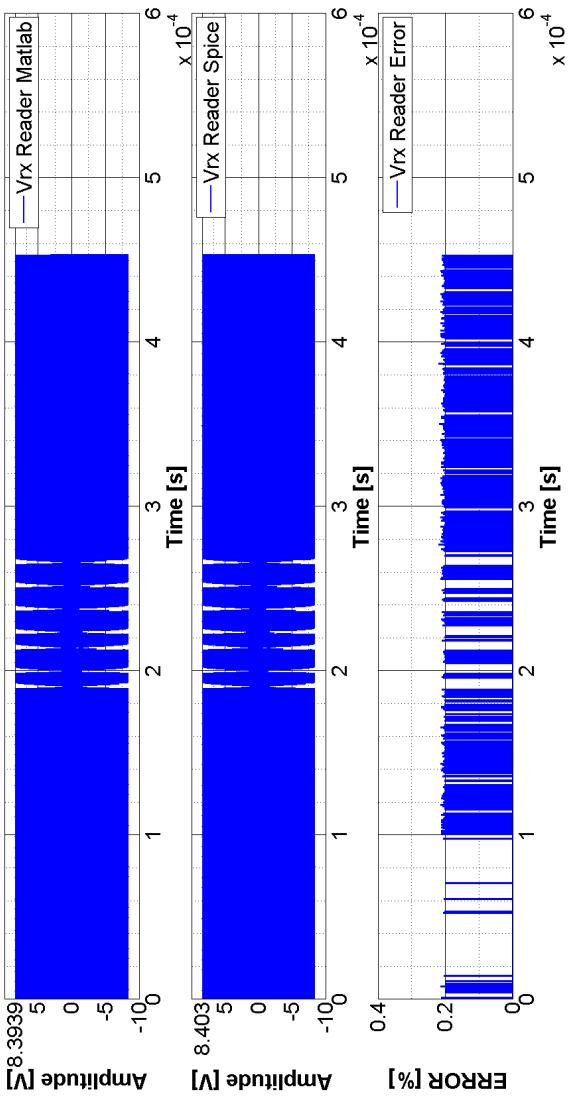

Figure 6.7: TX Voltage PCD

The error between Spice and Matlab is calculated in % of the maximum magnitude of the reference signal (Spice signal). In other words, the magnitude for both signals is calculated, the difference is calculated and normalized to % of maximum signal amplitude of the Spice signal. This way, the error is always defined as a relative error between the Spice model and the Matlab model. Due to the fact that the result vectors in Matlab are equally spaced in time and the result vectors in Spice are not equally spaced in time, an additional interpolation error is interfered.

The results show very low errors ($<$1 %), which is very good given the additional interpolation errors.

6.1.5 Discussion

Time-domain simulations with a shaped envelope yield very similar results for the Spice model and the automated Matlab model. Also the frequency response looks very similar. These results imply that the automated model fits the spice simulation well enough for further use. The simulation setup can be used to verify the ISO compliance tests with the Matlab model. An analog receiver model and a digital receiver model need to be attached to the output of the air-interface model. A successful decoding of the received envelope would then show the ISO-compliance.

6.2 Reader to Card Communication: Coupling Variation

This example shows the influence of the coupling between reader and card. Due to the fact that the coupling changes with the distance of PCD and PICC, the variable coupling k could be seen as a variable distance. To show the influence of the coupling, the circuit behaviour is shown for three different coupling factors. A simulation is shown where the card is 'moved' away from the reader. Finally, the behaviour is compared with the corresponding spice simulation to show that the coupling influences the spice and the Matlab model in the same way.

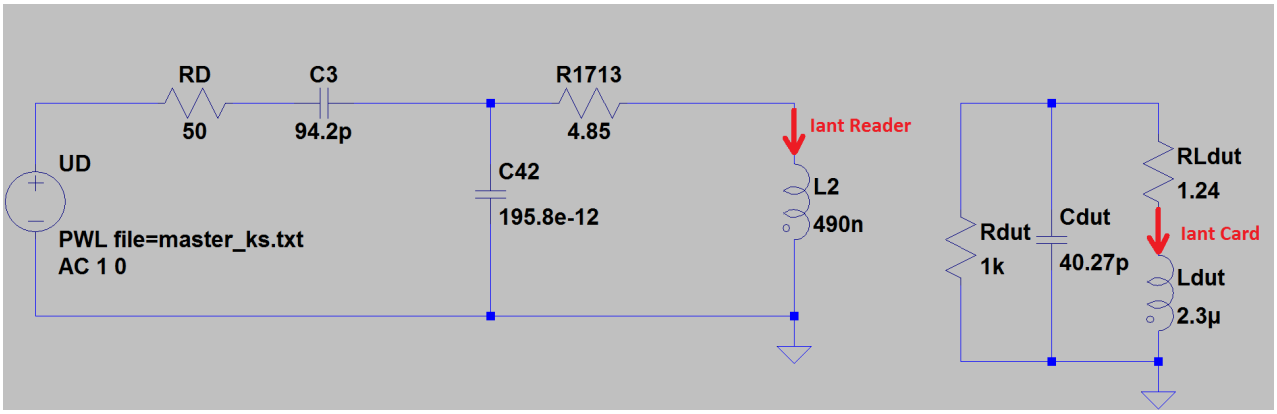

Figure 6.8: Spice schematic of PCD to PICC for variation of k

6.2.1 Circuit

The used matching circuit (see Fig. 6.8) is a 'dummy' PCD with a 'dummy' PICC just to illustrate the behaviour of the coupling. The netlist is directly exported from spice and put into the automated model generation part of the Matlab framework.

6.2.2 Stimulus Generation

The PCD sends a 'Type A' reader envelope (106k bitrate, Modified-Miller encoding, 32x oversampling, dataword: 'AF'). For the sake of comparability, the spice simulator and the Matlab framework use the same stimulus.

6.2.3 Sweep over k

A sweep of the coupling parameter k_{12} for a value range of 0.01 to 0.8 is done in 0.01 steps. Fig. 6.9 shows the result of the sweep.

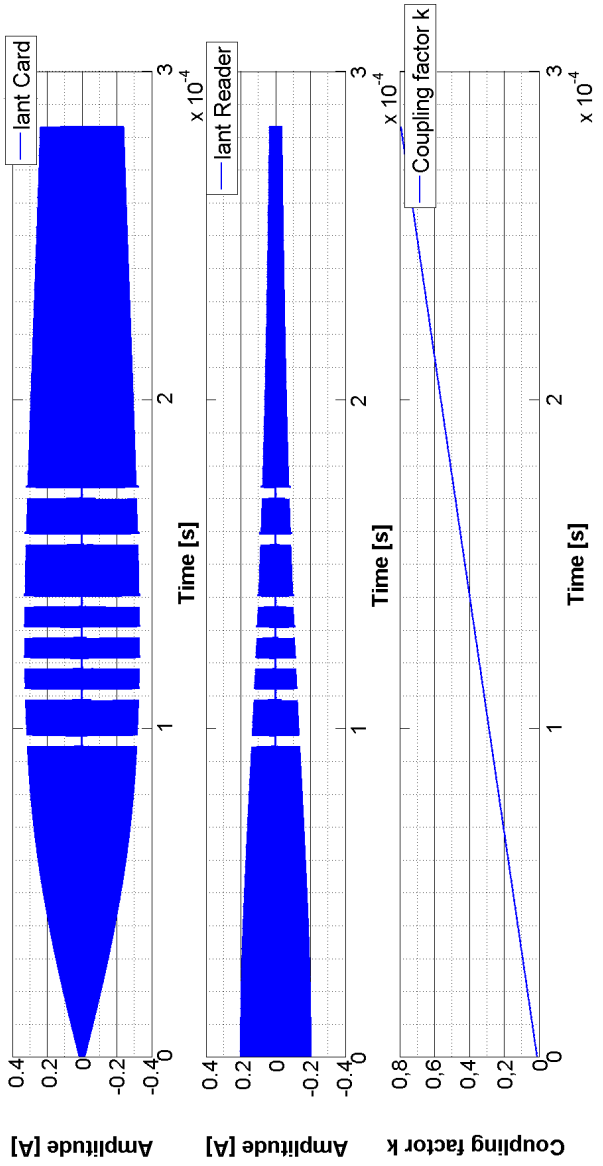

Figure 6.9: Sweep over k = 0.01 to k = 0.8

The result shows that the matching for a certain distance range. When moving the PICC towards the PCD, the coupling gets better and the matching network gets in tune. For a certain range around a coupling of k = 0.2 to k = 0.5 the current in the PICC antenna is at a maximum. For a closer distance where k >0.6 the tuning gets worse again. This comes from the definition of the use case. The distance (and therefore coupling value k) is optimal for the range that is average for most application use cases. The PCD current gets lower for higher coupling factors. This is due to the fact that the PICC is a higher load for the PCD for higher coupling factors.

6.2.4 Discussion

The 'online' variation of the coupling parameter is shown. This represents a model for a card that moves in the reader field. This kind of simulation can be used to determine the voltage and current values at different nodes in the PCD and the PICC to analyze the tuning behaviour over coupling.

6.3 Reader to Card Communication: Sweep over Component Value

A key aim of this framework is to perform tuning optimization tasks. To show how this optimization works, the coupling system from the previous task (PCD with dummy PICC, Fig. 6.8) is used. As stimulus the PCD just sends a plane carrier signal without data.

For a fixed coupling of $k = 0.3$ the voltage at RX of the card is simulated over the variation of the capacitor parallel to RX. The value of the capacitor 'Cdut' is varied from 10pF to 100pF. The tuning of the whole system will be influenced by this parameter value variation.

The point where the voltage at card RX is at a maximum can be seen as the point for the optimal tuning for the coupling of $k = 0.3$. The corresponding value for Cdut for which the RX voltage reached a maximum is the value for the optimal tuning.

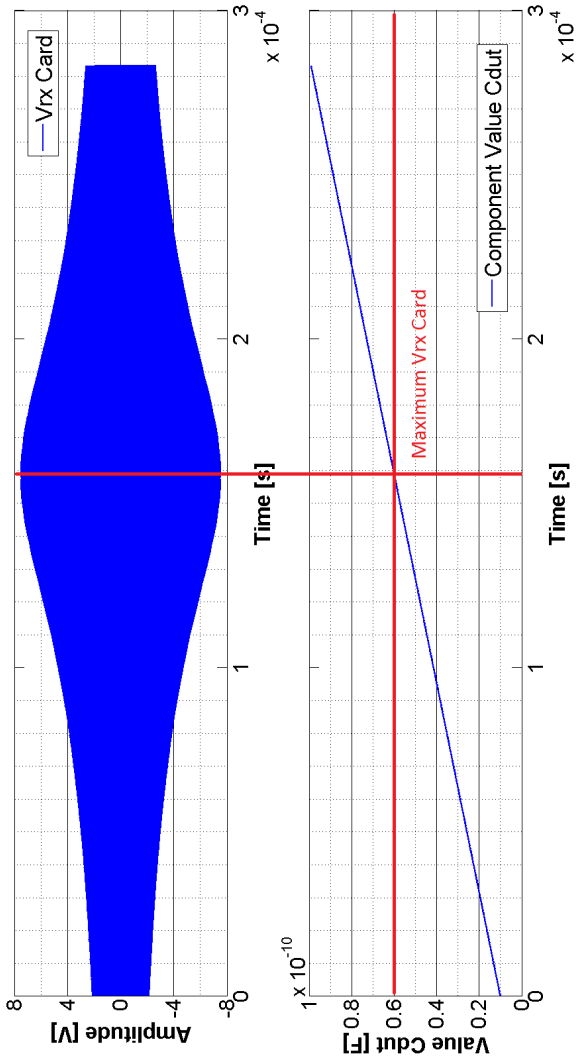

Figure 6.10: Sweep over Cdut = 10pF to 100pF

6.3.1 Discussion

Given a certain circuit topology and coupling parameters, reader receiver/-transmitters of card receiver/transmitter parts can be optimized according to given specifications. A variation of more than one component value at once is not implemented and not necessary. Due to the fact that for most applications one of the communication partners (PCD or PICC) is already given and therefore fixed, no optimisation can be done on this circuit part. A variation of the coupling factors and component values also is not necessary, because the coupling factor represents the geometric coupling situation and is part of the use case for which the optimization is done.

7 Concluding/Summary & Outlook

The outcome of this thesis is a Matlab framework that can model the physical behaviour of an NFC air-interface circuit. The model describes the behaviour at an appropriate level for R&D and V&V. Due to the holistic approach of the framework, the stimulus generation, model generation (and Matlab model vs. spice model comparison), simulation and output evaluation can be done in one tool. The modular construction of the framework provides the option of functional extensions. The framework is fully usable as a stand-alone tool for matching network analysis. It is also the basis for a more sophisticated design and simulation environment for concept engineering for future NFC products. The framework is already in use. Additional features, additional air-interface models, analog models and digital models are under development. The functions for spice model to Matlab model comparison are a big help for verifying used models and therefore integrating the framework in the traditional work flow. Possible extensions include models of the analog receiver and models of the digital receiver. These additional models are connected to the air-interface model within the same framework. In the digital receiver model the data can be decoded and interpreted. This represents a full NFC signal chain from signal generation to signal detection. With every extension to the framework, more and more project stages will include the usage of the framework.

Abbreviations

- RFID ← Radio frequency identification
- NFC ← Near field communication
- ALM ← Active load modulation
- SS ← State-space
- SSM ← State-space model
- KVL ← Kirchoff's voltage law
- KCL ← Kirchoff's current law
- PCD ← Proximity coupling device
- PICC ← Proximity card or
- DUT ← Device under test
- LTI ← Linear time-invariant
- LTV ← Linear time-variant
- DFS ← Depth-first search
- R & D← Research & Development
- V & V ← Verification & Validation

Bibliography

Burfield, Chandler (2013). *Floyd-Warshall Algorithm*. URL: http://math.mit.edu/~rothvoss/18.304.1PM/Presentations/1-Chandler-18.304lecture1.pdf (cit. on p. 60).

Cheever, Erik (2014). *Scam: Symbolic Circuit Analysis in Matlab*. URL: http://www.swarthmore.edu/NatSci/echeeve1/Ref/mna/MNA6.html (cit. on p. 20).

Finkelzeller, Klaus (2008). *RFID Handbuch*. sixth. Hanser. ISBN: 978-3446429925 (cit. on pp. 2, 3).

Franz Aurenhammer, Oswin Aichholzer (1999). *Datensstrukturen und Algorithmen*. TU Graz Skriptum (cit. on p. 60).

ISO/IEC (2008). *Identification cards - Contactless Integrated Circuit Cards - Proximity Cards, 2nd Edition*. URL: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=39693 (cit. on pp. 2, 5).

ISO/IEC (2013). *Information technology - Telecommunications and Information Exchange between Systems - Near Field Communication - Interface and Protocol*. URL: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56692 (cit. on pp. 2, 5).

Muehlmann, Ulrich (2013). *Matlab Modelling of Electrical Networks (Internal NXP Paper)* (cit. on p. 36).

N. Dourdoumas, M. Horn (2003). *Regelungstechnik*. Pearson. ISBN: 978-3-8273-7059-4 (cit. on p. 32).

T. Cormen Ch. Leiserson, R. Rivest (1990). *Introduction to Algorithms*. MIT Press. ISBN: 0-262-03141-8 (cit. on p. 60).

Wing, Omar (2008). *Classical Circuit Theory*. Springer. ISBN: 978-0-387-09740-4 (cit. on p. 19).