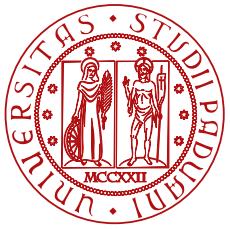


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Laurea Triennale in Ingegneria Elettronica

Application of model-based systems engineering in complex systems development: a power management IC case study

Laureando

Davide Bicego

Matricola 2000131

Relatore

Prof. Claudio Narduzzi

Università di Padova

Correlatore

Ing. Enrico Orietti

Infineon s.r.l

Anno Accademico
2022/2023

Alla mia famiglia per avermi permesso di fare tutto questo, al prof. Narduzzi e ad Enrico per il sostegno e la disponibilità durante questo percorso.

Abstract

This thesis explores the use of Model-Based System Engineering (MBSE) in the design and development of Power Management Integrated Circuits. PMICs are critical components in modern electronic systems, responsible for managing and regulating power supply to various components of a system; the use of MBSE can greatly improve the efficiency and reliability of systems design, by providing a systematic and hierarchical approach to the development and description of the system. The thesis work begins with a study of the principles and advantages of MBSE, followed by a detailed overview of sysML, the model description language chosen for the project, intending to be a preliminary approach to SysML and Enteprise Architect modeling tool for the reader. Proceeding further, the case study is presented, where MBSE is applied to the design of a first set of modules of PMIC, demonstrating the effectiveness of this technique and the quality of the final design, including the set of simulations performed on the model and final results.

Sommario

Questa lavoro di tesi esplora l'utilizzo dell'approccio MBSE, *Model-Based Systems Engineering*, nella progettazione elettronica, in particolare nel design di un PMIC. I PMIC sono componenti fondamentali nei moderni sistemi elettronici, responsabili della gestione e regolazione dell'alimentazione per le diverse componenti di un sistema; questa tecnica ha le potenzialità per notevolmente migliorare l'efficienza e l'affidabilità del sistema, fornendo un approccio sistematico e gerarchico allo sviluppo e alla descrizione del sistema.
Il lavoro di tesi inizia con uno studio dei principi e dei vantaggi del MBSE, seguito da una panoramica dettagliata di SysML, il linguaggio di modeling scelto per il progetto, che mira ad essere un approccio preliminare per il lettore all'utilizzo di SysML e di Enterprise Architect come ambiente di sviluppo. Successivamente viene presentato il case-study, dove l'approccio MBSE è applicato nello sviluppo di un set di moduli per un PMIC, dimostrando l'efficacia di questa tecnica e la qualità del design finale, compreso l'insieme di simulazioni eseguite sul modello e i risultati delle analisi condotte.

Contents

List of Figures

List of Acronyms

MBSE Model Based Systems Engineering

EA Enterprise Architect

UML Unified Modeling Language

SysML System Modeling Language

STM State Machine Diagram

1

Introduction

This thesis is based on my internship experience at Infineon Technologies s.r.l. [1] The focus of this work is the exploration of the model based systems engineering (MBSE) approach in developing a conceptual prototype of a power management integrated circuit (PMIC) and performing a first set of significant analyses. By doing so, this research aims to contribute to the transition from a standard document based system engineering approach to a model based method for the design of complex systems.
The second chapter will provide the driving factor behind the adoption of the MBSE approach in system development, it delves into the challenges faced by traditional document-centric approach, such as inconsistency, ambiguity and difficulty in managing complexity; highlighting the benefits of MBSE.
The third chapter will go through the technical part of the MBSE approach, providing the understanding of the sysML fundamentals, the modeling language employed, and how it can be implemented using Enterprise Architect (EA from now on), a Sparx Systems software [2] designed specifically for modeling purposes.
The fourth chapter will set the focus on the PMIC model design, providing a road-map of the MBSE methodology applied to the PMIC case study, this chapter will go through the creation of the set of sysML diagrams which will constitute the model. Simultaneously the step-by-step process will enable the reader to quickly grasp the modeling tool's functionality and gain proficiency in applying the MBSE approach.
The fifth and sixth chapter will focus on conducting analysis on the developed

[1] https://www.infineon.com/cms/en/
[2] https://sparxsystems.com/

model, integration between EA and external solvers will be investigated, along with all the problematics involved in the process of conducting dynamic analyses on a descriptive model, together with the consequent change of trajectory. Target of this two chapter is to evaluate the system behavior to external stimuli, in order to refine the model and simultaneously mastering the MBSE technique.

1.1 Infineon Technologies

This chapter offers an overview of Infineon Technologies, the framework within which this thesis was developed, its guiding principles, market position and key factors driving its growth.
Infineon Technologies is a world leader in semiconductor solutions and counts about 56200 employees worldwide and ranks in the 10 percent most sustainable companies in the world.
The company has its headquarter in Munich, but as of 30 September 2021 has 19 manufacturing locations and 59 R&D locations spread all over Americas, EMEA and Asia/Pacific region. Business growth in the semiconductor market is driven by the following segments:

Figure 1.1: Driving market areas

Infineon today encompasses four business areas:
Power & Sensor Systems (PSS) drives leading-edge power management, sensing, and data transfer capabilities, main applications are audio amplifiers, BLDC motor, cellular communications infrastructure, charging stations for electric vehicles, human-machine-interaction, Internet of Things, LED and conventional lighting systems, mobile devices and power management.
Connected Secure Systems (CSS) is at the heart of the IoT, core applications are Authentication, automotive, consumer electronics, government identification documents, IoT, Mobile communications, payment systems, ticketing, access

control and trusted computing.
Industrial Power Control (IPC) empowers a world of unlimited green energy, fields of application are energy generation, energy storage, energy transmission, home appliances, industrial drives, industrial power supplies, industrial robotics, industrial vehicles, traction.
Automotive (ATV) shapes the future of mobility with microelectronics, moving towards clean, safe, and smart car. The core applications are: Assistance systems and safety systems, comfort electronics, infotainment, power-train, security.

In the 2021 fiscal year the company reported sales of around €11.06 billion divided on the business areas as seen in Figure 1.2

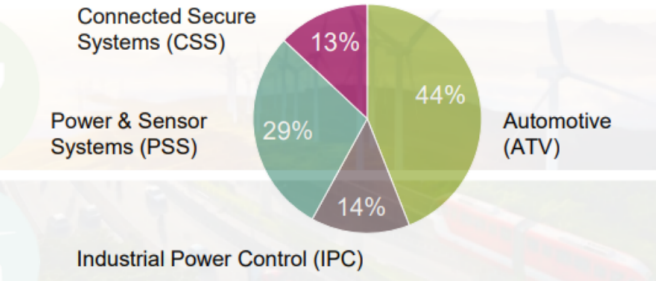


Figure 1.2: Revenue split by segment.

Infineon holds a prominent position in the industry as a leading provider of automotive semiconductors, power discrete modules, Microcontroller suppliers, Security ICs, MEMS die market share, and NOR Flash solutions. The company prides itself on establishing strong and enduring customer relationships built upon comprehensive system knowledge and deep understanding of various applications.
Infineon demonstrates its commitment to technological advancement through substantial investments in research and development (R&D). Approximately 13% of the company's annual revenues are dedicated to R&D activities. In the fiscal year 2021, Infineon invested 1.4 billion euros in this department, enabling continuous expansion of its extensive patent portfolio, which currently comprises nearly 30,000 patents. Such investments strengthen the company's ability to innovate and deliver cutting-edge solutions to meet the evolving needs of the market.
Infineon Italy comprehend a team of over 300 highly skilled professionals who are based in the research centers of Padua, Milan, and Pavia. These experts foster strong and mutually beneficial collaborations with renowned universities, facilitating a continuous knowledge sharing, vital in an industry where technological innovation serves as the cornerstone of achieving business success.

2

Model-Based systems engineering

Model-based systems engineering is a formalized methodology that is used to support the requirements, design, analysis, verification, and validation stages associated with the development of complex systems. In contrast to document-centric engineering, MBSE puts models at the center of system design. The increased adoption of digital-modeling environments during the past few years has led to increased adoption of MBSE. [6] Even though MBSE does not dictate specific procedures, essentially any specific design process should cover four systems-engineering domains:

- Requirements/capabilities
- Behavior
- Structure/architecture
- Verification/validation

Descriptions of these domains are well documented and discussed by, among others, Defense Acquisition University (DAU) [1] and NASA [2]. The difference that MBSE makes is that these fundamental systems-engineering domains are not defined as a set of documents, but by the model itself, i.e., in a formal way using a modeling language. The advantages of this technique will be covered in the following sections.

2.1 Motivation

In a traditional document-centric approach, systems are described and designed through textual documents such as requirements and design specifica-

[1] https://www.dau.edu/
[2] https://tinyurl.com/mrxx4stx

tions, however, this method faces challenges such as inconsistency, and ambiguity. Complex systems are difficult to represent accurately using text alone, making it harder to understand and communicate all their features and intricacies. On the other hand MBSE embraces a visual and graphical approach to system description. Instead of relying solely on text, MBSE employs models, constituted of different kinds of diagrams, to represent system requirements and elements, their relationships, and behaviors. The diagrams (or views) and elements of the model are hierarchically organized in a central model repository (see chapter 3). This allows for clearer communication, improved collaboration, and early issue detection. The unified nature of the models enhances consistency, and traceability, overcoming the limitations of document-centric systems. Traditional documents obviously lack built-in simulation, often requiring manual calculations or separate tools, whereas MBSE tools like Enterprise Architect enables various types of model verification features. In document-based systems, coordination among teams working on different documents can be challenging, leading to errors and delays, instead MBSE, together with modeling tools, provides a shared model environment where teams can work concurrently, enhancing collaboration, reducing errors, and streamlining the development process.

2.2 Benefits of MBSE

The MBSE approach, in combination with an efficient modeling tool, can drastically improve the design and description performances in systems engineering. This section will go through some major advantages of this methodology, comparing it with the standard approach.
MBSE is based on a standardized and visual representation of the system with a collection of different types of diagram, reducing ambiguity and enhancing communication among stakeholders and designer. This clarity and consistency in communication streamline decision-making processes and reduce the potential for misunderstandings that can lead to costly errors or delays.
Through the use of MBSE systems engineer can obtain a better understanding of the system needs and implications allowing to proceed more quickly and efficiently since the way design elements move together is explicitly shown. The model thus becomes an effective prototype, leading to significant gains in productivity and product quality.
In some situations, identifying stakeholder needs and priorities is a major challenge and the resulting ambiguity often delays decision making. MBSE methodologies and tools facilitate the elucidation of these requirements and priori-

ties,so that risks and uncertainties can be more accurately captured and reduced. Issues can be discovered early through the enforced consistency and relationship visibility between model elements; as a result, the early discovery of errors can reduce the cost and duration of the expensive integration and test phase.

One of the key benefits of Model-Based Systems Engineering (MBSE) is its ability to provide robust error checking and design validation capabilities. In traditional document-centric systems engineering approaches, these activities often rely on manual processes and are prone to human error, MBSE leverages the power of models and simulations to automate these processes, resulting in more reliable and efficient error checking and validation.

System models capture the structure, behavior, and requirements of the system in a unified representation, which allows to perform automated consistency checks within the model, so that all elements are properly connected and that there are no conflicting or inconsistent specifications. Early detection of flaws in the development process prevents potential downstream issues and rework. Error detection is also supported through simulations, that provide insight into system performance and enables design optimization before committing to a physical prototype.

Reuse of model elements is another major advantage of the MBSE methodology. Multiple consistent views can be produced from a single repository to communicate and analyze designs. By leveraging existing models, engineers avoid redundant work and utilize designs and especially elements of models that have already been validated. This accelerates the development process and ensures consistency between parts of the model.

Reuse of data items and elements allows the efficient creation and refinement of data dictionaries or Interface Control Documents., as these are refined they can be reused across different programs by creating element libraries, domain specific constructs, and generic conceptual patterns. This has been shown to greatly reduce the time needed to develop similar system models and documentation [2]. The reusing concept is similar to what lies behind electronic CAD tools, that contains libraries of electronic components, differentiated by levels of abstraction, with customizable and specified properties.

Documents still remain the primary means for most stakeholders to examine model contents, as it is unlikely that many non-specialist reviewers will be able to navigate a systems engineering model, therefore, document generation will continue to be important to support information exchange, reviews, and contractual obligations. Here MBSE tools can play an important role, allowing for the development of templates that automatically generate formatted doc-

uments from the repository. Using templates, developers can automatically generate updated documentation based on the current design status, regularly created with very little effort. [1].

3

SysML language and Enterprise Architect modeling

The choice of a modeling language is a major decision in Model-Based Systems Engineering (MBSE) since complex systems require a language that is both clear and unambiguous. Several modeling languages are available, Object Process Methodology (OPM), the Unified Modeling Language (UML), Enhanced Functional Flow Block Diagrams (EFFBD), and the Systems Modeling Language (SysML). These languages consist of both semantics and syntax, with semantics referring to the meaning behind words and symbols, and syntax representing the rules for expressing semantics and their relationships. It should provide formal and unambiguous semantics and syntax to prevent miscommunication and should support the complete characterization of static and time-dependent system characteristics, their hierarchy, and usage. Explicit representation of relationships between elements ensures traceability. Another important aspect is that an effective language is clear, intuitive, and easily teachable and understandable, facilitating efficient communication.

3.1 OMG Systems Modeling Language

Figure 3.1: OMG SysML logo.

The language adopted for this project is SysML, a modeling language that extends the capabilities of UML with a focus on systems engineering, which is the main reason behind this choice.

UML (Unified Modeling Language) is a general-purpose modeling language that is intended to provide a standard way to visualize the design of a system [4]. The creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design and to provide a common notation for visualizing, specifying, constructing, and documenting systems. It was developed at Rational Software in 1994–1995, with further development led by them through 1996.

In 1997, UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2005, UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard [5].

UML encompasses a rich set of graphical notations and diagram types, each serving specific purposes in representing different aspects of a system. This versatility allows UML to be applied to various domains, ranging from software development to business processes and system architectures. **SysML** is an extension of UML that specializes in systems engineering. Derived from UML, it preserves the foundational principles and notation of UML while incorporating additional constructs, diagrams and notations specifically tailored to address the unique challenges of systems engineering. SysML provides enhanced support for capturing system architectures, requirements, behaviors, and interfaces, enabling a more comprehensive representation of complex systems. In fact, unlike other modeling languages that have broader applications, SysML provides a tailored set of concepts, constructs, and diagrams that align with systems engineering practices, providing an efficient modeling solution. To this days, SysML has gained widespread recognition as an industry stan-

dard[1] for systems engineering modeling.

The diagram shown in Fig. 3.2, realized in Enterprise Architect, illustrates the diagram types that SysML defines for modeling system requirements, behavior, structure, and interactions.

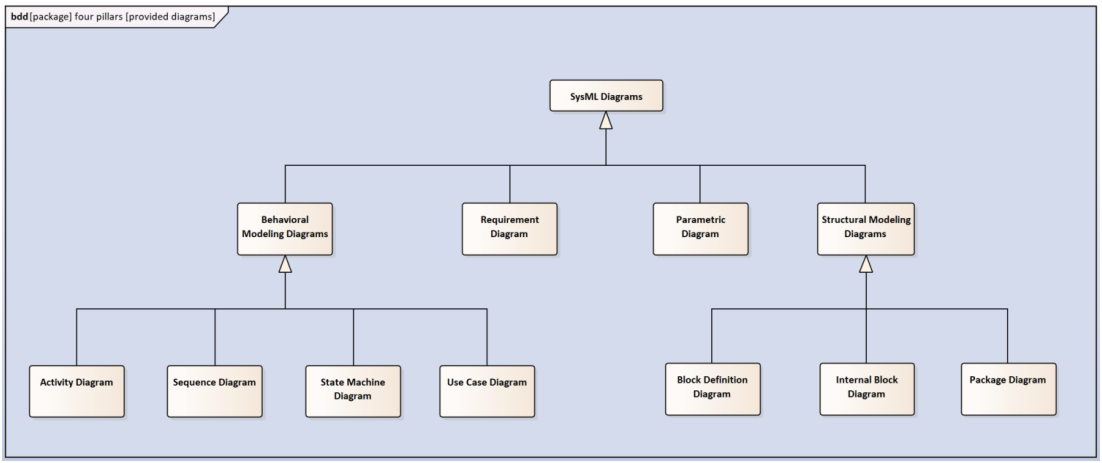


Figure 3.2: Diagram types available in SysML.

The SysML language is designed to ease the description and knowledge capture of the four systems-engineering domains described in chapter 1, to do that SysML embed different kind of diagrams and specific constructs for each of this domains.Considering the extensive variety of diagrams and constructs within SysML, the focus will primarily be on the specific constructs that are relevant to the case study's implementation and analysis.

[1]https://www.omg.org/spec/SysML/1.6/About-SysML/

3.2 Structural modeling

Structural modeling focuses on capturing the static aspects of a system. It involves modeling system components, their relationships, properties and constraints. By defining the system structure, SysML enables a clear understanding of its composition and organization.

A **Block** is the basic unit of structure in SysML. It can be used to model any type of entity in the system or in the environment that surrounds it. In the next chapter the components of the PMIC will all be represented by blocks.

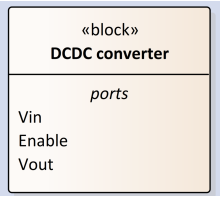


Figure 3.3: Block with parts and ports.

Ports are connection points through which a block interacts with the external world. They represents gateways of information flows, enabling the block to send or receive information to or from other blocks or external entities. It is

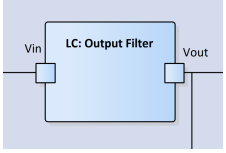


Figure 3.4: An instance of block with two ports.

important to mention that port should always be associated with appropriate **interfaces**, allowing the block to conform to specific sets of operations, signals, or properties defined by the interface.
An **Interface** is a set of specific properties embraced by the port that implements it, this enables blocks to interact with other entities whose ports implement the same interface.

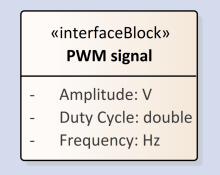


Figure 3.5: Interface block.

For the structural aspect of a system, two main diagrams are employed: the Block Definition Diagram (BDD) and the Internal Block Diagram (IBD).
The **Block Definition Diagram** depicts the blocks or components of a system and their relationships. It provides an overview of system structure by illustrating the major components and their connections.Most importantly it captures the hierarchical relationships among blocks. There are a variety of relationship to better suit the type of connection needed between two blocks, but hierarchy is displayed by the **composition relationship**.
There are no rigid rules for defining Block Definition Diagrams (BDDs), but it is crucial to create an appropriate number of BDDs for the model. Each BDD should effectively portray the functional details of a part or component, ensuring the best representation of its functionality.

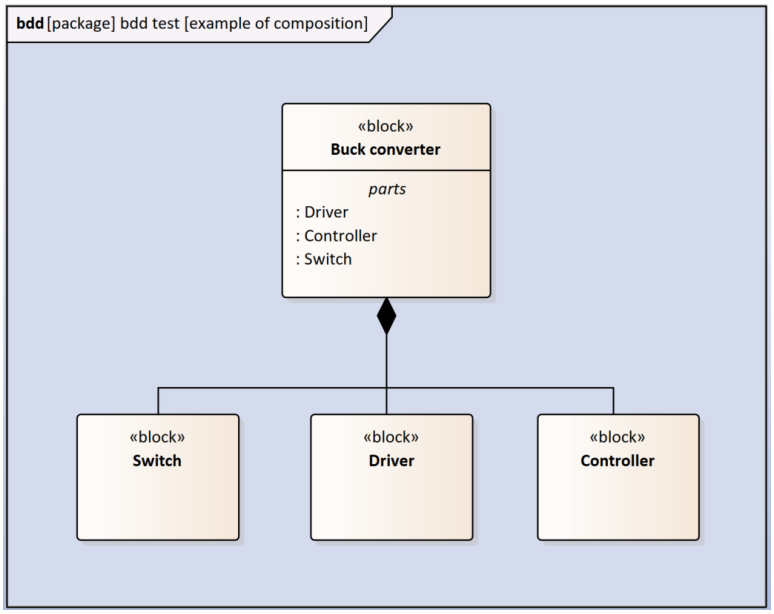


Figure 3.6: Hierarchical BDD structure with composition relationship.

The **Internal Block Diagram** captures the internal structure of a Block element, in terms of its properties (Ports and Parts) and the connections between those properties. It shows the interaction between the block, his components and other blocks. It's worth noting that a block can have multiple Internal Block Diagrams (IBDs). Each IBD can specifically represent and highlight different interactions and relationships of the same block.

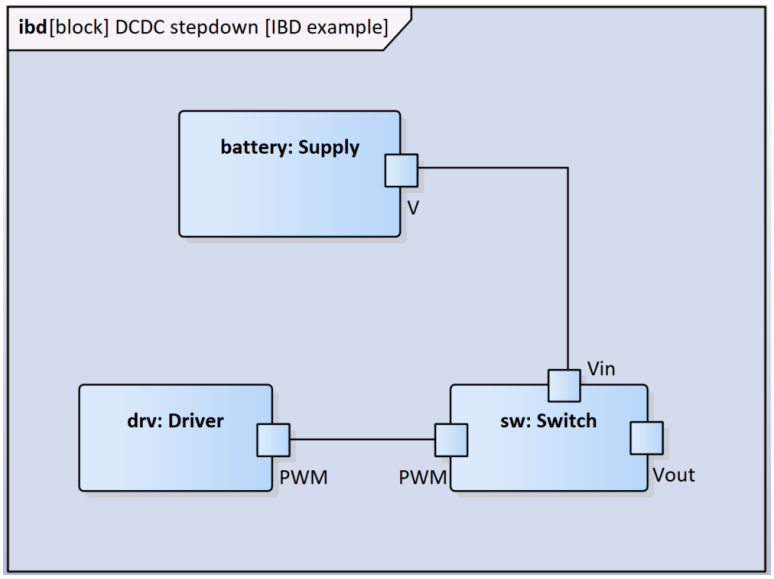


Figure 3.7: IBD of a block.

3.3 Behavioral modeling

The purpose of behavioral modeling in SysML is to capture and represent the dynamic aspects of a system. It focuses on describing system behavior, interactions, and the flow of activities over time. Behavioral modeling allows to analyze, design, and verify the system functionality. Given the variety of modeling possibilities, for the purpose of this work, behavioral modeling can be captured using two types of diagram, a high level use case diagram, that represents the interactions between the system and external users, and, most importantly, a series of state machine diagrams capturing every aspect of the system event-driven behavior.

Actors are the entities interacting with a system, namely users (represented as stick-man), external systems (represented as three-dimensional blocks), or other entities.

Use Case Diagrams are used to depict different high-level use-cases of the system, capturing the interactions between actors and the system under consideration. UC diagrams helps in understanding system functional requirements

and the various ways in which users and external entities interact with it. It is worth noting there are no restrictions on how many use case diagrams should be created, and in the same way no specific rules on the detail level of every single use case. The UC diagram can be hierarchical as well, to meet the desired level of detail.

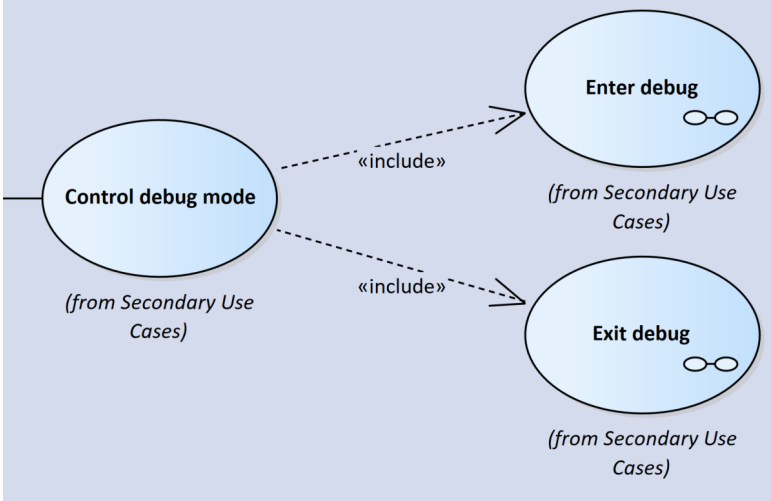


Figure 3.8: Snapshot of a Use Case Diagram.

The **State Machine Diagram** is designed to capture and represent the dynamic behavior and state transitions of a system or component. State Machine Diagrams provide a visual representation of the system behavioral states, events, and state transitions, illustrating how the system or component responds to stimuli and changes its internal state over time. For embedded systems, it is often advantageous to describe behavior in terms of operating states, triggering events and system actions. State machines diagram relate operating states of a system (or block) to triggering events [3]. The key constructs required to create an effective State Machine Diagram are essentially the same used to develop a standard state diagram in engineering fields, namely:
The **Initial State**, indicates the starting point of the state machine diagram. It represents the initial condition or state of the system or component before any event or action occurs.

States represent specific conditions or modes in which the system or component can be. They depict the different operational states of the system, such as *idle*, *active*, *error* or any other relevant state.

Events are the stimuli or occurrences that initiate state transitions. Events can be internal events (generated within the system) or external events (received from external sources).

A **Transition** defines the change of state in response to specific events or conditions. Transitions depict the triggers and actions that cause the system to transition from one state to another. E.g, an enable *event* could trigger a *transition* from the *idle state* to the *active state*.

Actions represent the activities or operations performed during state transitions. Actions specify the behavior or tasks that occur when the system moves from one state to another, e.g, monitoring an output signal when toggling to another load.

Guards specify the constraints or requirements that must be satisfied for a transition to take place. They are conditions associated with transitions that determine if a particular transition should occur.

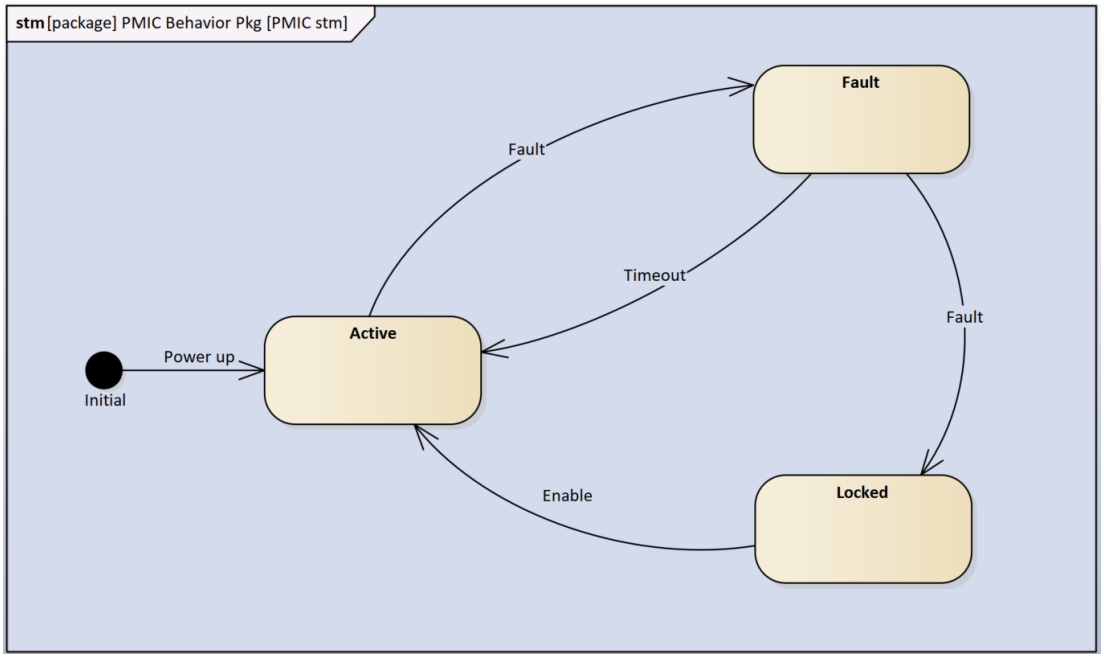


Figure 3.9: Example of a high level state diagram. The label on the transition represents the trigger.

3.4 Requirement modeling

Requirements are statements that define the desired characteristics, functionalities, and constraints of the system. With sysML they can be effectively captured and managed using requirements blocks and requirement diagrams, in order to provide a clear representation fulfilling the needs of ensuring a comprehensive understanding of the characteristic expected from the system.

Requirement Blocks allows to define requirements as specialized blocks, encapsulating the attributes, relationships, and behaviors associated with each requirement.

Requirement diagrams enable the visual representation of requirements and their relationships. These diagrams allows to depict the hierarchy, dependencies, and traceability among requirements, using the *satisfy* and *connector* relationship to illustrate how requirements are related to each other and to other system elements.

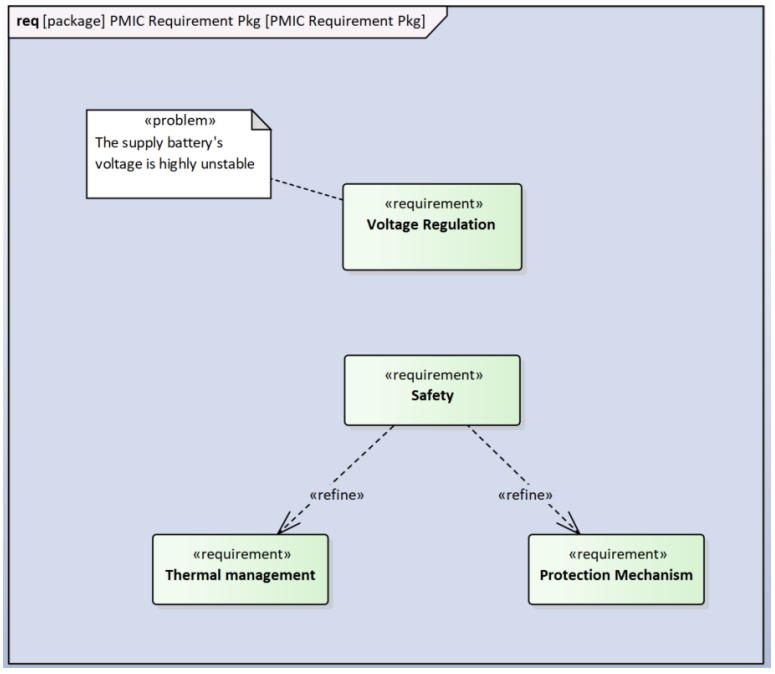


Figure 3.10: Partial high level requirement diagram.

3.5 Enterprise Architect

Enterprise Architect is the tool employed to effectively model the case study of this work.
EA is based on the Unified Modeling Language (UML), but the software can be set in a system modeling perspective that includes all the **sysML** diagram structures and stereotypes, providing a complete environment for creating and managing complex systems.

The modeling workspace is presented as in Fig. 3.11:

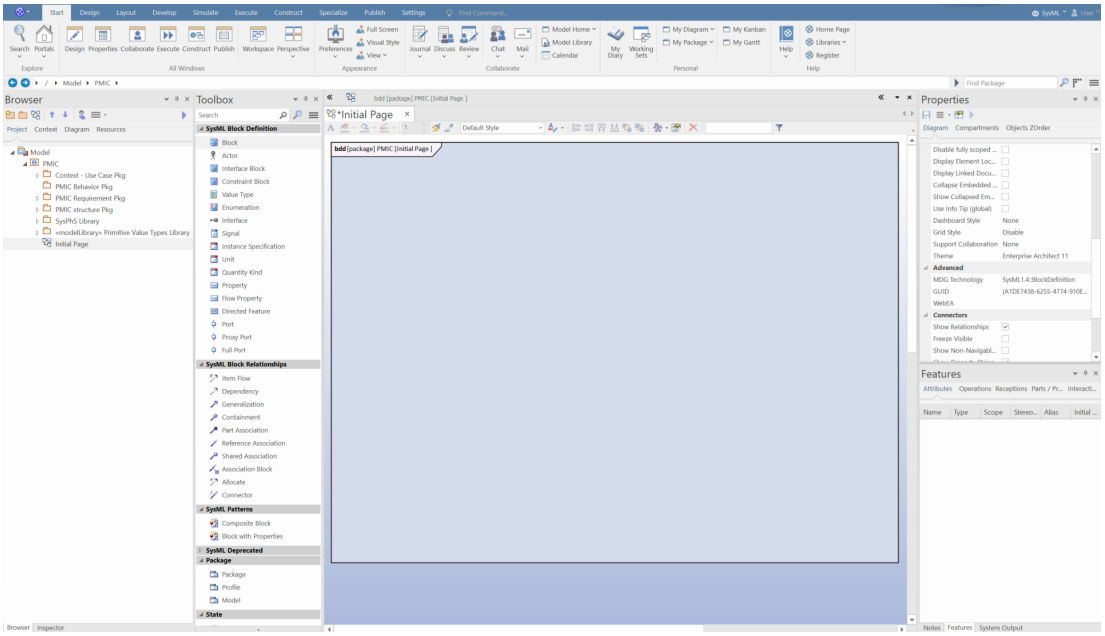


Figure 3.11: Enterprise Architect starting page.

The first of the two workspace key components is the project browser, it serves as a hierarchical representation of the model structure. It presents a tree-like view of the project's packages, diagrams and elements and allows to navigate through the model and access elements and diagrams within the project. It provides essential functionalities such as search capabilities, filtering options, and customizable views, enabling users to locate specific elements or focus on relevant parts of the model. Packages and blocks in the Project Browser can be expanded or collapsed to reveal or hide contents, facilitating a clear and structured view of the model, promoting easy traceability and maintenance of a coherent representation of the system. The other workspace element is the diagramming area, occupying the majority of the main page, provides a canvas for creating and editing diagrams within the system model. with the adjacent Toolbox, all the set of sysML diagrams, blocks and connections are available through drag-and-drop operations, allowing to visually represent system ele-

ments and their relationships in a clear and customizable way.

3.5.1 Model organization

Model organization is the key to **create a navigable and maintainable model;** it can be achieved by exploiting packages in an effective manner. Packages serve as containers for grouping related model elements, providing a logical structure and facilitating the management of the system's complexity.
Dividing the model views into packages and sub-packages allows for a clear, hierarchical and focused representation of system. Each package should be dedicated to a specific viewpoint or domain, such as structural, behavioral, requirements as the main packages, which further specialize in subsystem packages containing every concept and element that has to be modeled. This enables to focus on specific aspects without being overwhelmed by unnecessary details.

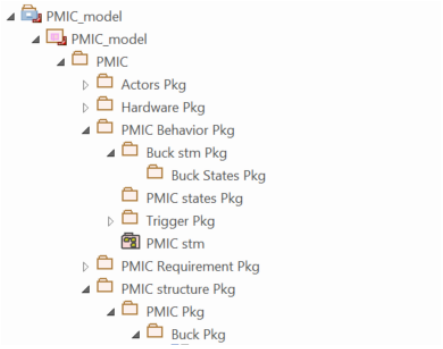


Figure 3.12: Part of the model browser of a package organized model.

A hierarchical package subdivision allows for scalability and flexibility in development.
As complexity increases, packages can be nested to represent subsystems, modules, or functional areas, enabling a flexible and adaptable structure that can evolve with the system while maintaining the consistency of the whole model.

4

Model realization

During the initial stage of the study, analyses were conducted to evaluate and identify relevant modules of the PMIC to start the modeling process.

In order to facilitate a thorough understanding of the modeling process, it was deemed essential to begin with a focused approach that involved a limited number of modules. This approach allowed for a detailed exploration of the selected modules, but most importantly it **enabled to gain proficiency in the use of the modeling tool and the modeling techniques**, without over-committing in complexity.

It was determined that particular emphasis should be placed on three specific modules: the digital control section, the step-down DC-DC converter (Buck), and the voltage monitor module.

The decision to prioritize these modules was based on several factors, **digital control** plays a crucial role in governing the overall operation, modeling this module means it would be possible to analyze and optimize the control algorithms and other digital functionalities, furthermore it allowed to explore the state machine modeling and to gain knowledge about finite state machines in digital electronics.
The **step-down DC-DC converter** module is the core of the IC, responsible for the voltage regulation. By modeling this module, it is possible to assess its working dynamics and performances and how the interactions with the rest of the system works. This understanding can be valuable in refining the design and enhancing the converter capabilities.
The **voltage monitor** module was chosen as it provides critical monitoring and protection functions within the PMIC. By including this module in the modeling process, it allows for the examination and knowledge development of fault

detection mechanisms, and the interaction between the monitoring circuitry and other modules.

By focusing on these selected modules, the modeling effort can be started with a clear direction, leading to valuable insights and improvements in the overall modeling knowledge and in the concept development of the case study.

4.1 Context and use cases

Prior to structural modeling, it is good practice to establish an overview of the entities that our system needs to interact with and, most importantly, the use cases it has to fulfill. This is accomplished by constructing a context diagram, which depicts the system's external entities and their connections, and a high-level use case diagram, which outlines the system's major functionalities and interactions with actors.
Especially in the early stages of new concept development, it is often not immediately clear how the system should be designed. In such cases, these initial steps help in refining the subsequent diagrams and models by providing a clearer understanding of the system desired functionalities and interactions.
The main context diagram is shown in Fig. 4.1:

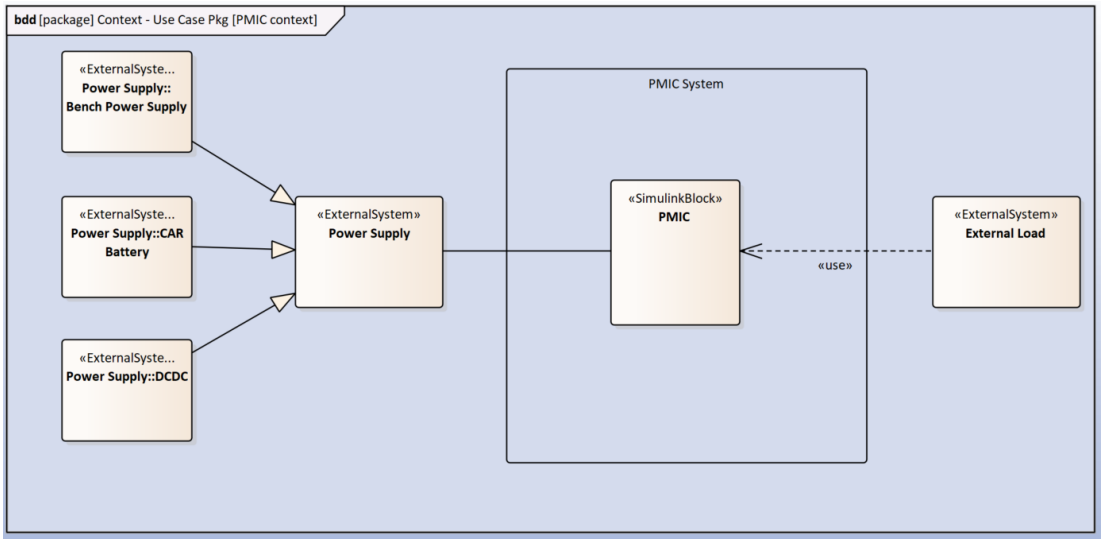


Figure 4.1: Model context diagram.

One of the use case diagram is shown in Fig. 4.2, it is a high level diagram, and it is further specialized into a set of more specific diagrams for every operating state of the system.

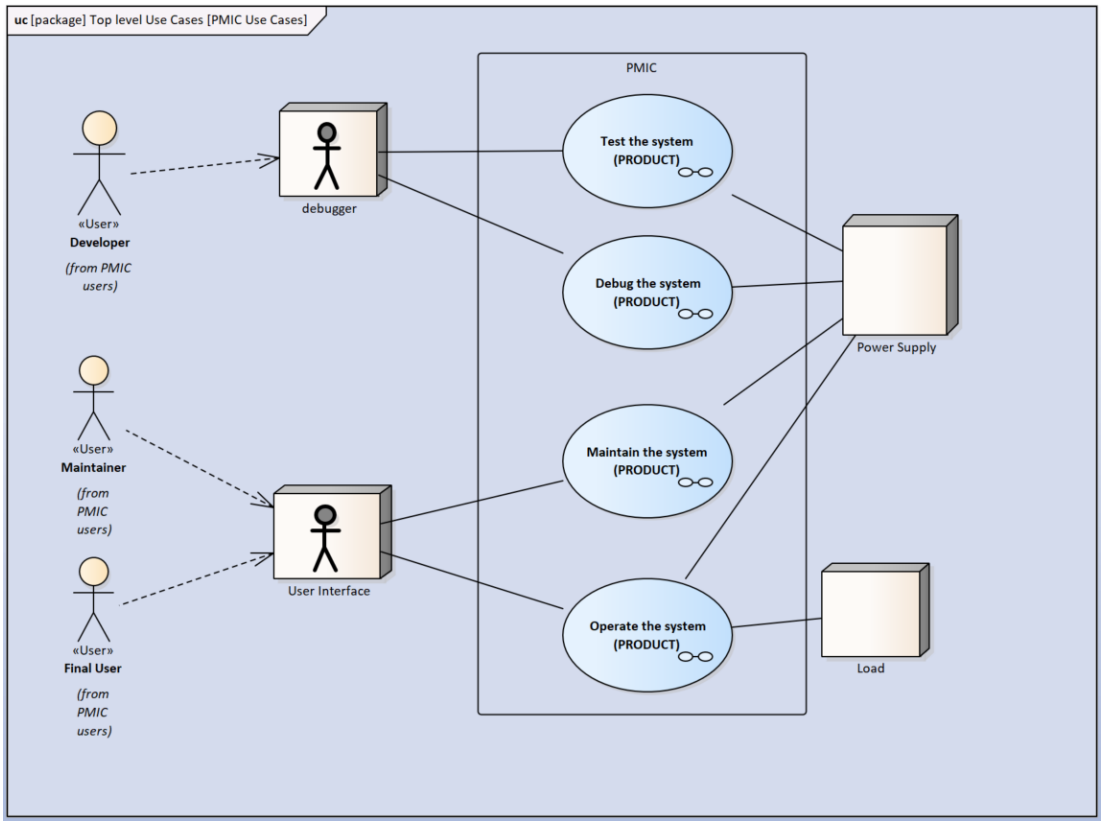


Figure 4.2: High level Use Case diagram.

4.2 Structure

Structural modeling constitutes the core aspect of the model, as it captures the system's composition and organization **at the desired level of abstraction**. This section of the model showcases the system's architecture, the hierarchical arrangement of components, their properties, and the connections between them.
Given the objective of effectively modeling complex systems, it was recognized as essential to adopt a high-level approach rather than delving into every minute physical detail of the prototype. This approach enables a comprehensive understanding of the system's structure while maintaining a conceptual level of abstraction. By focusing on the overall architecture, relationships, and essential components, the emphasis is placed on capturing the fundamental elements and their connections. This approach facilitates analysis, design, and communication by providing a manageable representation of the system, more importantly, **it allows fast scalability of the project**.
The buck converter module is responsible for the voltage regulation in the PMIC circuits, from the supply voltage it produces an output at the desired voltage level. It can be modeled as a block with three components:

- **Control logic:** it provides dynamic duty cycle control for the power stage switches, in order to maintain the output voltage at the desired voltage.
- **Power stage:** this part of the buck converter is controlled by the control logic, it is usually constituted by two transistors that provide the pulsed output voltage wave with the desired duty cycle.
- **Output filter:** it provides a smooth final output voltage from the pulsed output provided by the switching power stage.

These three entities will be the final specialization of this block, meaning they will represent the leaves of the hierarchy tree structure.

Ports and connections are defined in the Fig. 4.3 internal block diagram:

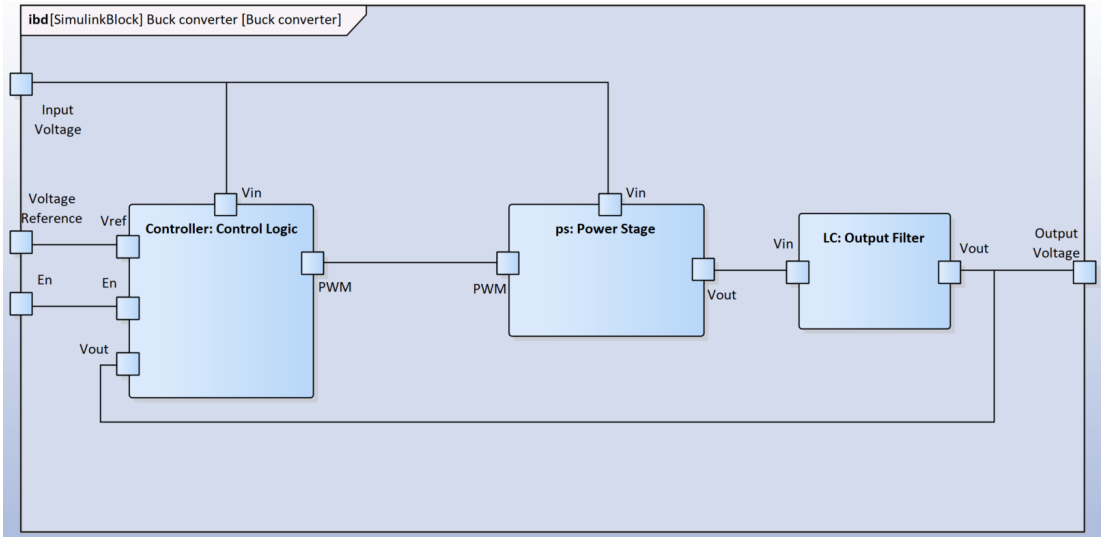


Figure 4.3: Buck Internal Block Diagram.

In the BDD the hierarchical structure of the buck is created, combining the different hierarchy level elements with the composition relationship, the results is reported in Fig. 4.4.

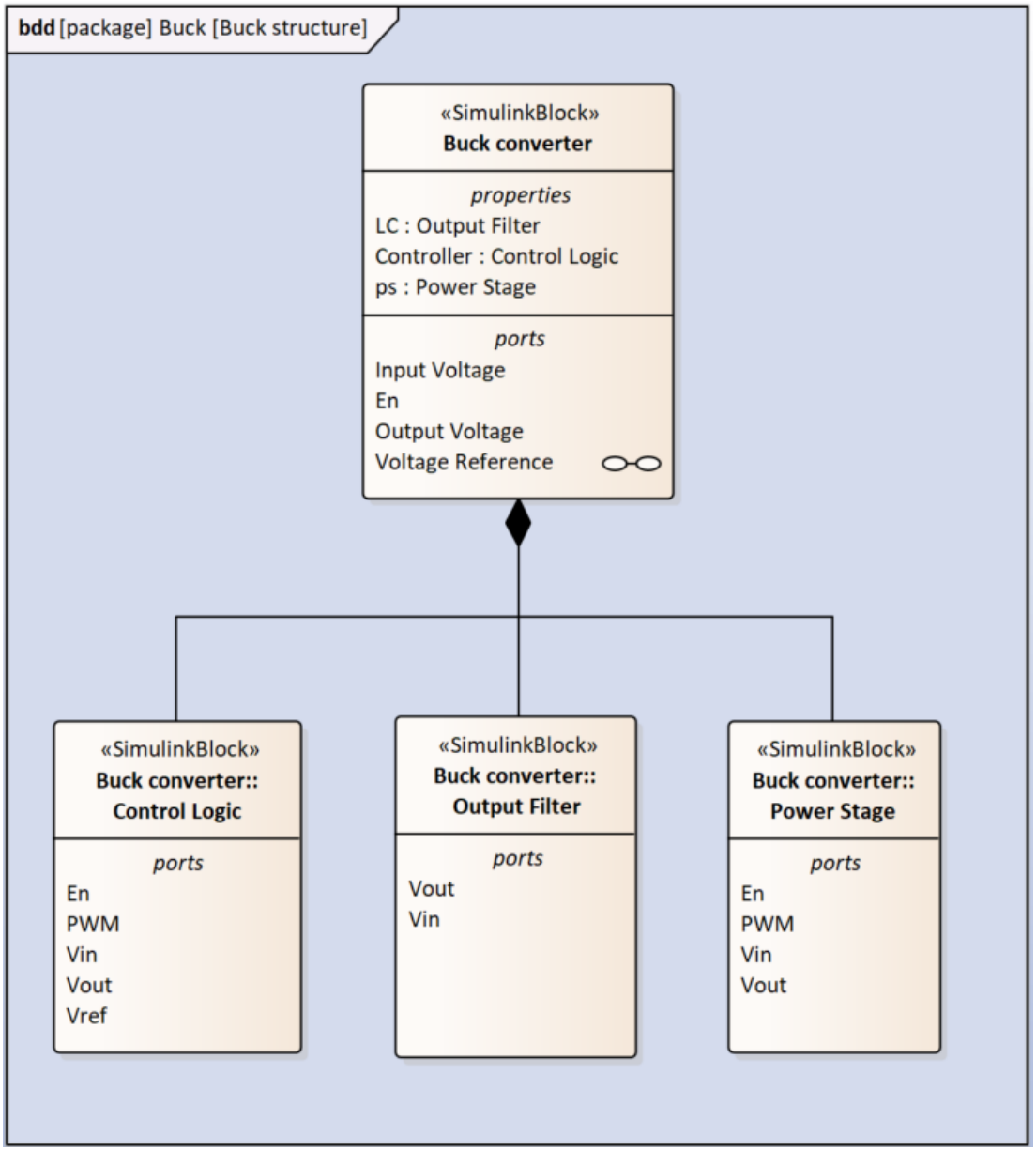


Figure 4.4: Buck block definition diagram.

Due to space constraints and **prioritization of the simulation aspects**, the development of the voltage monitor and digital modules will not be further pursued in this thesis; these two block will represent the final specialization.

Going on to a higher level in the hierarchy, the top-level structure BDD is created as in Fig. 4.5, representing the PMIC components with the usual hierarchical fashion:

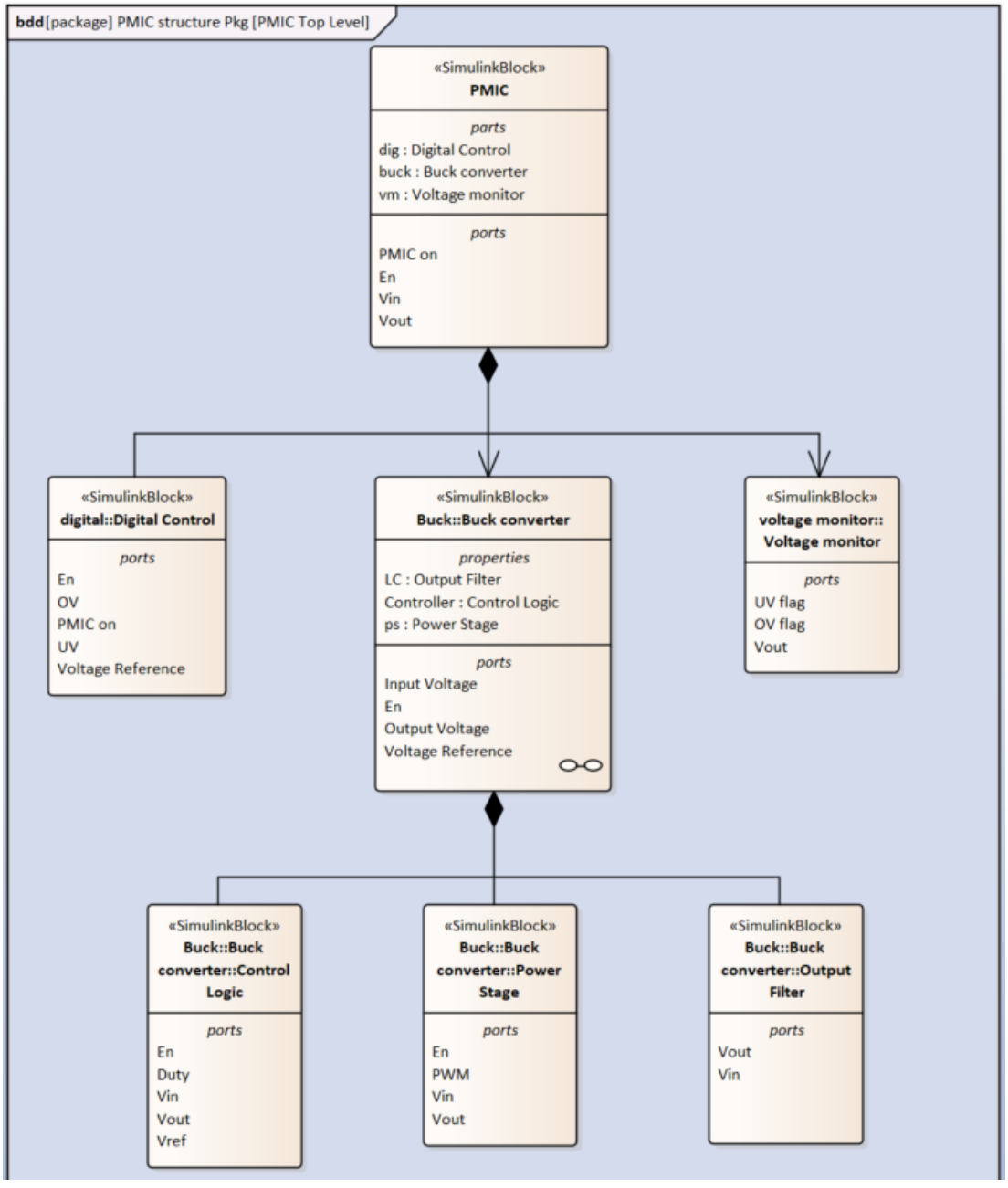


Figure 4.5: PMIC block definition diagram.

The Internal Block Diagram is key in understanding the operational dynamics of the system, facilitating subsequent model enhancement and integration. Upon defining all the necessary ports, the outcome is as follows in Fig. 4.6:

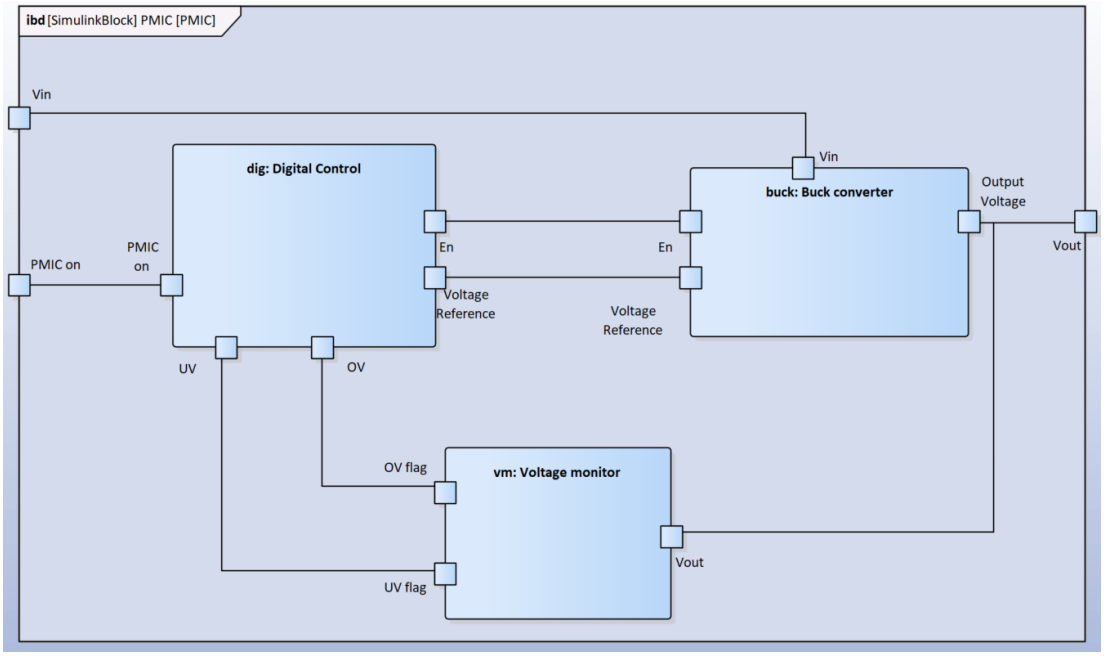


Figure 4.6: PMIC internal block diagram.

These are the main structural representations created for the project.
It is noteworthy to emphasize the adaptability of EA in diagram management. As the project gains momentum and progresses, additional types of diagrams and different BDD's can be defined within various packages. For instance, a variety of IBD's in the PMIC block can be utilized to represent only the voltage supply connections or only the communication bus, other BDD's can be employed to define the communication protocols employed within the system and so on.

4.3 Behavior

This section primarily centers on the digital module, which best aligns with a state machine representation in terms of the various states the PMIC can assume.
The primary state machine diagram presented below depicts the principal states that the PMIC can assume during its operation. This state machine diagram adheres to the formal notation and conventions of state machine modeling in sysML, where each state within the diagram is composed by three sections:

- **Entry action:** optional behavior associated with a state. It specifies the actions that are executed when a transition occurs into the state. It is typically used to initialize the state or set up necessary resources.
- **Do action:** represents the behavior that is continuously executed while the system remains in the given state. It defines the actions or operations that are performed during the state's active period.
- **Exit action:** defines the actions that are executed when a transition occurs out of the state. It is used to perform any necessary actions before leaving the state

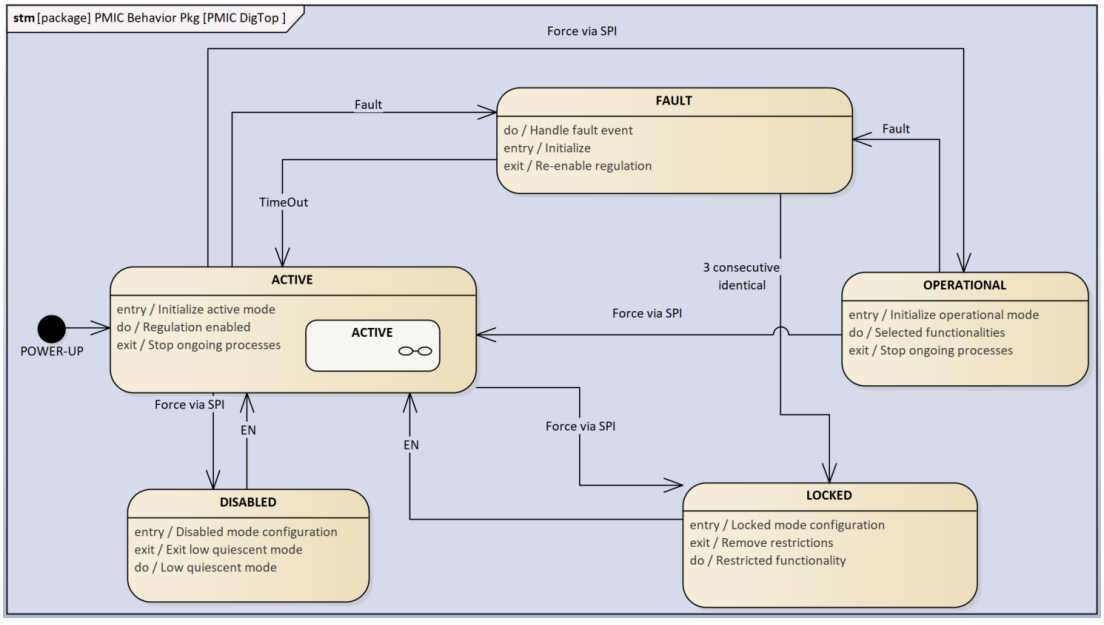


Figure 4.7: PMIC main state machine diagram.

Notably, active state is further specialized using a sub-state machine, which depicts the interactions and transitions among the sub-states of the active state. This subdivision allows for a more detailed representation of the different operational modes.
While not explicitly studied in this thesis, it is worth noting that Enterprise Architect provides the capability of efficient **HDL code generation**, specifically

for VHDL, SystemVerilog, and SystemC, if the state machine diagram adheres to a certain set of rules. This feature enables transition from the graphical representation of the state machine diagram to the corresponding HDL constructs, enhancing the effectiveness of the code generation process. The specific rules and guidelines for ensuring successful code generation should be consulted within the Enterprise Architect documentation or user guide. [1]

4.4 Requirement

The requirement section, although it may not appear useful for relatively simple models, becomes essential when dealing with more complex systems and utilizing the model for effective design purposes. It serves the purpose of ensuring consistency throughout the development process and acts as a guiding framework to ensure that the ultimate objectives of the system are successfully achieved. By explicitly defining and documenting the requirements, stakeholders can align their expectations and verify that the model faithfully represents the intended functionality and performance of the system. As the complexity of the model increases, the requirement section becomes increasingly valuable in managing system specifications, validating design decisions, and facilitating effective communication among project participants.
Two of the requirement diagrams are reported, created to explore the capabilities of the tool, drawing inspiration from real IC requirements. These diagrams represent the overall functional requirements of the PMIC, including the desired tolerances for the output voltage, as well as the electrical characteristics of the voltage regulator, highlighting the acceptable input and output voltage ranges, along with the current consumption specifications.
These visual representations serve as instrumental tools in system design. Unlike traditional tabular formats found in documentation, the visual representations provided by EA enhance navigability and facilitate ease of understanding, providing a holistic view of the requirements.
EA offers a diverse range of features specific for requirement tracking and validation. These features provide a systematic and rigorous approach to managing and monitoring life cycle of requirements, ensuring their completeness, consistency, and adherence to project objectives. With traceability capabilities, stakeholders can easily track the origin and evolution of requirements, facilitating impact analysis and decision-making.
The tool's requirement validation features empower users to verify the accuracy

[1] https://tinyurl.com/statemachineHDLgeneration

and feasibility of requirements through rule-based checks, cross-referencing, and other validation techniques. This systematic validation process helps identify any gaps, conflicts, or inconsistencies within the requirements, enabling timely rectification and alignment with project goals, and allowing automatic generation of requirements tables.

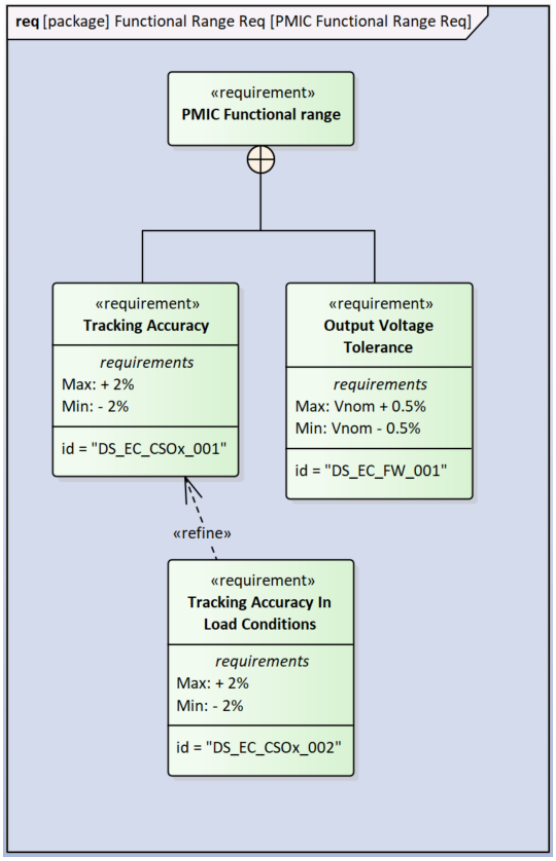


Figure 4.8: PMIC functional requirements.

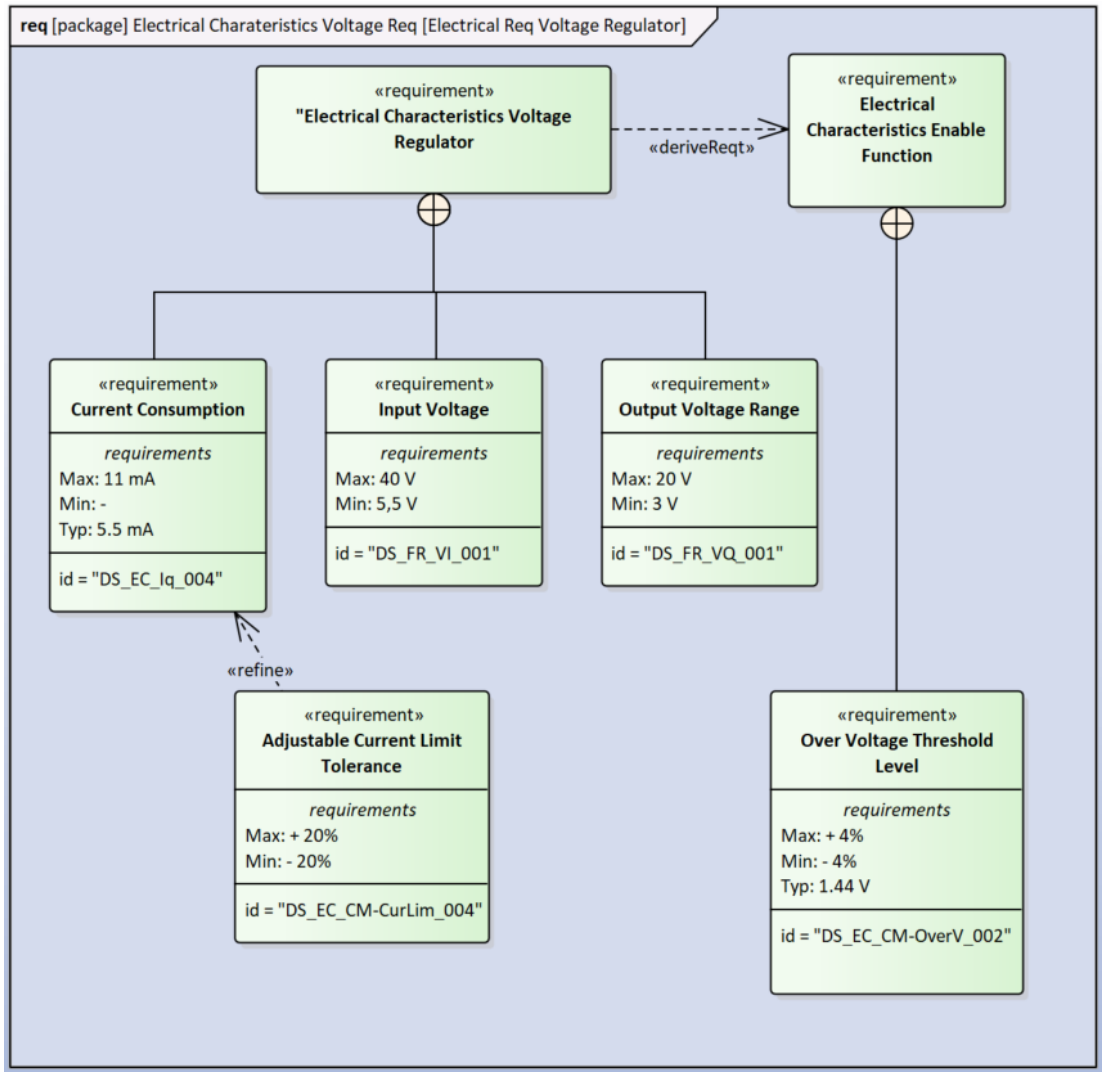


Figure 4.9: Voltage regulator electrical characteristics.

5

EA-Simulink integration

One of the objectives of this research is to identify a feasible approach for conducting model simulations, aiming to evaluate the performance and verify the overall functionality of the system. Initially, an attempt was made to implement a parametric simulation using parametric diagrams, unfortunately, it was soon realized that this level of modeling was too low and exceedingly complex. Parametric modeling consists in creating diagrams, constraints blocks and defining the physics equations and associated variables that govern the behavior of the actual model. A few of the difficulties involved were the limited support for parametric Modeling: Enterprise Architect primarily focuses on static and behavioral modeling aspects and while it does provide some support for parametric modeling through its Parametric Diagrams, the capabilities in this area are relatively limited compared to specialized tools specifically designed for this simulation technique.
Parametric modeling involves defining and manipulating mathematical relationships and constraints between model elements and this can be complex, especially for users who are not familiar with parametric modeling concepts and techniques.
Another problem, which was not an imminent problem, but for sure it would have been once the model increased in dimensions, is the lack of solver functionality, in fact EA does not have built-in solver functionality for solving complex mathematical equations and optimization problems. This limitation hinder the ability to perform parametric simulations and analysis within the tool itself, and an external solver is needed.

5.1 SysPhS simulations

The SysPhS standard [1]of the Object Management Group (OMG), is used to integrate the EA model with the Simulink model. With the SysPhS Simulink pattern, EA provides a wide range of Simulink components as blocks, making it easy to model physical and electrical systems, and allows to model translation to either of two simulation platforms, Modelica and MATLAB's Simulink/Simscape.
Multiple test cases were explored in an attempt to replicate and comprehend the simulation capabilities. The primary objective was to ascertain the feasibility of constructing a model using the Simulink pattern within EA and subsequently generating a corresponding Simulink model for simulation purposes.
A notable result is the following realization of a DCDC converter by means of this method.
The modeling process is the same as the previously seen model realization, with the difference that the block used are the physical components of the system itself. Elements are instantiated using the Simulink pattern and subsequently composed into the buck block, then the IBD is created to establish the connections among the components.

[1]https://tinyurl.com/sysphs

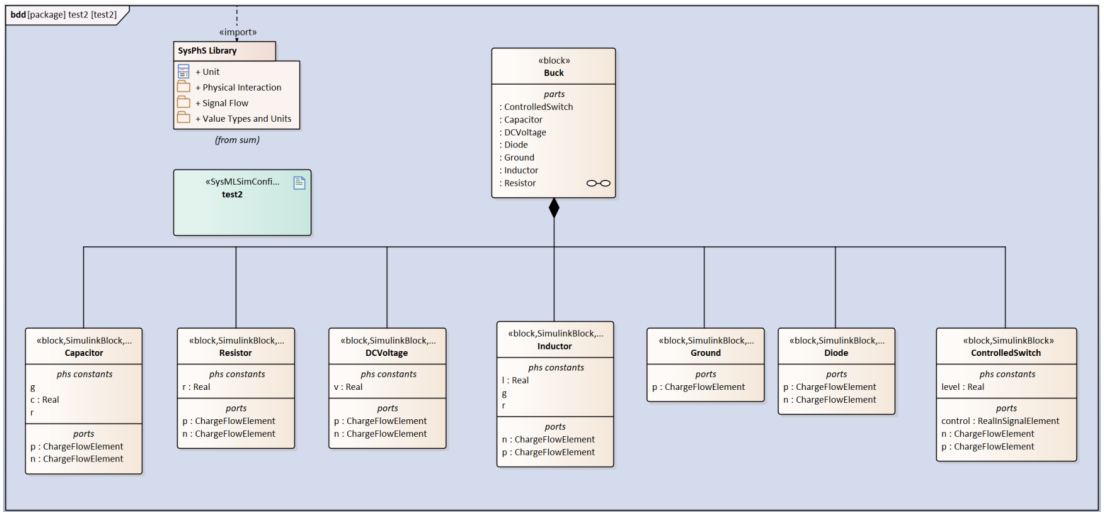


Figure 5.1: SysPhS buck converter BDD.

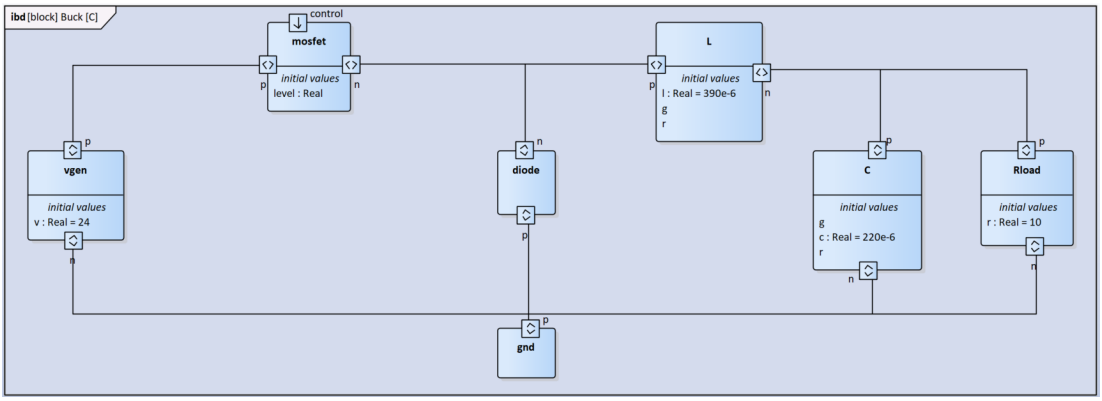


Figure 5.2: SysPhS buck converter IBD.

The simulation configuration is possible by mean of the EA sysMLsim configuration artifact, which allows to select the model to generate and the properties to plot.

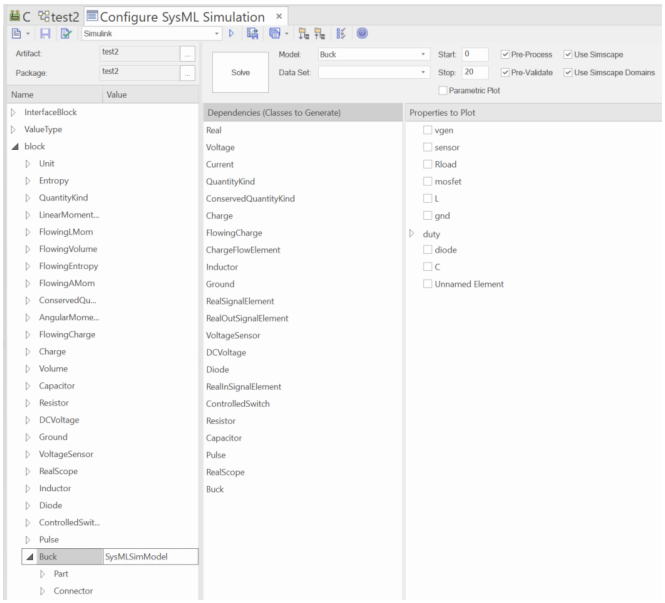


Figure 5.3: sysMLsim configuration artifact.

Due to encountering a series of technical issues, the functionality for plotting properties was found to be non-functional. Interestingly, these issues were not limited to this example but extended to the models provided by Sparx Systems as well.
However, with the code generation feature it is possible to generate a Simulink (.slx) file and its corresponding MATLAB code from the SysMLsim artifact.

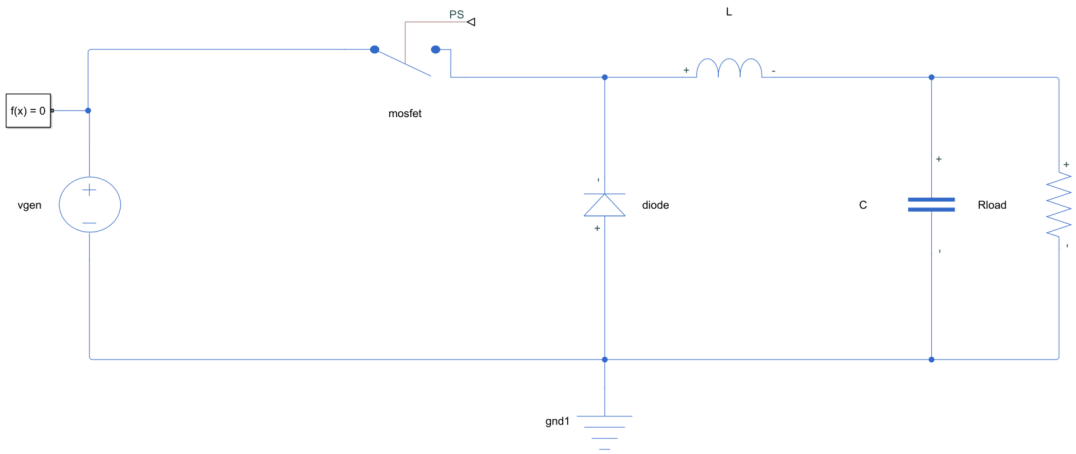


Figure 5.4: EA generated Simulink model.

Achieving a correct net-list representation in the form of a Simulink model, derived from the Enterprise Architect model is a significant accomplishment,

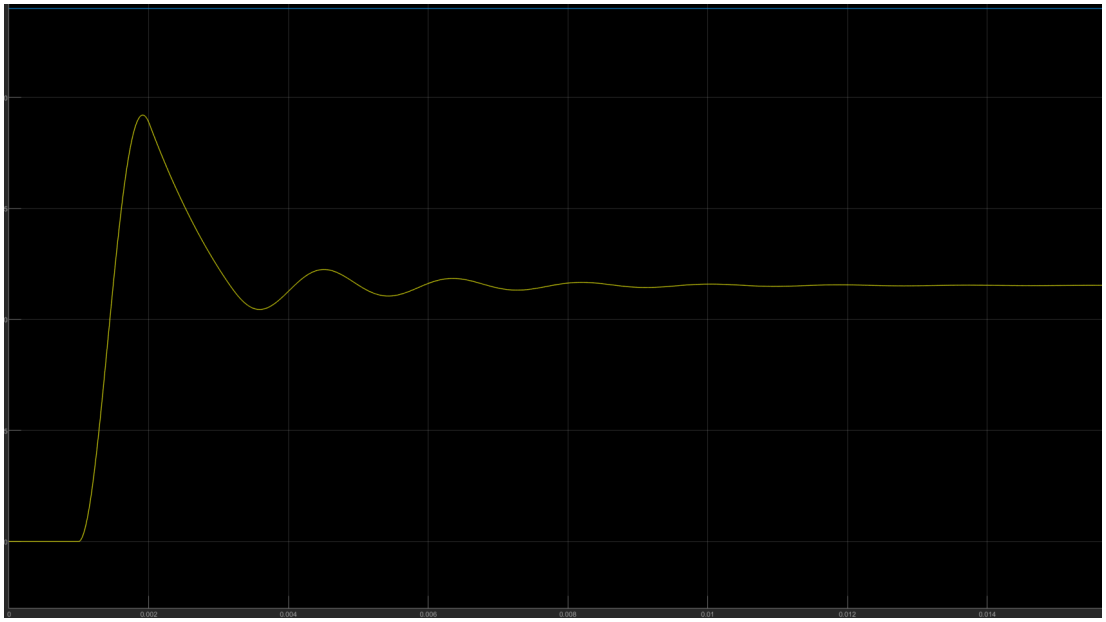


Figure 5.5: Generated circuit test.

highlighting the possibility of **integrating the modeling capabilities of EA with the computational and solver capabilites of Simulink.**
The current approach raises several challenges, first, the reliance on pre-existing Simulink blocks restricts the modeling process to a very low physical detail, where the goal is to enable high-level simulation, facilitating the conceptual representation and analisys of large-scale systems. It becomes evident that attempting to model an entire integrated circuit at this level of detail would quickly lead to uncontrollable complexity.
Furthermore, the generated Simulink model, while functional, lacks readability. The components are scattered throughout the workspace, and the connections between them are convoluted. This lack of organization and clarity would significantly impede the understanding of larger systems, requiring substantial effort to unravel and make the model more comprehensible.

5.2 MATLAB functions for high-level simulation

In the pursuit of a higher-level simulation, this section is dedicated to the exploration of MATLAB functions.
Given the functionalities offered by the sysPhS pattern, it becomes possible to create **custom Simulink blocks** within EA. These EA blocks are designed with ports and parameters that align with their Simulink counterparts.
The underlying idea is to utilize MATLAB functions encapsulated within MATLAB function blocks in Simulink. By doing so, the conceptual behavior of the

PMIC can be reproduced, enabling testing of its functionalities under various scenarios such as load steps, voltage reference variations, and other relevant factors.
Simulink is purpose-built for simulation tasks, providing a comprehensive suite of features for model creation and time-domain response analysis, and the utilization of MATLAB function blocks simplifies the implementation of code which emulates system behavior.
Leveraging this environment facilitates the simulation process and a high-level perspective is maintained, enabling the replication of system functionality without delving into physical details.
Simulink allows creating personalized libraries with custom blocks that can only be modified by the library owner, the process involves creating MATLAB function blocks to encapsulate the behavior of various model entities, then in the EA model, a linkage is established between the block definition and the corresponding Simulink custom block through the Simulink library path.
The modeling process remains unaffected, as all the modeling functionalities are preserved, with the addition that the structural block definitions include a designated section specifying the path of the associated Simulink block.
The adoption of MATLAB functions to simulate the system behavior represents a progressive step towards achieving a broader objective, which is not addressed in this particular work. The aim is to eventually replace them with VHDL behavioral models.

5.3 Custom Simulink blocks

Custom blocks development followed after series of feasibility tests, aligning with the buck EA model. Specifically, three sub blocks were created to accurately represent the system's functionality.
The output filter, designed as an LC filter transfer function incorporating a damping resistance R:

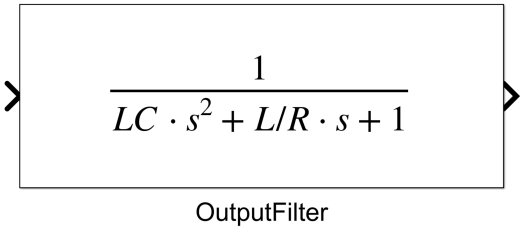
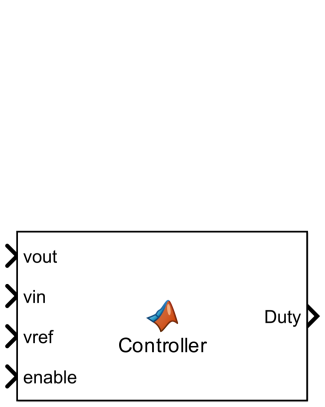


Figure 5.6: Output Filter transfer function.

For the control algorithm, decision was made to refrain from implementing a specific controller, such as PI or PID controller. Instead, a more general control logic was employed. It is important to acknowledge that this approach will not yield exceptional performance results, however, it is an integral part of the modeling process to refine and optimize these behaviors in order to attain better performance and accurately reflect the specific target requirements.



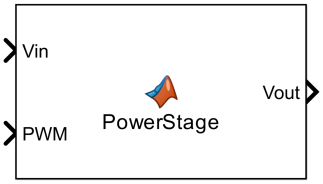

```
function Duty = Controller(vout, vin, vref, threshold, enable)
    persistent d;

    if(isempty(d))
        d=vref/vin;
    end

    if(enable~=0)
        Duty=d;
        if(vout>vref+threshold && d>=0.1)
            d=d-0.1;
            Duty=d;
        elseif(vout<vref-threshold && d<=0.9)
            d=d+0.1;
            Duty=d;
        end
    else
        Duty=0;
    end
end
```

Figure 5.7: Controller Block and Function.

The power stage is responsible for generating a square wave output that operates at the level of the supply voltage, its duty cycle is determined by the PWM controlling signal applied to it.




```
function Vout= PowerStage(Vin, PWM)
         Vout = Vin * PWM;

end
```

Figure 5.8: Power Stage Block and Function.

The voltage monitor block is implemented in the following code, which sets the over-voltage flag or under-voltage flag to a high state based on the output voltage exceeding the over-voltage threshold or falling below the under-voltage threshold. When either of these flags goes high, it is responsibility of the digital monitoring module to initiate the safety routine.

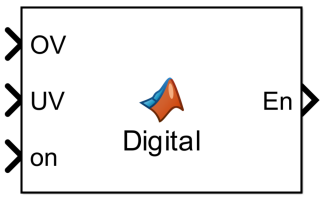
An important feature is that voltage monitoring must differentiate whether the under-voltage is caused by a regulation issue or is a result of the initial transient phase, during which the output of the buck converter is reaching its steady regulating state. This discrimination is accomplished by introducing


```
function [OV,UV]=vmonitor(vout, Vov,Vuv,Ts)
    persistent elapsedtime;
    timeout=5*Ts;

    if isempty(elapsedtime)
        elapsedtime=0;
    end
    OV=0;
    UV=0;
    if(vout>Vov)
        elapsedtime=elapsedtime+Ts;
        if(elapsedtime>timeout)
            OV=1;
            UV=0;
        end
    elseif(vout<Vuv)
        elapsedtime=elapsedtime+Ts;
        if(elapsedtime>timeout)
            OV=0;
            UV=1;
        end
    else
        elapsedtime=0;
        OV=0;
        UV=0;
    end
end
```

Figure 5.9: Power Stage Block and Function.

a fixed time interval delay after receiving the ”on” signal at the PMIC. PMIC digital control is implemented within the following block. When the turn-on external signal is activated, the control module enables the buck converter, allowing active power regulation to occur. Additionally, in the event that an over-voltage or under-voltage condition is detected by the voltage monitor, the control module disables the voltage regulation.




```
function En = Digital(OV, UV, on)
En=0;
if(on==0 || OV==1 || UV==1)
    En=0;
elseif(on>0)
    En=1;
end
end
```

Figure 5.10: Digital control Block and Function.

6

Simulation analyses

This chapter is dedicated to the designed system verification, wherein various analyses will be conducted under different operational scenarios of the PMIC. The simulation environment emulates the digital logic of the integrated circuit, employing a fixed sampling time as the step size for the simulink solver and for all the timings in the system. The following scheme is employed for simulation and system debugging purposes:

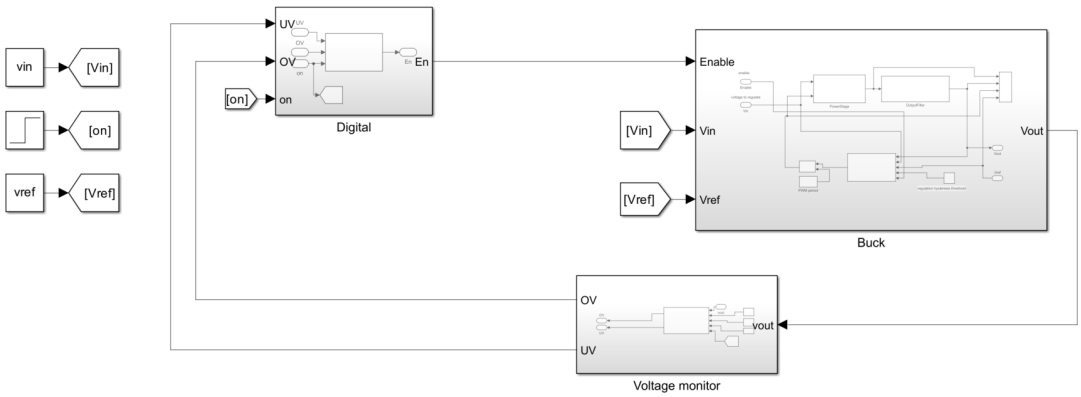


Figure 6.1: PMIC Simulink scheme.

It comprises three subsystems: voltage monitor, buck, and digital control. The voltage monitor and digital control blocks, which were previously created, are encapsulated within these subsystems to achieve a more organized and coherent system representation. The buck subsystem, consisting of its three components, is depicted in the figure below:

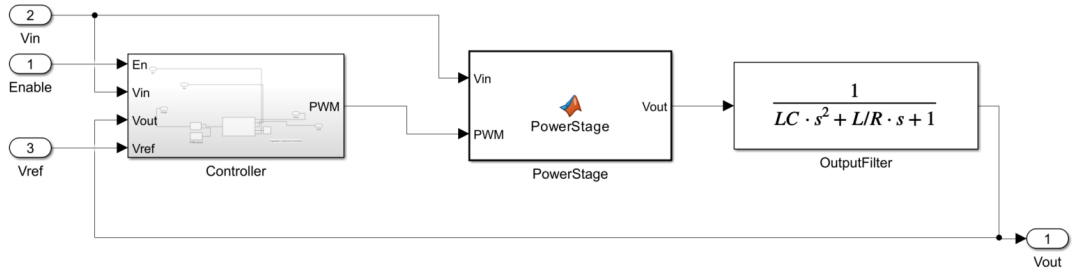


Figure 6.2: Buck Simulink scheme.

6.1 Step reference test

The aim of this analysis is to assess the regulatory performance of the system under steady reference conditions. As stated in the requirements, the system must maintain perfect regulation within the voltage interval of $[3, 20]V$, while operating within an input voltage range of $[5.5, 40]V$. To thoroughly evaluate the system's capabilities, tests will be conducted at the extremes of these operating range limits to subject the system to maximum stress.
Specifically, the following scenarios will be investigated:

- From $Vin = 5.5V$ to $Vout = 3V$.
- From $Vin = 40V$ to $Vout = 3V$

The first test illustrates the step down from $Vin = 5.5V$ to $Vout = 3V$, and the results can be evaluated in Fig. 6.3:

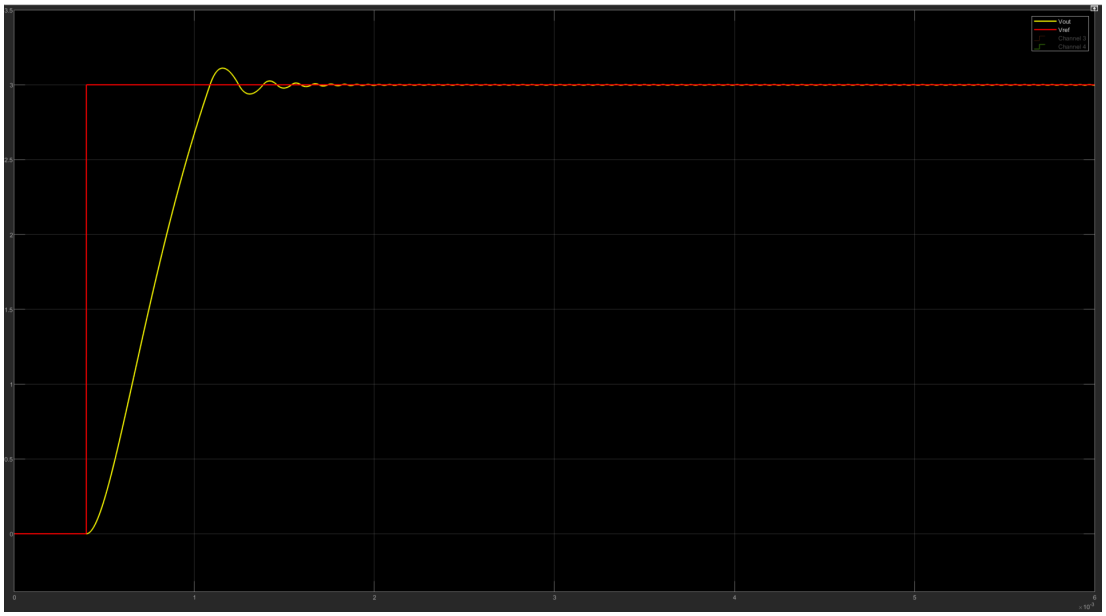


Figure 6.3: PMIC step reference waveforms.

Observations of the test reveal that the overshoot exhibited by the system is con-

siderably limited, measuring approximately $M_P \approx 8.3\%$. This restrained overshoot is to be attributed to the bandwidth of the controller, complemented by the small regulation step employed. Consequently, the step response demonstrates a clean and controlled behavior.
Furthermore, the steady-state regulating ripple is depicted in Figure 6.4. In this instance, the ripple settles at around $V_{RP} \approx 2.2mV$, falling well within the prescribed range of $V_{RPreq} = 6mV$, which corresponds to a 2% deviation as per the specified requirements.

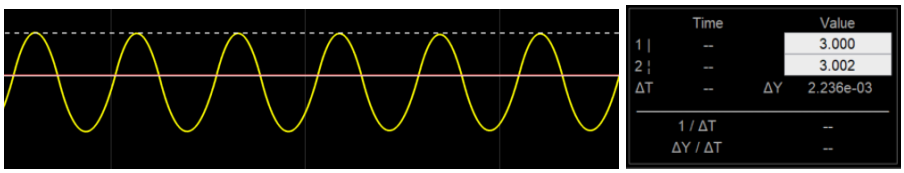


Figure 6.4: 5.5 to 3 V Regulation Ripple.

The second test is the $Vin = 40V$ to $Vout = 3V$ regulation, which produces the following results:

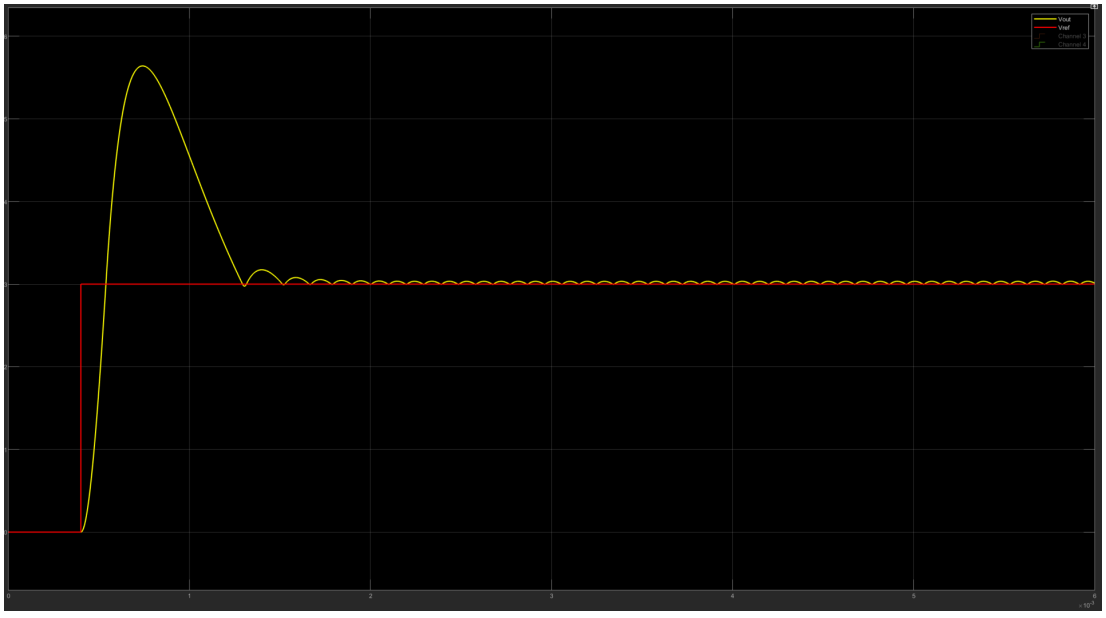


Figure 6.5: PMIC step reference waveforms.

In this scenario, it can be observed that the overshoot is significantly larger, approximately $M_P \approx 80\%$. To address this issue, further adjustments need to be made to the filter dynamics in order to slow down the transient response and reduce the overshoot. Additionally, improvements in the controller dynamics are required to enhance its speed. It may be beneficial to explore the implementation of known control strategies such as PI or PID controllers.

In Figure 6.6 the switching waveform generated by the power stage is illustrated, the active duty cycle adjustments performed by the controller are clearly visible in the waveform.

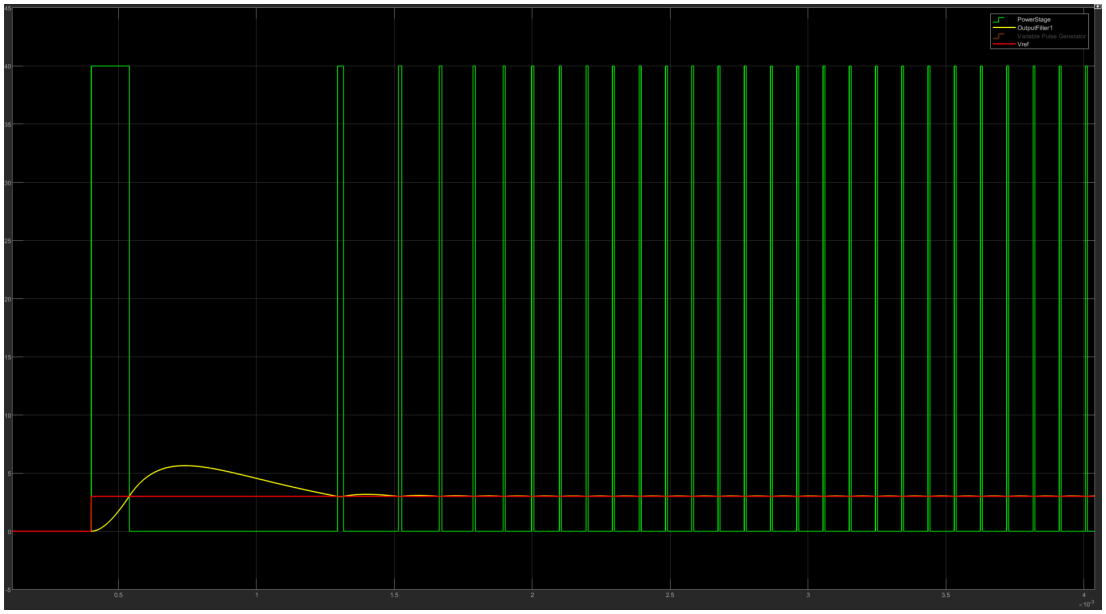


Figure 6.6: PMIC step reference switching waveform.

The steady-state ripple is higher, measuring around $V_{RP} \approx 36mV$ as in Fig. 6.7, value that does not meet the specified requirements. To cover this issue, two potential approaches can be considered, first, refining the output filter to provide better smoothing of the output voltage or, alternatively, discussions with stakeholders could be initiated to explore the possibility of relaxing the imposed requirements.

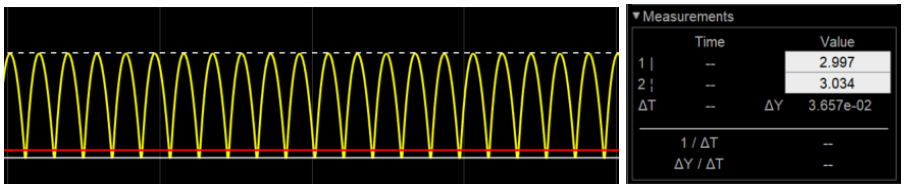


Figure 6.7: 40 to 3 V Regulation Ripple.

6.2 Dynamic reference test

In this analysis, the PMIC is subject to a square wave voltage reference to assess the controller's ability in tracking a changing reference.
One important aspect to note about dynamic reference tracking in buck converters is the ability of the system to accurately and quickly respond to changes in the reference voltage. Buck converters are commonly used in applications where the output voltage needs to be regulated and adjusted dynamically based on varying operating conditions or load demands; achieving precise and fast dynamic reference tracking is fundamental for ensuring stable and reliable operation.
Reference voltage oscillate at fixed frequency $f_{ref} = \frac{1}{T_{ref}} = \frac{1}{600 \cdot T_s} \approx 330Hz$, where $T_s = 0.5 \cdot 10^{-5}s$ is the switching frequency of the system.
The regulating ranges tested are the same as before, with the maximum excursion required for the output.

- From $Vin = 5.5V$ to square wave $Vout$ oscillating in the range $[3, 5.5]V$.
- From $Vin = 40V$ to square wave $Vout$ oscillating in the range $[3, 20]V$.

In the first test, the achieved outcomes can be summarized as follows: precise tracking, minimal ripple, and prompt settling of the $M_p \approx 11.3\%$ transition undershoot, as in Fig. 6.8.

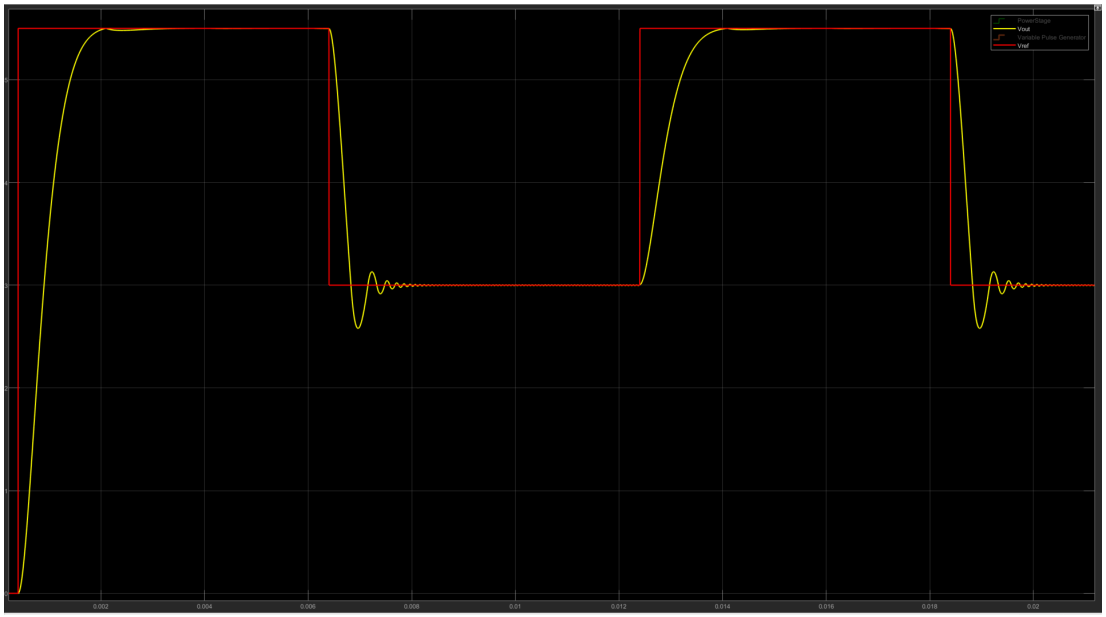


Figure 6.8: PMIC test 1 output.

Switching waveform is reported in Fig. 6.9.

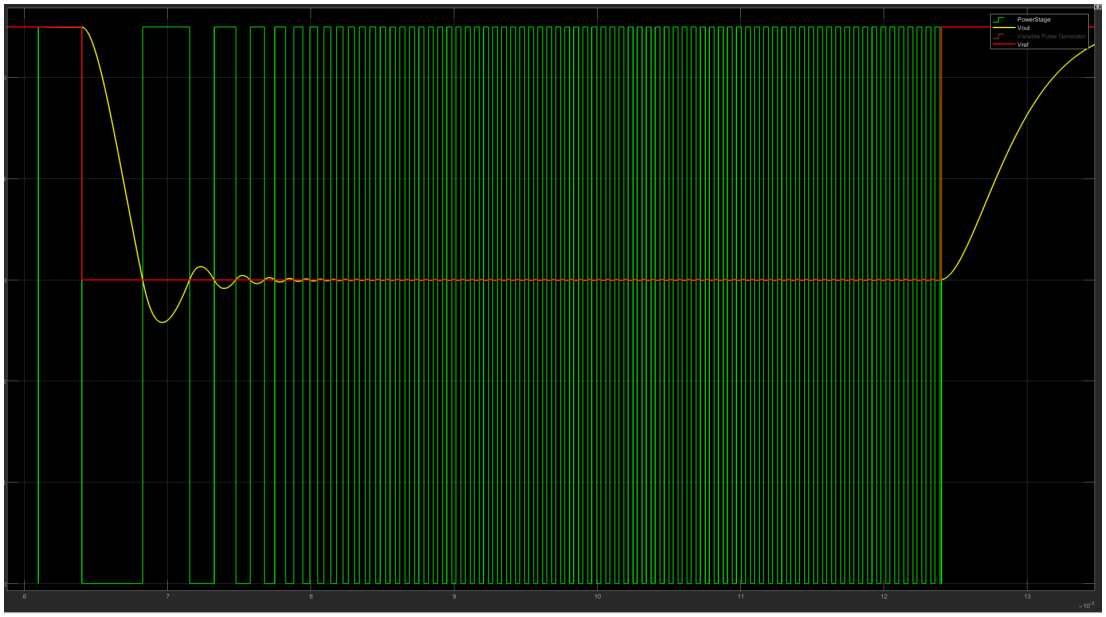


Figure 6.9: PMIC test 1 switching waveform.

The second test is performed at same switching frequency of test 1, except for an increased reference excursion, thus increasing tracking effort. Consequently, noticeable deviations become apparent, higher overshoot, reaching approximately $M_p = 12.5\%$, and a discernible ripple effect at the 3V regulation level, akin to the reference test conducted earlier.

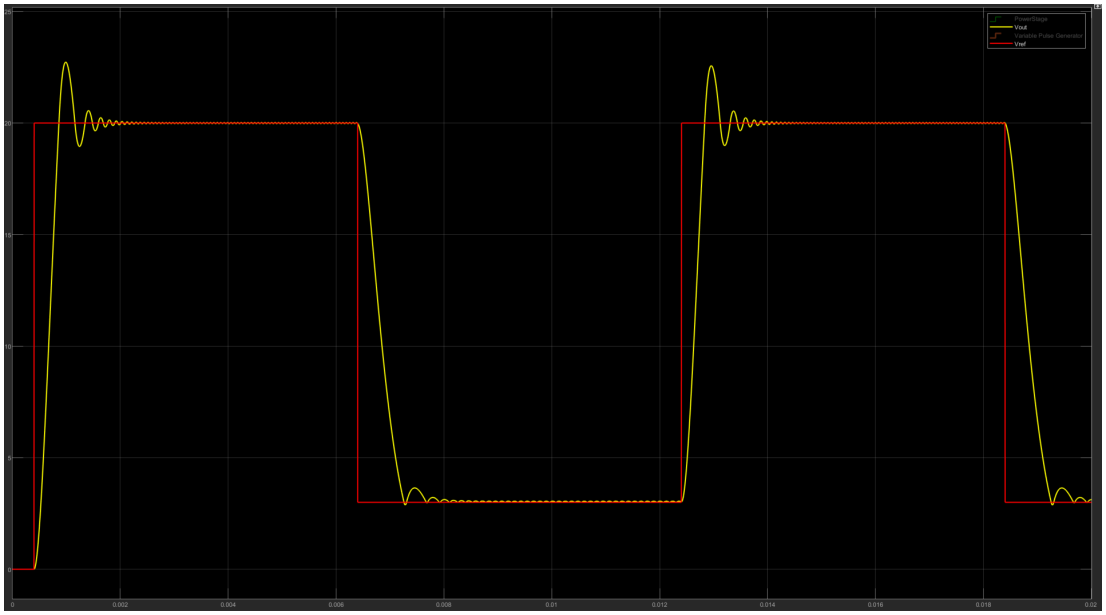


Figure 6.10: PMIC test 2 output.

As expected, the duration of the transient period is prolonged due to the larger reference step.
Interestingly, in contrast to the previous test, no undershoot is observed during the transition from $20V$ to $3V$, while the overshoot measured during the transition from $3V$ to $20V$ stabilizes at approximately $M_P \approx 16\%$. The switching waveform provides closer insight to the regulation throughout the transitions.

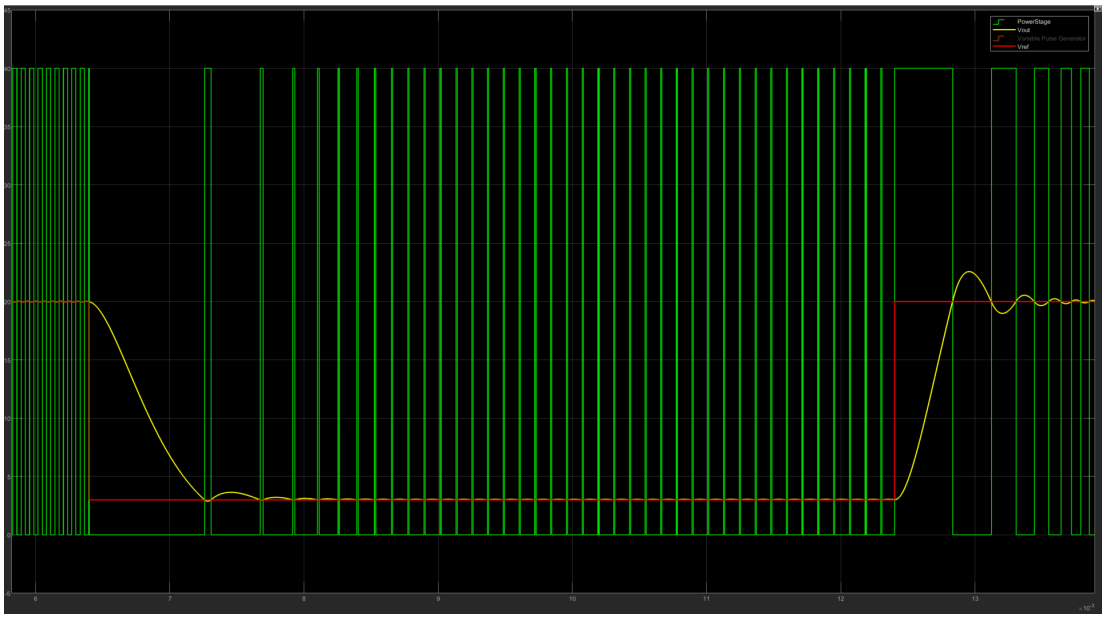


Figure 6.11: PMIC test 2 switching waveforms.

6.3 Load step test

A load step, specifically in the context of a purely resistive load, induces a modification in the transfer function of the filter and consequently affects the dynamics of the filter, altering the damping factor $\xi = \frac{1}{2 \cdot R_{load}} \cdot \sqrt{\frac{L}{C}}$ of the system. To modify the load resistance parameter $Rload$ is required to halt the simulation, manually adjust the $Rload$ value, and then resume the simulation. However, a straightforward, yet effective, solution has been devised. As depicted in the figure below, two filters with the intended difference in load resistance are generated. At a specific time, the output of the power stage is switched from the first filter to the second filter, effectively simulating a load variation.

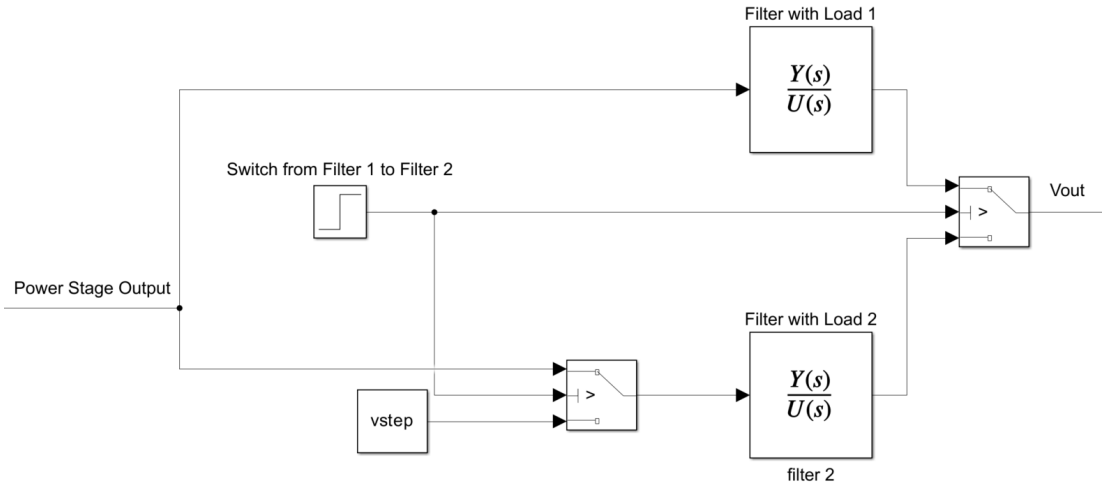


Figure 6.12: Load Step Setup.

The test is conducted in $40V$ to $3V$ regulation condition, the test load is doubled from $R_1 = 1\Omega$ to $R_2 = 2\Omega$, the corresponding output waveform is reported below:

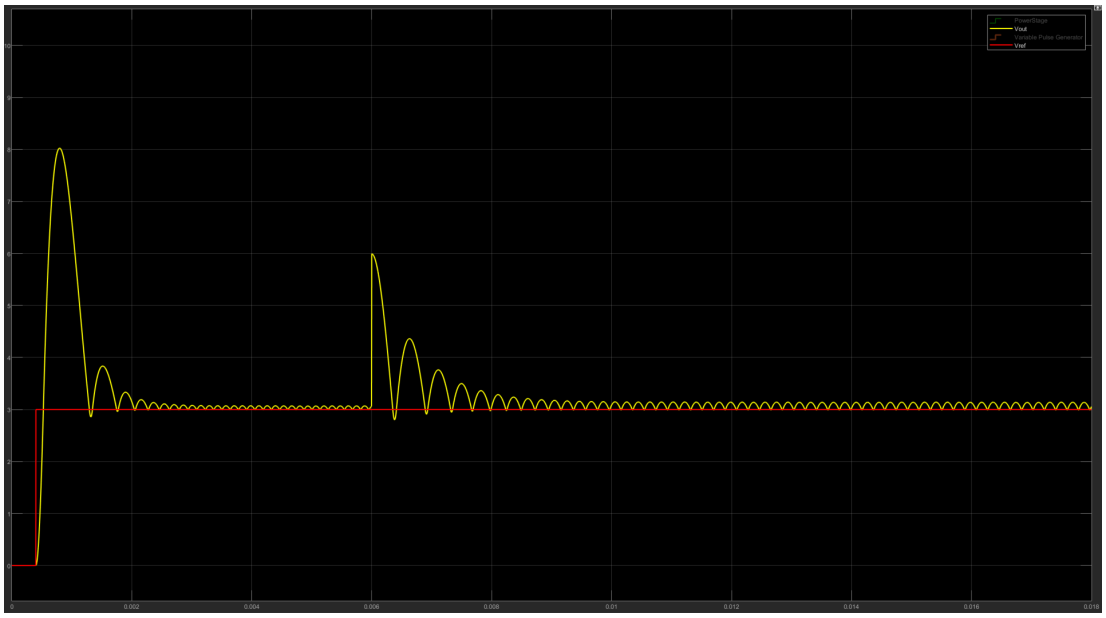


Figure 6.13: Load Doubling results.

The analysis reveals notable distinctions in the steady-state ripple between loads due to the variation introduced in the filter transfer function. A noteworthy observation is the spike that corresponds to the load variation, which is directly proportional to the ratio $\frac{R_2}{R_1}$. Consequently, it is mandatory to guarantee adequate protection in situations with significant load value difference.

The test next is conducted in the same regulation condition, now the test load is halved from $R_1 = 2\Omega$ to $R_2 = 1\Omega$, the corresponding output waveform is reported blow:

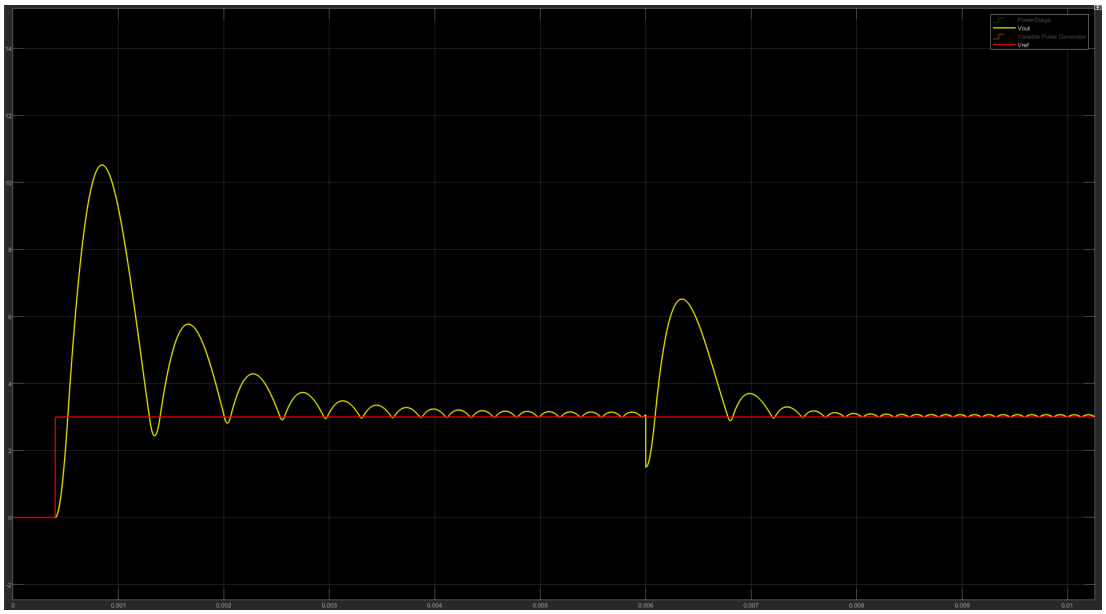


Figure 6.14: Load Halving results.

In the general scenario of a decreased load, the spike will consistently be lower than the reference voltage.

Conclusions and future works

The aim of Model Based System Engineering (MBSE) is to manage the complexity of present-day system design, by relying on formal structures defined by a standardized modelling language. SysML is an adaptation of the Unified Modelling Language (UML) standard to the requirements of system engineering, and in this thesis the use of a commercial tool based on it, called Enterprise Architect, has been assessed. Through this work, study of sysML language resulted in proficiency in the modeling technique, as well as a deeper comprehension of the objectives pursued by model-based systems engineering.
The multitude of benefits offered by this technique has been thoroughly explored, establishing foundation for the eventual embracement of this methodology in the development of extensive projects.
The main challenge during the course of this research pertained to the identification and utilization of appropriate sysML constructs and patterns for modeling a mixed-signal integrated circuit, encompassing the interplay between analog circuitry and digital modules, which necessitates distinct modeling approaches for each domain. It was fundamental to find a balance between the level of abstraction employed and the precision required to accurately capture essential details, all the while ensuring that the resultant complexity remained manageable. Attainment of this balance proved critical, as an excessive level of abstraction risked overlooking crucial interactions or phenomena, while a higher level of detail could lead to an excessively detailed abstraction level model, conducting to lengthy simulation duration and scalability problems. In this regard, Enterprise Architect exhibited remarkable versatility, enabling the practical actuation of the model-based systems engineering methodology and facilitating the management of design requirements and the overall modeling endeavors.
The verification phase of the model was also investigated, where it became evident a dedicated simulation tool was necessary to facilitate this aspect.
Simulink emerged as a suitable choice due to its robust solver capabilities, EA-like block structure framework and libraries, key in organizing and reusing refined blocks. This integration enabled simulations on the model, allowing for

the fine-tuning of behaviors, in parallel to the EA description, which lead to a **complete model across two integrated platforms.**
Future research will focus on the exploration of advanced modeling tool features, to enhance the design process in terms of efficiency and effectiveness, and the integration of Enterprise Architect within the Infineon design workflow will be studied.
An important milestone entails investigating methods to integrate existing HDL descriptions of blocks into the system model, facilitating the verification process without the need for writing Matlab code, as was done in this study.

References

[1] London Brian. "Documentation." In: *A Model-Based Systems Engineering Framework for Concept Development*. 2012, pp. 40, 41.

[2] London Brian. "Specifying Customer Requirements in a Model Based Systems Engineering Environment." In: 2011.

[3] Sam Mancarella Doug Rosemberg. "Embedded Systems Development using SysML". In.

[4] Ivar Jacobson Grady Booch James Rumbaugh. "The Unified Modeling Language User Guide". In: May 19, 2005.

[5] "ISO/IEC 19501:2005 - Information technology - Open Distributed Processing - Unified Modeling Language (UML)". In: April 1, 2005.

[6] Nataliya shevchenko. "An Introduction to Model-Based Systems Engineering." In: December 21, 2020.