POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Real-Time Emulation of a Power Converter using Microcontroller-Based Systems

Tesi di Laurea Magistrale in
Electronic Engineering - Ingegneria Elettronica

Author: **Giulio Salbego**

Student ID: 252000
Advisor: Prof. Luigi Piegari
Co-advisors: Daniele Caltabiano
Academic Year: 2025-2026

Abstract

This thesis investigates the potential of the Hardware-in-the-Loop (HIL) approach for the development and validation of models dedicated to power electronics applications. In particular, the work focuses on the detailed analysis of a widely adopted high-efficiency AC–DC converter topology known as the Totem-Pole converter. The topology is thoroughly examined, mathematically modeled, and subsequently implemented on an STMicroelectronics development platform included in the STEVAL-HILE1G4K1 reference kit.

The kit consists of two boards: the first is used to emulate the reference power converter model, while the second operates as the controller, executing the regulation algorithms required for the converter under investigation.

The thesis will therefore begin with a careful analysis of the initial choices made, the reasons for using this approach, and the actual advantages and difficulties of undertaking this work. Special attention will be paid to a detailed description of the modeling tools used and the various implementation steps, in order to obtain clear and transparent results.

The developed model is then validated by executing the theoretical converter model on the HIL board and interfacing it with the control firmware used in the STEVAL-7BIDIRCB product. This firmware generates the actual control signals that would be applied to drive the physical converter topology. Such an approach enables a comprehensive assessment of the proposed model and provides a practical demonstration of its correct operation within a realistic control environment.

The ultimate goal of this thesis is the development of a computationally efficient and highly accurate digital model of a power electronic converter, capable of faithfully reproducing the behavior of the corresponding analog system even at high switching frequencies, while leveraging a low-cost microcontroller as the emulation platform.

Keywords: Digital emulation, Hardware In the Loop (HIL), Model Based Design (MBD)

Contents

Introduction

In recent years, power electronics has assumed an increasingly central role in the development of highly efficient energy systems, finding application in numerous sectors such as electric mobility, renewable energy, industrial systems, and consumer electronics. The growing demand for high performance, reduced energy losses, and increased power density has led to the development of increasingly sophisticated converters, featuring high switching frequencies and advanced control strategies.

In this context, design and validation methodologies have also undergone significant transformation. This has led to the need for new techniques for low-cost rapid prototyping. Many companies require these types of structures, which, through digital emulation, aid and fuel the development of new high-efficiency converters.

Hardware-in-the-Loop (HIL) is a well-established design and validation technique that enables simulation, testing, and debugging of complex electronic systems. The core idea is to replace a portion of the physical system with a simplified model that replicates, with high fidelity, the same behavior as perceived by the remaining components. In this way, it is possible to test the overall system by combining real hardware with simulated elements.

A natural question arises:

> *"Why adopt this approach if the real system is already available?"*

The answer can be summarized in three key points:

- Ability to continue testing activities even when some system components are not available
- Risk reduction when testing new and unvalidated systems
- Increased flexibility and speed in modifying system parameters

When developing complex systems composed of multiple subsystems, it is common that some components are not yet available. This may be due to incomplete development at the testing stage or to practical difficulties in physically accessing the hardware. This

issue is particularly relevant in power electronics, where loads often consist of industrial equipment installed in specific locations and are therefore not easily transportable.

As a result, testing control strategies for power converters supplying such loads may require direct access to the physical installation.This introduces logistical challenges due to the need to reach potentially distant locations, as well as significant risks, since unsuccessful tests may damage expensive equipment. For this reason, developing an emulation of these subsystems makes it possible to test and debug, for instance, control units and embedded controllers directly on the emulation platform. Preliminary tests can then be performed simply by flashing the emulated model onto the device, connecting the controller under test to its I/O pins, and observing the behavior of the *Device Under Test* (DUT), namely the real hardware portion of the system.

As can be easily understood, such an approach is both highly safe and portable. Indeed, HIL platforms can emulate high-power loads, generators, and devices absorbing several kilowatts while operating on a compact low-power board consuming only a few watts. Consequently, risks related to electric shock and the need for dedicated high-power laboratory facilities are significantly reduced or eliminated.

Another noteworthy aspect of the proposed approach is the flexibility with which the model can be modified and adapted. By simply updating the firmware, it is possible to implement and emulate a wide range of converter topologies and electronic components without requiring changes to the underlying hardware platform. This capability enables the execution of numerous test scenarios under specific operating conditions with minimal setup effort, significantly reducing the need for complex laboratory equipment and dedicated prototypes.[1]

With these considerations in mind, this thesis explores the Hardware-in-the-Loop paradigm through the following stages [2]:

1. selection of the emulation platform;
2. selection of a target subsystem to emulate (a Totem-Pole AC-DC power converter);
3. development of a model compatible with the chosen emulation board, based on a study of the converter topology;
4. validation of the model through software simulations using Simscape;
5. implementation of the model on the hardware platform;
6. verification, through hardware testing, that the emulation board behaves as expected within the system.

1 | Hardware-in-the-Loop (HIL)

1.1. Concept

As said before Hardware-In-the-Loop is a technique in which the idea is to replace a portion of the physical system with a simplified model that replicates, with high fidelity, the same behavior as perceived by the remaining components.

By the way if in the introduction we gave a motivation for moving towards this type of digital emulations, working on power electronics a further question may arise:

> *"How can a low-power device such as a HIL platform emulate high-power systems rated in the order of tens of kilowatts while maintaining high fidelity?"*

Once again, the answer is relatively straightforward, although some clarifications are required.

First of all, the HIL platform does not replace the actual device during real operation; rather, it reproduces only the effects observable at its electrical interfaces. More precisely, the HIL board emulates not only the subsystem of interest, but also the sensors and communication signals required to interact with the remaining parts of the system. Naturally, the platform operates with low-voltage electrical signals (typically in the 0–5 V range). Therefore, the firmware programmed onto the HIL device must describe as accurately as possible both the subsystem behavior and the associated sensing interfaces.

An example can help clarify this concept.

Consider a system composed of an electric motor and its control unit. In order to test the controller in the conventional way, it would be necessary to program the control board, connect the sensors and the motor drive stage, supply both the controller and the motor with the required power, and finally observe the control response. Assuming the use of a microcontroller produced by STMicroelectronics, the debugging process could be carried out through STM32CubeIDE, allowing the monitoring of variables and execution flow during runtime.

At this point, the role of the HIL platform becomes evident. In such a system, instead of connecting the controller to the real motor, the controller can be directly interfaced with the HIL device. The control board is programmed with the original firmware, while the HIL platform is flashed with a dedicated model-based firmware that reproduces, at its output interfaces, the same signals that the real motor would generate in response to the controller commands.

Therefore, the HIL platform does not perform the actual mechanical work of the motor; rather, it faithfully emulates the sensor-mediated electrical behavior as perceived by the controller. Assuming an accurate emulation model, the results observed in STM32CubeIDE during controller execution should be identical, or as close as possible, to those obtained when operating with the real motor.

Consequently, the physical motor is no longer required during the preliminary validation phase of the controller, allowing all the previously discussed advantages in terms of safety, portability, and convenience.

To conclude the example, consider a situation in which the motor experiences an overcurrent condition due to an immature controller algorithm that fails to reduce the current promptly. If the real motor were used, the machine could overheat, become damaged, or even fail irreversibly. By contrast, with the HIL platform, the engineer would simply observe an excessive current value reported to the controller. No physical damage would occur, since the HIL does not absorb the actual high power involved. Nevertheless, the exact information required to improve the control strategy would still be available before real-world deployment.

Let us now consider the major advantage of HIL systems: parameter tuning and configuration flexibility.

Continuing with the previous example, the technical characteristics of the emulated subsystem can be easily modified by changing the parameters of the firmware deployed on the HIL platform. In this way, the same real system can be virtually reproduced under multiple operating conditions.

For instance, if the control board must be tested with different motor configurations, a traditional setup would require physical modifications of the machine, such as rewinding stator windings or applying adjustable mechanical loads to the shaft. Such hardware changes are often expensive, time-consuming, and dependent on the availability of dedicated resources.

On the other hand, with an HIL platform, these variations can be reproduced almost

instantaneously by simply editing firmware parameters. Moreover, even more advanced scenarios become possible. In many cases, it is desirable to test the system within a specific operating region, for example under steady-state conditions. When using real hardware, each test must start from the initial startup sequence, and time is required before reaching the desired operating point.

With HIL emulation, since the system behavior is described through mathematical equations, the simulation can start directly from any desired initial condition, eliminating unnecessary waiting times. Similarly, the introduction of fault conditions or abnormal scenarios can be performed rapidly and in a fully controlled manner.

Naturally, developing a detailed and accurate real-time model is not a trivial task. Nevertheless, the advantages offered by this approach are undoubtedly significant.

1.2. State of the art

Numerous commercial platforms are currently available for Hardware-in-the-Loop applications. One of the main technological distinctions among these solutions concerns the computational architecture adopted by the processing board, which is generally based either on *Field Programmable Gate Arrays* (FPGA) or on Microcontrollers.

These two classes of devices differ substantially both in operating principle and in development methodology. Although both are programmable platforms, they rely on distinct design paradigms.

Microcontrollers are typically programmed using sequential languages such as C/C++, or through higher-level environments such as MATLAB/Simulink, where control models and algorithms can be automatically converted into C code through code-generation tools. In these devices, execution is performed by an integrated CPU that processes instructions sequentially, while internal peripherals—such as timers, analog-to-digital converters (ADCs), PWM modules, and communication interfaces—operate as auxiliary units and interact with the main program flow primarily through interrupts or direct memory access mechanisms.

By contrast, an FPGA is not programmed by defining a sequence of instructions to be executed, but rather by directly describing the desired hardware architecture. This process is carried out through *Hardware Description Languages* (HDL), among which the most common are VHDL and Verilog. The acronym VHDL stands for *Very High Speed Integrated Circuit Hardware Description Language*. Through these languages, the designer specifies logic blocks, finite-state machines, arithmetic units, registers, and interconnections,

which are subsequently synthesized and physically implemented within the programmable device.

The resulting structure is a dedicated hardware architecture capable of executing multiple operations in parallel, with extremely low latency and deterministic timing behavior.

From the perspective of HIL applications, where real-time execution and high fidelity in reproducing physical phenomena are required, FPGA-based solutions often represent the most suitable choice. Real systems are inherently continuous in time and frequently characterized by very fast dynamics. The capability to perform parallel computations, guarantee extremely short sampling times, and maintain strictly deterministic timing makes FPGAs particularly well suited for the numerical emulation of such systems.[3]
FPGAs, however, present two fundamental drawbacks:

- Slow development and programming time
- High cost

Indeed, the execution of the bitstream for relatively simple circuits emulation, even those composed of a limited number of high-frequency components, may require several hours or even days. This is due to the fact that FPGA bitstream generation involves multiple computationally intensive stages, including logic synthesis, hardware resource mapping, and, most critically, the *place and route* process.

The latter consists of the physical placement of logic blocks and the routing of internal connections while satisfying strict area and timing constraints. This task can be interpreted as a complex combinatorial optimization problem, which often requires multiple iterations before converging to a valid and timing-consistent solution.

In addition, FPGA devices are significantly more expensive compared to alternative solutions. Their cost typically ranges from hundreds to thousands of euros, whereas microcontrollers are generally available at a cost of a few tens to a few hundred euros, even for relatively advanced devices.

Given these considerations, the use of microcontrollers becomes increasingly competitive. Although this choice leads to a reduction in accuracy when reproducing very high-frequency dynamics, it provides substantial advantages in terms of cost and, in particular, development simplicity.

This aspect is far from trivial. While large companies can afford to enter the market by acquiring dedicated emulation devices, small and medium-sized enterprises often cannot. By adopting a microcontroller-based approach, it becomes possible to reach a broader

and strategically important customer base, making emulation more accessible prior to final implementation.

From a market feasibility perspective, particularly in an industrial context, it is important to note that this type of emulation is not intended to be integrated into the final system, but rather to serve as a debugging testbench. For this reason, a low-cost solution is significantly more attractive.

Given the current trend toward boards featuring combined FPGA–CPU architectures, it is useful to analyze what the market currently offers and to evaluate the performance of the various boards specifically designed for Hardware-in-the-Loop (HIL) emulation that are presently available.

1.2.1. FPGA Based

Typhoon HIL

Typhoon HIL is a company specialized in hardware-in-the-loop (HIL) solutions for the real-time simulation and testing of power electronics systems.

The model is built using a proprietary schematic editor, in which the user selects and connects typical power electronics components such as semiconductor switches, diodes, passive elements, and electrical machines. Once the circuit and the control logic are defined, the system automatically compiles the model into an optimized FPGA representation.

This process is fully transparent to the user, who is not required to interact directly with low-level hardware programming. This approach enables rapid prototyping and significantly reduces implementation complexity, making these platforms particularly suitable for academic environments and for applications focused on switching power converters.

OPAL-RT Technologies

A more flexible but also more complex approach is adopted by OPAL-RT Technologies.

In this case, the modeling is typically performed in MATLAB/Simulink or through dedicated tools such as HYPERSIM. The model is then partitioned into different parts depending on the system dynamics: slow dynamics are executed on a real-time CPU, while faster dynamics, particularly those related to power device switching, are implemented on FPGA hardware.

This partitioning step requires a deep understanding of the system and of its timing

constraints, but it enables high scalability and the simulation of complex multi-domain systems.

The workflow therefore involves separate code generation for the CPU and FPGA targets, followed by deployment on the real-time hardware platform.

Plexim

The solution proposed by Plexim through the *PLECS RT Box* platform represents one of the most straightforward workflows.

The model is developed within the PLECS environment, which is also used for offline simulation. Once validated, the same model can be executed directly on the real-time platform without substantial modifications. This is made possible by a discrete solver that is specifically designed to satisfy real-time constraints.

This approach eliminates the need for hardware-specific adaptations or optimization steps, making the platform highly accessible. However, it is less flexible in terms of customization and may be less suitable for applications involving extremely fast transient phenomena.

Speedgoat

In the case of Speedgoat platforms, the modeling workflow is tightly integrated with the Simulink environment.

The model is developed using standard Simulink blocks and then configured for real-time execution by selecting a fixed-step solver and ensuring compliance with timing constraints. The code is automatically generated via Simulink Coder and executed on a real-time target machine.

In this context, the FPGA is typically used for deterministic handling of I/O operations or for specific computational tasks, while the main simulation runs on the CPU. This approach represents a good trade-off between simplicity and flexibility, although it may become limiting for applications involving very high switching frequencies.

National Instruments

An even lower-level approach is offered by National Instruments solutions, based on PXI architectures with programmable FPGA modules.

In this case, modeling can be performed using LabVIEW FPGA or directly through hardware description languages such as VHDL or Verilog. The user has direct access to

the logic implemented on the FPGA and can precisely define system behavior, including execution timing and hardware resource management.

This provides maximum flexibility and high performance, but requires advanced expertise in digital design and typically results in significantly longer development times.

All these hardware configurations represent excellent solutions from a technical standpoint. Indeed, they provide multiple high-frequency analog and digital I/O channels that are fully configurable and suitable for a wide range of real-time applications.

However, the main drawbacks are represented by their extremely high cost and the overall implementation time. The price of a single development platform can range from tens to hundreds of thousands of euros, to which several additional thousands of euros must be added for the software licenses required to develop the model. As an example, a MATLAB license costs approximately €2,645, while an individual Plexim license is around €8,000.

Of course, the concept of cost is relative. However, when the objective is to identify a low-cost emulation platform, with a target budget in the order of a few hundred euros, these development platforms become considerably less accessible, especially in academic or early-stage prototyping contexts.

Furthermore, the advantages gained during model development at the software level are often partially offset by long synthesis and compilation times, since high-frequency simulations must be implemented and mapped mainly onto FPGA hardware.

The following table presents the main boards, enabling a direct comparison in terms of performance and cost—these being the only truly objective parameters, as opposed to the inherently subjective nature of software usability.

Platform	Indicative Cost	Hardware	Typical Timestep	Development Environment
Typhoon HIL604	∼120k €	Proprietary FPGA	20 ns	Typhoon HIL Control Center
OPAL-RT OP5707	∼100k €	Intel CPU + Xilinx FPGA	145 ns	RT-LAB / HYPERSIM / Simulink
PLECS RT Box 1	∼18k €	FPGA + CPU	7.5 ns	PLECS Blockset + RT Box Toolchain
Speedgoat Performance	∼60k €	Intel CPU + FPGA	∼ 100 ns	MATLAB/Simulink Real-Time
NI PXIe-7822R	∼12k €	Xilinx FPGA	20 ns	LabVIEW FPGA / NI VeriStand
ST HIL board	∼200 €	MCU	∼10 us	Stellar-Studio (free) / Simulink

Table 1.1: Concrete examples of FPGA-based HIL platforms for power electronics applications [4–8]

1.2.2. Microcontroller Based

Restricted to microcontroller-based HIL solutions, the market is relatively limited: except with STMicroelectronics with the *STEVAL-HILE1G4K1* kit [9] there are no large-scale industrial platforms dedicated exclusively to this approach, but rather relevant technological families and reference solutions is proposed by different manufacturers.

While the FPGA-based HIL market is well established, the use of microcontrollers in this context represents a more recent direction of innovation and ongoing research.

The general approach consists in providing hardware, software, and reference designs that enable the development of simplified HIL systems, in which the microcontroller can act either as the controller under test or as a low-complexity system emulator.

Indeed, as observed in the analysis of FPGA-based boards, the inclusion of a CPU component within the architecture responsible for executing a significant portion of the computations is a common feature in the OPAL, Speedgoat and Plex offerings.

With the significant increase in microcontroller computational performance, many emulation tasks can now be effectively handled by ultra-low-cost devices while still achieving a satisfactory level of fidelity.

Apart from some CPU solutions that are not very relevant and often used only as an integration to FPGA-based boards, only STMicroelectronics offers an alternative to the previous HIL development boards.

STMicroelectronics

The solution currently available on the market, released at the beginning of this year, is the HIL board included in the *STEVAL-HILE1G4K1* kit. This kit represents one of the most advanced solutions within the STM ecosystem oriented towards real-time emulation and HIL applications.

This board is probably the only microcontroller fully dedicated natively for the HIL implementation.

The hardware platform embeds a set of peripherals tailored for power electronics and control applications, such as precision PWM generation units, fast parallel ADC acquisition channels, and standard industrial communication buses (CAN, SPI, UART). The architecture is designed to ensure deterministic coordination between sensing and actuation paths, minimizing control-loop delays to the microsecond range, which makes it particularly effective for real-time Hardware-in-the-Loop emulation of systems like electric drives

and power conversion stages.

In addition, the system benefits from an integrated graphical interface that allows continuous real-time observation of signals, streamlined debugging operations, and straightforward interaction with the eDesignSuite environment. Structure which significantly improving the overall development and validation workflow.

Since this board is the one adopted for the emulation development presented in this thesis, the following section is entirely dedicated to its description and to the motivations why this kit currently represents the best alternative in terms of cost, reliability, design simplicity, and portability.

1.3. HIL board

At STMicroelectronics, and more specifically within the laboratory where this thesis work is being carried out, the second approach microcontroller oriented has been adopted. More precisely, for HIL applications in power electronics, it is generally not necessary to emulate systems operating at extremely high frequencies, since the dominant dynamics are inherently in the order of kilohertz.

As a consequence, the use of a microcontroller operating in the range of several hundreds of kilohertz provides a level of fidelity that is fully compatible with the target application, while simultaneously benefiting from the advantages previously discussed in terms of cost reduction and development simplicity.

1.3.1. Functionalities

As laboratory for the various HIL projects has been adopted a custom HIL board designed internally by *Daniele Caltabiano* . The reference kit through which this hardware platform can be acquired is the STMicroelectronics evaluation kit named *STEVAL-HILE1G4K1* [9].

The kit includes two separate boards:

- Emulator board based on the Stellar E1 microcontroller
- Control board featuring the STM32G474RE microcontroller

The kit has been specifically designed to provide high versatility and compatibility across different applications, particularly with respect to the accessibility and exposure of the microcontroller pins. Furthermore, a common standard has been adopted for LEDs, push-

buttons, and pinout organization in order to maintain consistency among the various applications. Nevertheless, the microcontroller configuration can still be adapted and customized according to the specific requirements of each project.
Let us initially focus on the HIL board included in the kit.

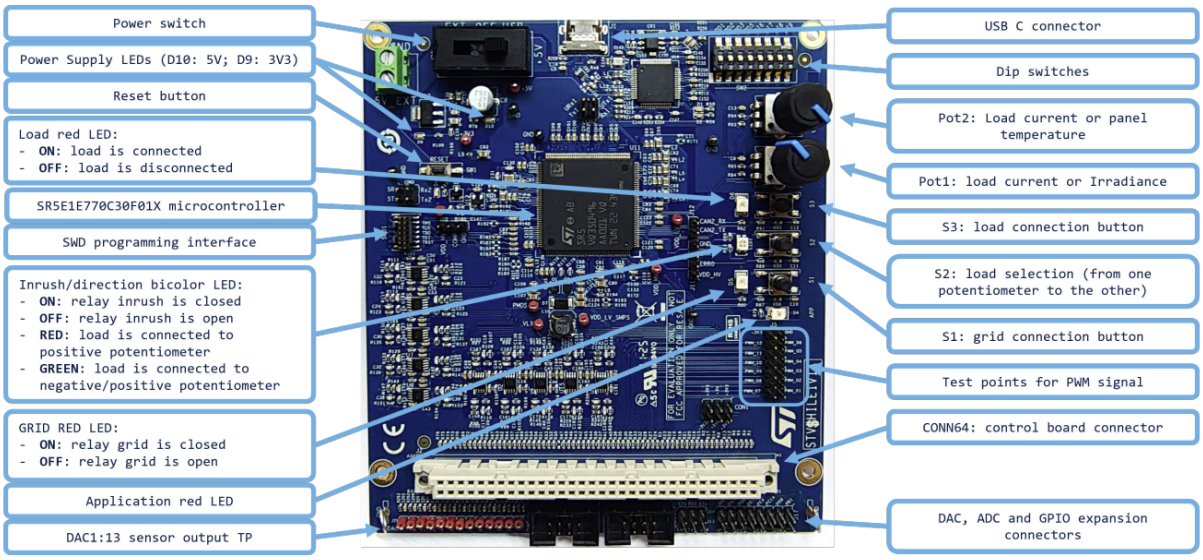

Figure 1.1: HIL board

As can be observed, the board provides multiple connectors that can be used both for the generation and acquisition of analog and digital signals. These interfaces can be connected to external boards through terminal blocks, while dedicated expansion headers allow convenient oscilloscope probing directly on the board.

The main connector is the `CONN64`, which exposes:

- 13 12-bit Digital-to-Analog Converters (DACs)
- 12 Timers (Output PWM or Input Capture mode)
- 2 UART TX/RX channels
- 3 Successive Approximation Register (SAR) Analog-to-Digital Converters (ADCs)
- 11 Microcontroller GPIOs configurable in Alternate Function mode if required
- Power supply and GND connections

The details of the pinout will not be discussed at this stage, since they will be analyzed later in the context of the specific application presented in this thesis.

The same signals are also exposed through grouped dedicated connectors:

- DAC channels 1:13 located at the bottom-left side of Figure 1.1
- GPIOs and ADCs located at the bottom-right side of Figure 1.1
- PWM outputs or input capture channels located on the right side of Figure 1.1

The board also includes a main power switch and three push-buttons (`S1`, `S2`, `S3`), which, together with two onboard potentiometers, allow runtime modification of HIL parameters in order to emulate different system configurations.

Additionally, the board features 8 DIP switches, 8 LEDs, UART communication terminals, SWD debugging interfaces, and a USB-C connector, all of which are useful both as peripherals and for programming the *Stellar E1* microcontroller.

1.3.2. Model-Based Design Coding

Two different approaches can be adopted for programming the microcontroller:

- StellarStudio / STM32CubeIDE
- MATLAB Simulink + Hardware Support Package

Traditionally, STMicroelectronics microcontrollers have been programmed in C language, supported by integrated libraries providing *Hardware Abstraction Layer* (HAL) functions. These functions are primarily intended to simplify the management and configuration of the various microcontroller peripherals.

When the target device belonged to the automotive-oriented families, the adopted *Integrated Development Environment* (IDE) was typically *StellarStudio*, whereas *STM32CubeIDE* was commonly used for STM32 microcontrollers. These development environments allow the user to write, compile, and upload firmware directly onto the target board, integrating within a single platform the code editor, compiler, and debugging tools.

However, the approach promoted by STMicroelectronics and extensively adopted within our laboratory consists in using a different development environment based on MATLAB, and more specifically on *Simulink Embedded Coder* coordinated with the dedicated *Hardware Support Package* provided by the company.

This environment enables the programming of microcontrollers through the automatic generation of source code starting from graphical models developed in Simulink. Compared to a traditional IDE, where the developer manually writes firmware in C/C++, the model-based approach relies on block diagrams describing algorithms, control strategies, and logical flows; the corresponding source code is then automatically generated by the

code generation engine.

From a technical perspective, a traditional IDE provides direct control over memory allocation, hardware registers, interrupts, scheduling mechanisms, and peripherals, making it particularly suitable for low-level development and resource optimization. Simulink, on the other hand, introduces a higher level of abstraction by enabling simulation, model verification, and automatic firmware generation, although at the cost of reduced direct control over the hardware implementation and the final generated source code. The remarkable advantage of using the *Hardware Support Package* (HSP) for code generation does not lie solely in the intuitive *Model-Based Design* approach offered by Simulink, but also in the complete compatibility that allows even complex Simulink blocks to be compiled and deployed directly into the microcontroller firmware simply through a drag-and-drop workflow within the model. [10]

As a result, the entire firmware structure becomes significantly more readable and maintainable compared to a traditional C implementation.

Furthermore, the use of an environment such as MATLAB greatly simplifies the creation of testbenches for firmware development. This is possible because all blocks, except those specifically related to the HSP hardware interfaces, can be fully simulated directly in Simulink without the need to flash the firmware onto the target microcontroller, thus enabling preliminary validation of the control logic. [11]

In many cases, step signals or dedicated test waveforms are applied at the inputs of the simulated firmware, effectively emulating the behavior of real hardware inputs. The response of the block-based implementation can then be analyzed and validated before deploying the firmware onto the actual board, where these signal generators are replaced by the physical HIL peripherals.

An additional and extremely valuable feature is provided by the *Data Inspector*, a Simulink tool that allows monitoring, recording, and plotting virtually every internal firmware variable, thereby enabling precise analysis of the system behavior.

This functionality is available both during standalone Simulink simulations and while operating the HIL board in *External Mode*. This operating mode is conceptually very similar to the *Live Expressions* feature available in the STM32CubeIDE debug environment, while additionally benefiting from the advanced plotting and visualization capabilities provided by MATLAB.

The communication between the target board and the host computer is performed through the USB-C serial interface and is therefore limited by the serial baud rate. Nevertheless,

for low-frequency signals, the acquired waveforms remain sufficiently accurate and representative for faithful plotting and analysis.

Another particularly interesting aspect arising from the use of MATLAB in model development is the possibility of directly comparing the obtained results with *Simscape*.

Simscape is a physical modeling environment integrated within Simulink that enables the development of multidomain physical models including electrical, electronic, mechanical, and thermal systems through a graphical *Model-Based Design* representation based on the components of the real system.

By leveraging advanced numerical techniques, Simscape allows a highly detailed description of complex systems, combining the physical plant model with the associated control algorithms.

Simscape is already widely adopted not only for system-level simulations but also for circuit-oriented design activities. This means that, instead of relying exclusively on experimental measurements performed on electro-mechanical systems intended for emulation, Simscape itself can be employed as a *reference system* for validating the firmware model developed within the HIL framework. [12]

This approach provides not only immediate access to the system behavior through simulation, but also the possibility of creating a realistic testbench environment that can be seamlessly integrated with the HIL firmware logic under development.

The canonical workflow typically adopted for *Hardware-in-the-Loop* projects based on the *Model-Based Design* approach is the following:

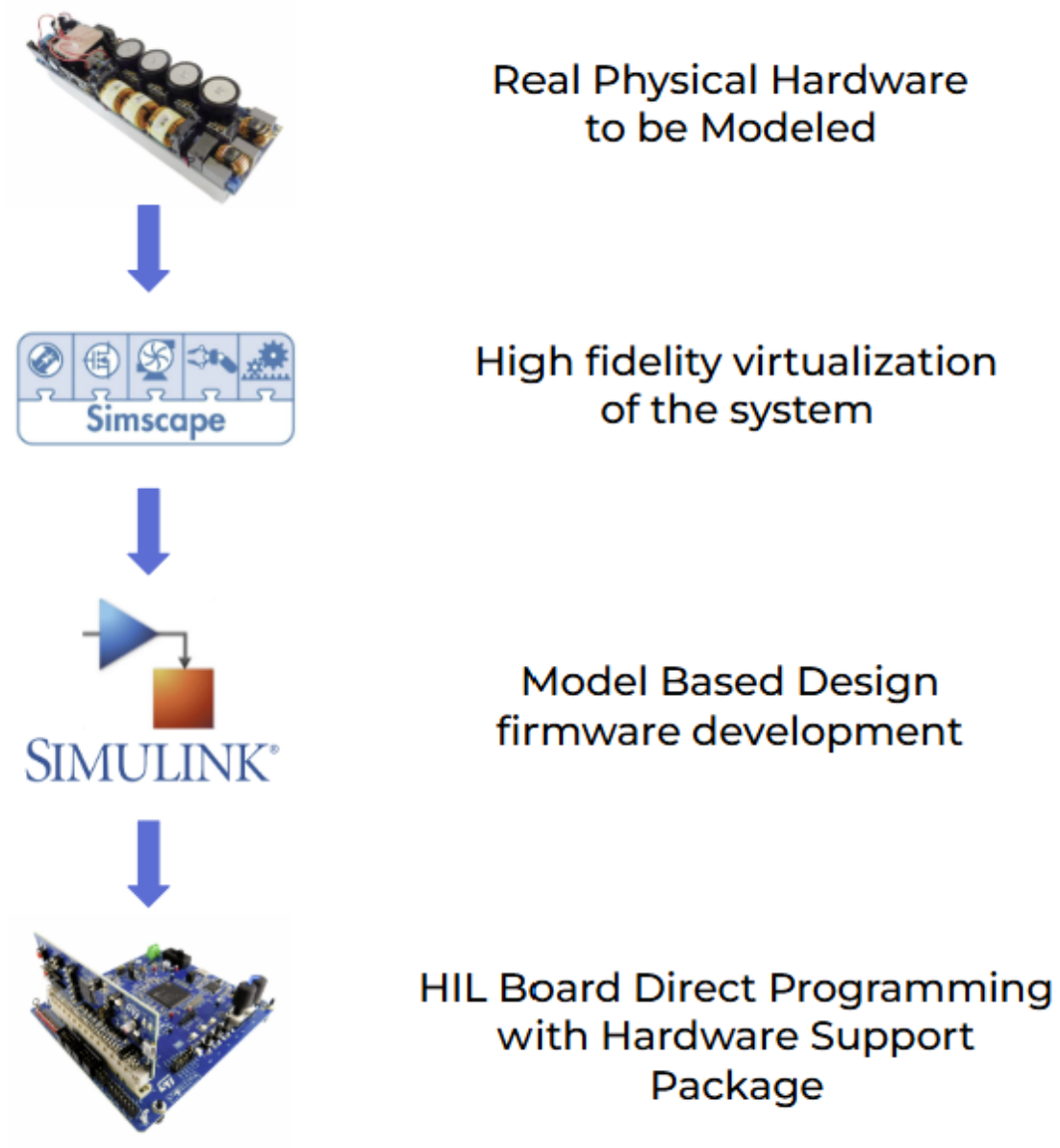

Figure 1.2: Flow HIL Design

1.3.3. Board technical details and HIL challenge

Given the possibility of performing detailed simulations through Simscape, which is one of the tools integrated within the MATLAB environment, a natural question arises:

> *"Why not directly use Simscape to develop the firmware model to be flashed onto the HIL board?"*

At first glance, it seems reasonable to assume that, through the HSP environment, all

the blocks available within Simulink, including those provided by Simscape, could be directly employed for firmware generation. Following this idea, one might simply consider deploying the Simscape model onto the HIL platform through the appropriate porting process, thereby obtaining the desired emulation system.

However, there exists a critical limiting factor that must be taken into account: the computational capability of the target board.

Indeed, when simulations are executed in Simulink, all computations are performed by the processor of the host computer running the simulation. The objective of Simscape is to provide accurate simulations through advanced numerical techniques, executing them as efficiently as possible, although still far from achieving true real-time operation.

By the way, this represents one of the key concepts in HIL emulation and arguably one of its greatest challenges, tightly coupled with the accuracy of the obtained results. The presence of highly detailed physical models inevitably leads to computational times that exceed the duration of the simulated real-time interval.

In practical terms, depending on the selected discretization step and on the complexity of the adopted model, simulating a transient lasting one second may require more than one second of actual computation time for the processor to solve all the underlying equations.

At the end of the simulation, Simscape is therefore capable of providing accurate results; however, these results become unsuitable for direct HIL deployment, except as a validation reference.

Returning to the previous controller motor example, the controller is designed to interact with a physical motor whose behavior is inherently analog. Consequently, all signals acquired by the controller from the motor are intrinsically continuous and effectively instantaneous from the perspective of the control system. When using Simscape, and similarly when operating an HIL board, the output signal must first be processed according to the provided inputs and only afterward generated at the output interface.

Achieving the exact same level of detail as a truly analog response is inherently impossible. Nevertheless, according to the *Shannon Sampling Theorem*, it is possible to determine whether a signal reconstructed from its discrete samples still retains the complete information content of the original analog waveform.

Many real-world signals are not perfectly band-limited, especially in the presence of abrupt transitions such as switching events. Their spectra may contain very high-order harmonic components, theoretically extending toward infinite frequency. However, in practical applications, beyond a certain frequency threshold the signal energy becomes negligible;

therefore, this frequency can reasonably be considered as the effective maximum frequency of interest.

By sampling the generated signal at a frequency at least twice as high as this maximum reference frequency, it is possible to obtain a sufficiently accurate simulation and/or emulation of the original analog behavior.

In Simscape, satisfying this constraint is relatively straightforward because the simulation is not bound by real-time execution requirements. The numerical solver can employ extremely small integration steps, thereby increasing the internal sampling frequency of the simulation, even at the expense of significantly longer execution times.

Instead, for a proper emulation, it is sufficient to ensure that all equations are solved within a time constraint dictated by twice the maximum frequency of the analog signal. By the way we must also take into account the limited computational capability of the target hardware.

The HIL board integrates a microcontroller SR5E1E770C30F01X belonging to the Stellar E1 family , selected for its high performance in real-time applications compared to competing solutions.

The device is based on a dual-core 32-bit Arm Cortex-M7 architecture, featuring a double-precision Floating Point Unit (FPU), L1 cache memory, and dedicated Digital Signal Processing (DSP) instructions aimed at efficient signal processing operations.

Thanks to this architecture, the microcontroller can operate at frequencies up to 300 MHz, ensuring high computational capability in real-time execution and low latency in critical operations [13].

According to the datasheet, optimal signal generation is guaranteed up to 100 kHz; however, it is essential to continuously verify that the computations performed by the model do not lead to CPU saturation. This happens because the management of all the structure is feasible only in many CPU operations to produce the desired output signals.

Considering that Simscape is capable of resolving fast transient phenomena in simulation at effective frequencies in the GHz range, it becomes evident that directly deploying a Simscape model onto the HIL board is not realistic.

For this reason, it is necessary to develop a dedicated reduced-order model that can emulate the behavior of the real system with a limited number of computational steps, while still preserving a good level of accuracy, even during fast transient conditions.

2 | Chapter two: Application

One of the main application fields of the *Hardware-in-the-Loop Model-Based Design* (MBD) approach is the emulation of power converters.

Regardless of whether the converter is AC–DC or DC–DC, every power conversion system generally follows the same fundamental structure: a *plant*, responsible for energy conversion and power delivery, and a *controller*, responsible for regulating the system behavior.

Due to the safety concerns associated with high-power operation and the rapid prototyping requirements discussed in Chapter 1 (Section 1.1), virtual testbench are frequently developed for such systems.

Through digital emulation, numerous advantages can be achieved, motivating STMicroelectronics to further investigate this field in order to expand its product portfolio, both for external customers and for internal prototyping activities.

One of the main projects carried out within our laboratory consists in the development of an online repository containing multiple power converter topologies freely accessible to users. The proposed platform provides:

- Selection of the plant topological parameters
- Functional analysis of the closed-loop topology
- Download of the customized HIL firmware for the plant
- Download of the control firmware
- Download of the graphical user interface for real-time signal monitoring

Using the reference kit *STEVAL-HILE1G4K1*, the HIL board can therefore be programmed with the plant model, while the second board executes the real control firmware. At the same time, the signals exchanged between the plant emulator and the controller can be monitored in real time through the dedicated graphical user interface (GUI). The official platform where the entire infrastructure is managed is the STMicroelectronics *Dig-*

ital Power Workbench, available at:
`https://eds.st.com/eds-console/#/digital-power-workbench?all_specs=false`

2.1. Totem-Pole Power-Factor-Converter Topology

In this thesis, the focus is placed on the study of a specific and widely used class of AC–DC converters, namely the Totem-Pole (TP) topology. The same considerations presented for this configuration can, in principle, be extended to any converter topology; however, in order to clarify the discussion and to gain a deeper understanding of the concepts underlying Hardware-in-the-Loop emulation, it is convenient to adopt this topology as a representative example.

First of all, it is necessary to examine its structure and operating principle.

The Totem-Pole AC–DC topology is a particular Power Factor Correction (PFC) converter architecture widely used in high-efficiency power supplies. The main idea behind this converter is to replace the conventional diode bridge rectifier with a structure composed primarily of active switches (MOSFETs or GaN FETs), in order to significantly reduce conduction losses. [14]

Due to its symmetric structure, the converter can be used both as a regulated DC voltage source from an AC input and, conversely, as an inverter when operated in the reverse direction.

As in any AC–DC converter, the key element governing voltage and current dynamics is the control of the line inductor. Thanks to the feedback control loop that actively drives the switches, it is possible to regulate the current flowing through the inductor and consequently set the desired output voltage.

To better understand its operation, we start from the schematic representation.

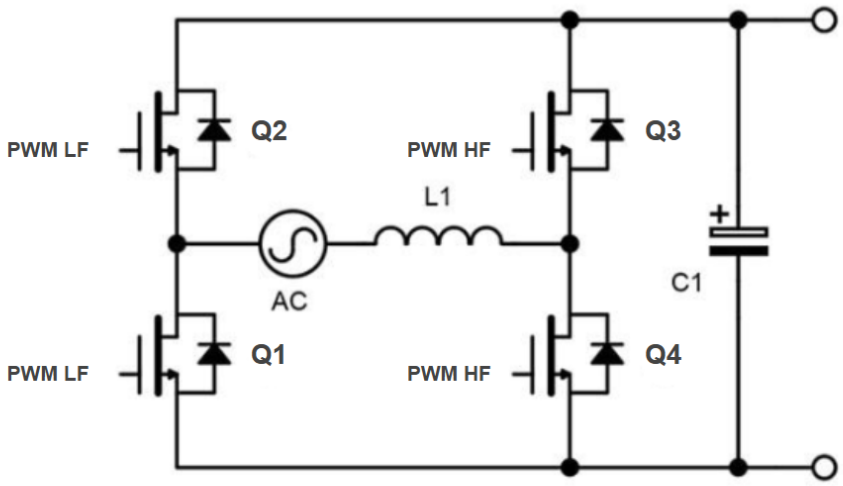

Figure 2.1: Totem Pole Schematic

Let us begin by considering the converter operating in AC–DC mode. The input consists of a single-phase sinusoidal voltage, which is applied to the input inductor L_1.

The power stage is composed of four power transistors, as highlighted by the presence of the body diode in the schematic. Intrinsically, within the structure of a MOSFET, there exists a parasitic PN junction between the body and the drain, which behaves as a diode. In power MOSFETs, this element is explicitly represented in the schematic because it is not merely a negligible parasitic effect, but rather a key component that is actively exploited during startup phases and for the recirculation of current spikes generated by fast switching transients.

Power MOSFETs therefore include an optimized intrinsic diode specifically designed for such applications.

The operating principle of this topology relies on the fact that the two left-side transistors are commutated synchronously with the AC source, effectively acting as a controlled rectifier, while the right-side leg operates at high switching frequency in order to generate the boost conversion effect.

A more detailed derivation of the governing equations will be presented in the following sections; however, to develop an intuitive understanding, the circuit can be represented through four distinct operating states. [15]

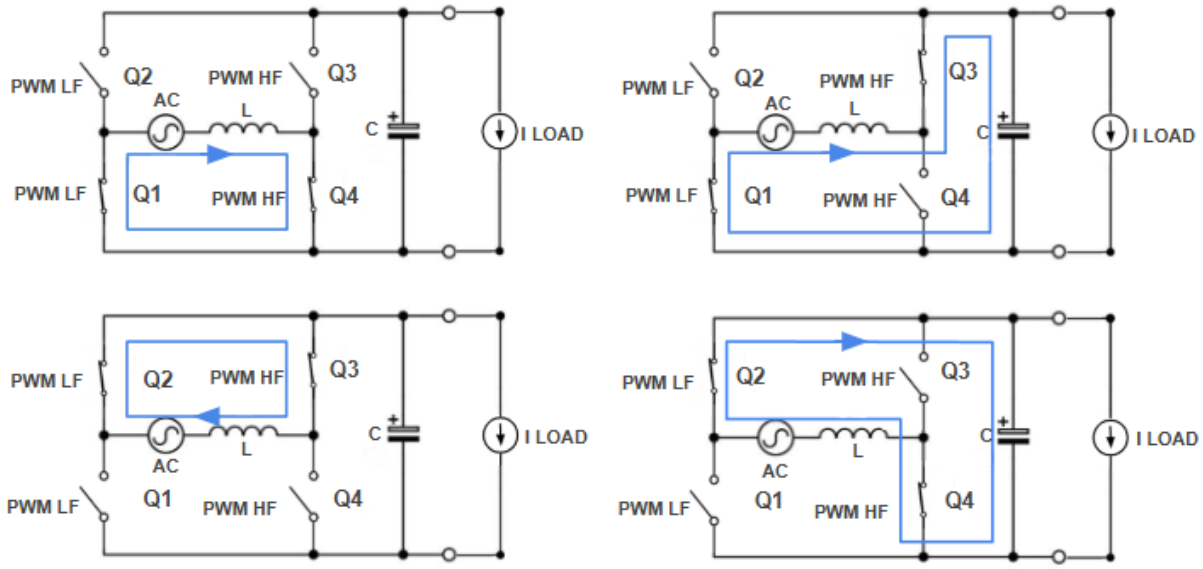

Figure 2.2: Model 4 Phases Totem Pole

The main idea behind this topology is to use the low-frequency switches as a form of controlled rectifier, while the high-frequency switches are responsible for the charge and discharge of the inductor in order to generate the classical boost conversion effect.

As shown in the Figure 2.2, the current direction is on the DC side always positive, in order to continuously supply the output capacitor, which therefore maintains a strictly positive voltage.

This topology is designed to convert an AC mains supply, acting as a power source, into a regulated DC output delivering power to the load. The load power demand determines the resulting input current waveform.

Since the objective is to transfer all available power to the load with minimal reactive power exchange, the converter is designed to achieve a power factor as close as possible to unity.

This factor determines the phase shift between the sinusoidal voltage waveform and the input current waveform. In this converter, which is directly connected to the electrical grid, the input voltage is sinusoidal at 50 Hz with an RMS amplitude of 230 V.

As the inductor is connected in series with the AC source, the current supplied from the grid is equivalent to the current flowing through the input inductor. So it must also be a sinusoidal waveform at 50 Hz for a complete power transfer.

The control system therefore drives the PWM signals applied to the MOSFETs according

to specific control laws in order to enforce this current shape.

Through active rectification, the output capacitor is able to charge even though the input current is sinusoidal with zero average value. The boost effect is instead obtained through the interaction between the inductor and the high-frequency switching action (in the order of tens of kHz) of the MOSFETs in the right leg.

During this analysis, the states of the two low-frequency MOSFETs and the high-frequency switching action are assumed to be constant over each period. The corresponding equivalent circuits are represented, respectively, in the two rows of Figure 2.2.

It follows that, during one phase, the inductor is charged by the input voltage source, while during the switching action of the high-frequency MOSFETs, the inductor releases its stored energy by injecting current into the output capacitor. Due to current continuity, this process results in an increase of the capacitor voltage, which can rise above the peak value of the grid voltage.

2.2. Circuit Equations

We now proceed with the analytical derivation of the equations describing this topology. First, it is important to outline the fundamental characteristics of the circuit, namely:

- a second-order system
- non-linear switching behavior

The second-order dynamics arise from the presence of two independent energy storage elements. The system therefore exhibits two independent modes of time evolution, governed by the differential equations associated with the capacitor and the inductor.

$$i_C(t) = C\frac{dv_C(t)}{dt} \tag{2.1}$$

$$v_L(t) = L\frac{di_L(t)}{dt} \tag{2.2}$$

The nonlinear behavior of the converter is introduced by the four MOSFETs operating as ideal switching devices. Their transitions between ON and OFF states continuously modify the equivalent circuit topology, thereby altering the current conduction paths.

The system can therefore be modeled as a piecewise-linear switching system, characterized by four distinct operating modes. Each mode is linear and time-invariant, while the overall

nonlinearity arises from the switching transitions between these modes.

As shown in Figure 2.2, the upper-left and lower-left operating phases both charge the inductor and, from an analytical point of view, are described by the same set of equations.

The other two phases, instead, differ in sign, consistently with the rectification behavior described previously. By carefully tracking the sign conventions and maintaining consistent reference directions for all electrical quantities across all operating modes, we obtain that

State Variables:

$$x = \begin{bmatrix} i_L \\ v_{dc} \end{bmatrix}$$

Phase 1 up left - PWM LF DOWN and HF DOWN = ON

$$\begin{cases} L\frac{di_L}{dt} = v_{in}(t) \\ C\frac{dv_c}{dt} = -i_{load} \end{cases}$$

Phase 2 up rigth - PWM LF DOWN and HF UP = ON

$$\begin{cases} L\frac{di_L}{dt} = v_{in}(t) - v_c \\ C\frac{dv_c}{dt} = i_L - i_{load} \end{cases}$$

Phase 3 down left - PWM LF UP and HF UP = ON

$$\begin{cases} L\frac{di_L}{dt} = v_{in}(t) \\ C\frac{dv_c}{dt} = -i_{load} \end{cases}$$

Phase 4 down rigth - PWM LF UP and HF DOWN = ON

$$\begin{cases} L\frac{di_L}{dt} = v_{in}(t) + v_c \\ C\frac{dv_c}{dt} = -i_L - i_{load} \end{cases}$$

We have therefore obtained a piecewise-defined system. The branch and node variables of the circuit are determined by the state variables, which, due to continuity, define the boundary conditions at the switching events.

Within each switching interval, however, the system is linear and time-invariant.

The classical analytical solution of such a system consists in solving the differential equations by integrating them within each operating region.

As a first step, it is important to understand the expected waveform during the steady-state operation of this topology. Naturally, during the initial phase there is a transient behavior, which will be described in detail in the following sections; however, a steady-state analysis already provides valuable insight into the fundamental behavior of the converter.

As previously discussed, during one phase the inductor stores energy, while during a subsequent phase it releases this energy by injecting current into the circuit due to continuity.

Two possible operating cases can therefore be identified: either the inductor completely discharges its stored energy within a switching cycle, or it does not.

These operating regimes are conventionally referred to as

- Discontinuous Conduction Mode (DCM)
- Continuous Conduction Mode (CCM)

The main difference between the various conduction regimes lies in the amount of energy stored in the inductor during the charging phase. If this energy is not sufficient to sustain current transfer over the entire switching period, the inductor fully discharges before the beginning of the next cycle, resulting in intervals where the current drops to zero, which is characteristic of Discontinuous Conduction Mode (DCM).

As the load current demand varies, and consequently the energy required during the discharge phase changes, the system may transition from Continuous Conduction Mode (CCM) to DCM, passing through the boundary condition known as Boundary Conduction Mode (BCM), in which the stored energy is exactly depleted at the end of each switching period.

It is worth noting that during the transition between CCM and BCM there is a variation in the average inductor current, whereas in DCM it is primarily the current slope, and thus the energy discharge rate, that determines the resulting waveform shape.

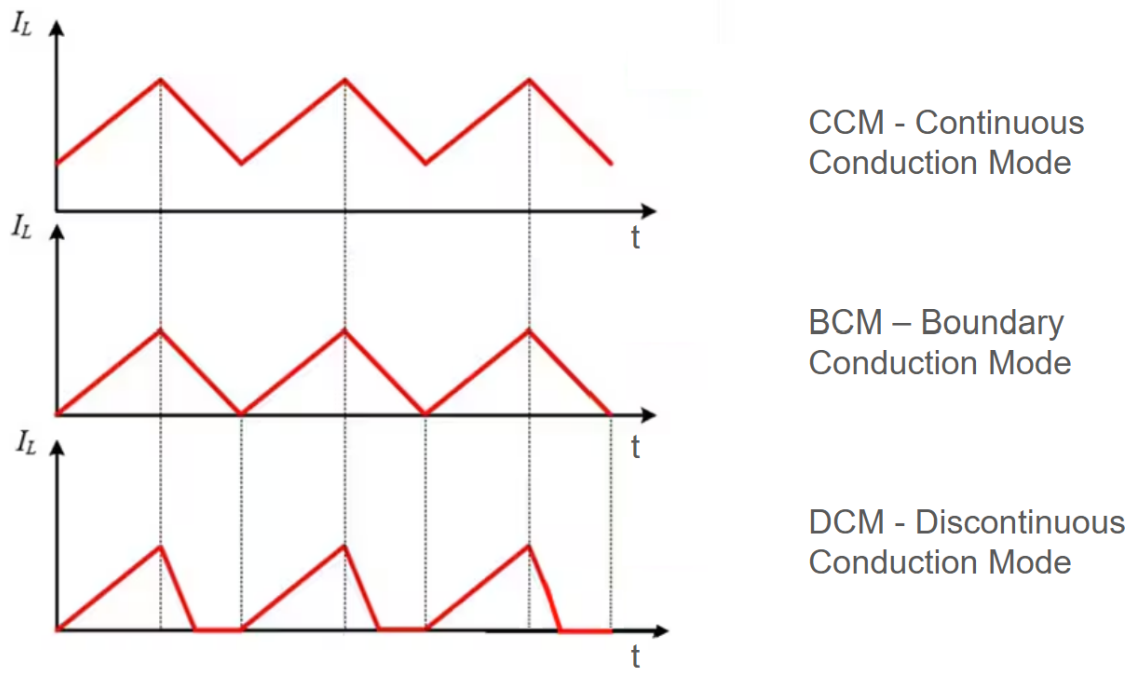

Figure 2.3: Operative regime Inductor at steady state

Although the Totem-Pole topology can operate in different conduction regimes, the analysis presented in this work is limited to CCM operation. This simplification is justified by the operating conditions and by the reference converter selected for the emulation, namely the STEVAL-7BIDIRCB, which is designed to operate predominantly in CCM.

In the following chapters, these data will be used to validate the proposed emulation, demonstrating the equivalence between the behavior of the physical Totem-Pole converter and its HIL-based implementation.

This practical possibility of validation has been a key factor in the selection of this specific topology, in addition to the company's research focus on this type of converter.

With this in mind, let us focus on the waveform of the topology operating in Continuous Conduction Mode and on the reason why we can assume a linear behavior within each switching phase, as shown in Figure 2.3.

Although the dynamics of inductors and capacitors are generally described by differential equations and may lead to exponential or nonlinear responses in many practical circuits, in CCM operation the inductor current is often approximated as linear within each switching interval. This is due to the fact that, during each sub-phase, the voltage applied across the inductor can be considered approximately constant, resulting in a linear variation

of the current if the parasitic resistance of the inductor is neglected (from the inductor relation, if $v(t)$ is constant, then $i(t)$ varies linearly).

This approximation becomes particularly accurate at high switching frequencies. Its validity will be further demonstrated in chapter 3 through a high-fidelity simulation of the topology performed in Simscape.

2.3. Original schematic and sensors

Let us now better understand which signals actually need to be emulated.

Ideally, all the electrical quantities of the circuit should be observed. However, since the available hardware resources are limited, it is important to identify the signals that are effectively measured by the sensors.

The schematic of this power converter is available at the following link:
`https://www.st.com/resource/en/data_brief/steval-7bidircb.pdf`

The converter is a complete system, consisting of both the plant that will be emulated and the control stage. [16]

To recall, the objective of this thesis is to:

1. develop a model of the Totem-Pole converter (plant) that can be performed at frequencies compatible with the working frequency of the HIL board ($\sim 100kHz$)
2. validate it through software simulations using Simscape;
3. implement it on the HIL hardware platform;
4. connect the HIL board to the control board mounted on the STEVAL-7BIDIRCB product;
5. verify through hardware testing that the controller responds to and regulates the HIL model as if it were interacting with the real converter;
6. verify that the regulated waveforms match those observed on the real power system.

As a first step, it is therefore necessary to gain a thorough understanding of the topology and identify the sensors used by the controller, both for input and output signals.

It is precisely through these sensors that the conversion between the high-power domain of the plant and the low-power domain of the controller takes place. As discussed previously, these sensors are an integral part of the HIL emulation.

Based on the schematic, it is possible to build a detailed Simscape simulation of the plant.

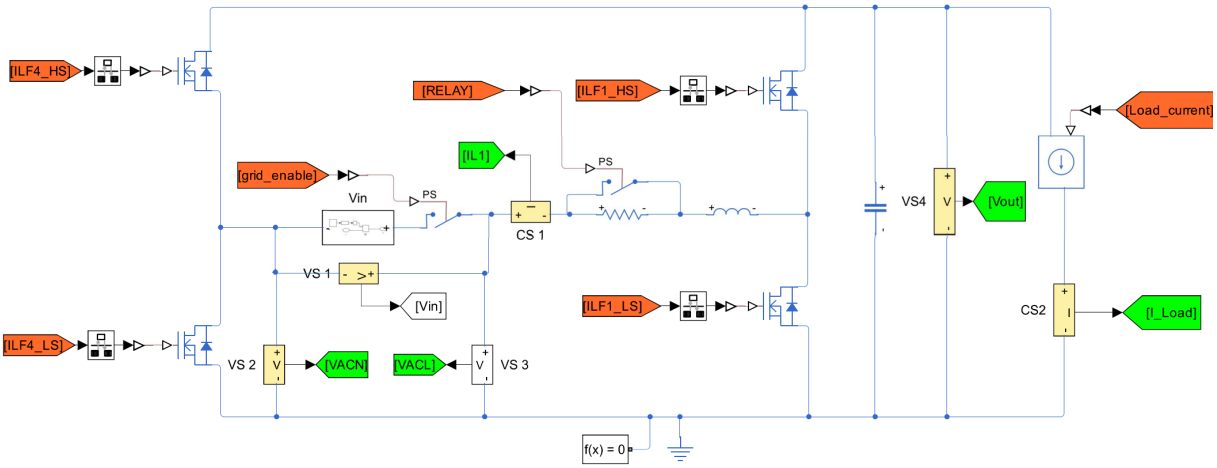

Figure 2.4: Simscape circuit model

The schematic describes all the signals that must be handled as inputs and emulated as outputs by our HIL board.

INPUT (Orange)

- `grid_enable`: grid enable signal
- `RELAY`: signal that closes the switch and bypasses the inrush resistor at the input
- `ILF4_HS`: low-frequency PWM signal for the high-side MOSFET
- `ILF4_LS`: low-frequency PWM signal for the low-side MOSFET
- `ILF1_HS`: high-frequency PWM signal for the high-side MOSFET
- `ILF1_LS`: high-frequency PWM signal for the low-side MOSFET
- `Load_current`: signal defining the load current

OUTPUT (Green)

- `VACN`: AC source voltage at the negative terminal with respect to ground
- `VACL`: AC source voltage at the positive terminal (towards inductor L) with respect to ground
- `IL1`: current flowing through the inductor
- `Vout`: output capacitor voltage (DC bus)
- `I_Load`: load current

With regard to the outputs, it is important to clarify a conceptual aspect. According to the original schematic of the *STEVAL-7BIDIRCB*, the control system does not directly measure the input voltage V_{in} as the voltage difference across the source terminals, but instead measures each terminal with respect to ground.

In fact, by computing the difference $VACL - VACN$, the input voltage V_{in} is reconstructed.

The only reason for this decomposition is that the negative node of the DC bus, which defines the circuit ground, is always topologically the lowest potential node in the system. Therefore, by measuring the voltages in this way, both $VACL$ and $VACN$ remain positive, which simplifies signal processing in the control system, as there is no need to introduce a bias in order to represent the negative half-cycle of the waveform.

Let us now focus on the effect of the sensors. The control system present in the *STEVAL-7BIDIRCB* measures the signals defined as outputs of the HIL board.

Each signal is, however, scaled in amplitude by a given gain, may be shifted by an offset when necessary, and is filtered by the sensor circuitry.

In this way, the output signals of the sensors are always constrained within the range 0 V to 3.3 V, ensuring full compatibility with the ADC inputs of the control unit.

Sensor	**Gain**	**Bias**	**Sensor Type**	**LP Filter f_c**
VACN	5.3 mV/V	0 V	Resistive divider	15.92 [kHz]
VACL	5.3 mV/V	0 V	Resistive divider	15.92 [kHz]
Vout	5.3 mV/V	0 V	Resistive divider	72.38 [Hz]
IL1	44 mV/A	1.67 V	ACHS-7123-000E$^{(*)}$	28.4 [kHz]
I_Load	66 mV/A	1.67 V	ACHS-7123-000E $^{(*)}$ and Resistive divider	1.59 [kHz]

$^{(*)}$ Isolated Hall-effect current sensor.

Table 2.1: Sensor gains and bias values from schematic

In order to emulate these sensors on the HIL board, it is necessary to reproduce the same gains. In this context, the Digital-to-Analog Converter (DAC) plays a central role. The DAC introduces a fixed gain determined by its operating range and resolution.

The objective is to satisfy the following relationship:

$$G_{\text{digital}} \cdot G_{\text{DAC}} = G_{\text{analog}} \tag{2.3}$$

Since the DACs mounted on the HIL board are 12-bit devices with a 0 V–3.3 V range, the

conversion gain is constant and given by:

$$G_{DAC} = \frac{3.3}{2^{12}} \approx 0.00080566 \text{ [V/count]} \tag{2.4}$$

From this, the digital gain G_{digital} of each sensor can be derived and used within the Model-Based Design implementation to convert the results obtained by solving the system of differential equations into DAC counts on the HIL board.

Through this scaling, the effect of the sensors is accurately emulated.

Sensor	Gain	Bias	Digital Gain	Digital Bias
VACN	5.3 mV/V	0 V	6.5785 [count/V]	0 [count]
VACL	5.3 mV/V	0 V	6.5785 [count/V]	0 [count]
Vout	5.3 mV/V	0 V	6.5785 [count/V]	0 [count]
IL1	44 mV/A	1.67 V	54.6 [count/A]	2048 [count]
I_Load	66 mV/A	1.67 V	81.9204 [count/A]	2048 [count]

Table 2.2: Digital Gains and Digital Bias

For the input sensors, i.e., those that receive low-power signals from the control system and convert them to high-power signals for the plant, no additional modeling effort is required.

This is because the signals generated by the controller are purely digital in nature, such as PWM or enable signals. The sensors placed between the control and the plant are only used to scale the voltage levels in order to drive MOSFET gates and enable relays.

In the model being developed, it is sufficient to detect when a logical transition occurs, rather than the exact voltage level at which it takes place.

Obviously, some transient effects introduced by these components will be neglected. However, in practice, by relying only on the logical state of the control signals, it is possible to determine the operating state of the system. By solving the corresponding equations, all relevant physical quantities required for the output sensors can then be computed.

2.4. Control Firmware

As discussed previously, our objective is to develop an emulation of the plant. For validation purposes, we aim to interface the plant emulation with the controller implemented in the ST product STEVAL-7BIDIRCB. This choice is motivated by the fact that the

associated firmware has already been documented, implemented, and validated. The firmware is publicly available through the official eDesign website, the online repository described previously. According to the datasheet, the code has been developed in C for the STM32G474RET6 microcontroller.

By opening the downloaded project in STM32CubeIDE, we can first analyze its pinout and, more importantly, understand the logic governing the relay actuation and PWM generation. This will allow us to develop a robust model that is also computationally efficient by adapting to the converter operating mode.

By inspecting the source code, it can be observed that the controller is structured as a state machine, which can be represented as follows:

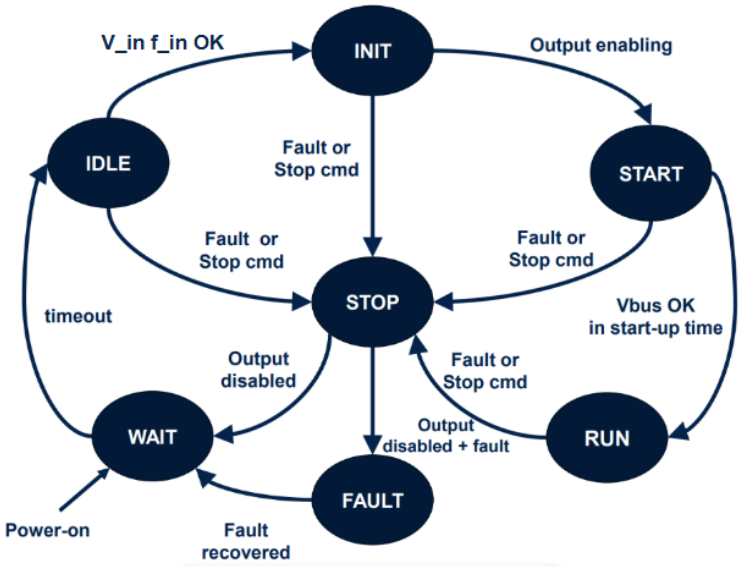

Figure 2.5: Firmware State Machine

State	Description
WAIT	Persistent state where the application is intended to stay, for a defined wait time, after fault conditions disappeared or at power-on. Following state is normally IDLE.
IDLE	Persistent state, following state can be INIT if all required conditions (input voltage and/or external command) are satisfied or STOP if there is a fault condition. Inside this state, the Inrush Current Limitation is performed, then it is checked if bus voltage reaches the 90% of mains peak voltage before going in INIT state.
INIT	"Pass-through" state, the code to be executed only once between IDLE and START states to initialize control variables and to enable outputs. Following state is normally START but it can also be STOP if there is a fault condition. The system can also stay in this state if the gate driver boot capacitor pre-charge, needed to supply HS MOSFET by gate driver, is enabled.
START	Persistent state where the converter's start-up is intended to be executed: Vbus is increased up to reference value. The following state is normally RUN as soon as start-up procedure is completed correctly. The following state is STOP if there is a fault condition.
RUN	Persistent state where converter is running normally. The following state is STOP if there is a fault condition or a stop command is received.
STOP	"Pass-through" state where PWMs are disabled. The following state is FAULT if a fault caused the stop of the converter, WAIT otherwise (external command received).
FAULT	Persistent state after a fault. Following state is normally WAIT as soon faults are cleared or recovered (not all faults can be recovered without a system reset)

Table 2.3: State machine description

Given this state machine, we can reconstruct the events that perturb our plant, represented in Figure 2.4, namely:

- **Inrush phase**: the grid relay controlled by `grid_enable` is activated, while the relay bypassing the line resistor remains open. The state machine transitions from `WAIT` to `INIT`.
- **Burst phase**: this phase is structured into two stages. First, the relay that bypasses the inrush resistor is closed. Then, after a timeout expires, a soft-start phase begins,

during which the PWM signals are activated in order to charge the output capacitor through short current pulses of constant amplitude until a predefined threshold is reached. The state machine operates in the `START` state.

- **PFC phase**: this is the phase in which the feedback loop between the Plant and the Controller is closed. The controller regulates the PWM signals driving the MOSFETs through a structure based on two Proportional-Integral (PI) controllers.[17] The PWM management implements the Power Factor Correction strategy described previously. The state machine operates in the `RUN` state.

3 | Chapter 3: Circuit Modeling

Let us now move to the core of our work, namely the development of a robust, discrete model that can be implemented on the HIL board.

It is important to keep in mind that the maximum computational rate of the HIL board is approximately 100 kHz, corresponding to a minimum sampling period of 10 μs.

With this constraint in mind, by analyzing the complete *STEVAL-7BIDIRCB* system, and in particular its control strategy, three operating phases can be identified. In chronological order from converter startup, these are:

1. Inrush phase
2. Burst phase
3. Power Factor Correction phase

Following this sequence, the next sections will describe and model each of these operating phases.

3.1. Inrush phase

When thinking about power converters, the primary focus is usually their operation at steady state. This condition is reached after a certain amount of time, which depends on the electrical quantities and parameters of the circuit, when the waveforms become periodic and repeat indefinitely.

At this point, all system variables have stabilized, reaching an equilibrium condition which, for the correct operation of a power converter, must be stable.

Since reaching this equilibrium requires time, the initial portion of the converter operation is known as the transient phase, during which the electrical quantities evolve before settling following an event that has perturbed the previously established equilibrium.

Proper management of this phase is fundamental, since it is often characterized by voltage and current peaks that may lead to component damage or even failure. A precharge phase

is therefore required, which in the modeled topology corresponds to a soft-start procedure for charging the output capacitor.

At startup, the DC bus capacitors are completely discharged. If the converter were to immediately enter steady-state regulation, it would attempt to bring the output voltage to its nominal value as quickly as possible, resulting in a very large inrush current. The soft-start phase is specifically introduced to avoid excessive inrush currents, component stress, and control instability in this transient.

During startup, when the input voltage V_{in} is connected to the converter, the closure of the relay causes an almost instantaneous change in the operating conditions. This transient can be particularly critical for LC circuits, which may generate large voltage and current peaks in the absence of a controlled voltage ramp.

It is sufficient to consider that an ideal infinite voltage derivative applied to a capacitor would theoretically result in an infinite current, regardless of the amplitude of the voltage step.

For this reason, a soft-start phase is also implemented in the Totem-Pole topology.

To understand its operation, let us consider the following figure.

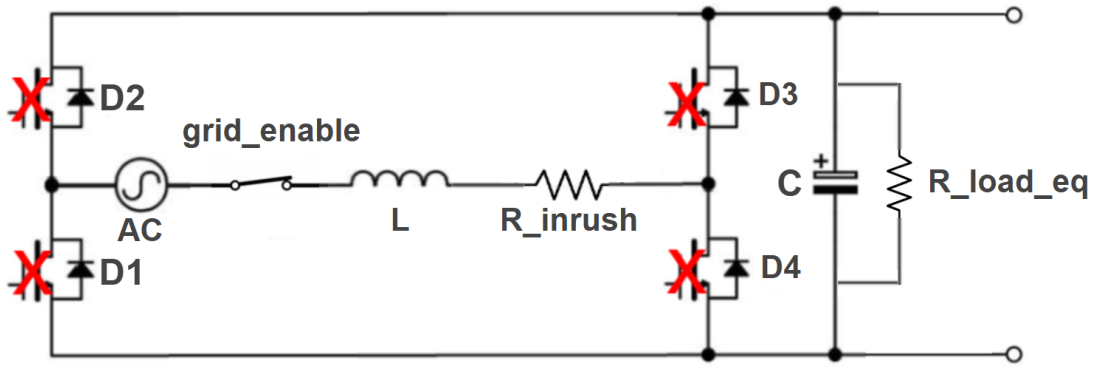

Figure 3.1: Inrush circuit model

We start from the initial state, where the relay controlled by `grid_enable` and `RELAY` is open. The capacitor C and the inductor L are assumed to be initially discharged.

As soon as the `grid_enable` relay is closed, the inrush phase begins. This phase consists of charging the DC bus capacitor through the inrush resistor.

During this stage, the current flows through the body diodes of the power MOSFETs, which effectively form a passive rectifier bridge. All MOSFET gates are kept in the OFF state.

Depending on the selected resistance value, the charging current is limited, and the DC bus capacitor exhibits the classical exponential charging behavior across its terminals.

Once the charging process is completed, approximately up to the peak value of the input sinusoidal voltage, the inrush resistor is bypassed by closing the relay controlled by `RELAY`, allowing full power transfer from the source to the load.

To understand how to model this behavior, we begin by analyzing the relevant quantities during this initial transient.

In this topology we have

- grid voltage: 230 V at 50 Hz
- electrolytic capacitors in the order of mF
- inductors in the order of hundreds of μH

With these characteristics, the response of the circuit to the 50 Hz input is inherently slow, due to the presence of large capacitor banks at the output and the current limitation imposed by the inrush resistor.

Due to the rectifier bridge effect, the system of equations describing the transient can be modeled as a piecewise linear system.

Topologically, three operating phases can be identified, based on Figure 3.2:

- D1 and D3 ON while D2 and D4 OFF
- D2 and D4 ON while D1 and D3 OFF
- all diodes OFF

The other combinations are not possible, since, due to the sinusoidal input voltage, whenever two diodes are forward-biased, the other two are necessarily reverse-biased.

Furthermore, it is not possible for only one of the two forward-biased diodes to conduct: once the voltage exceeds a certain threshold, both devices turn on simultaneously.

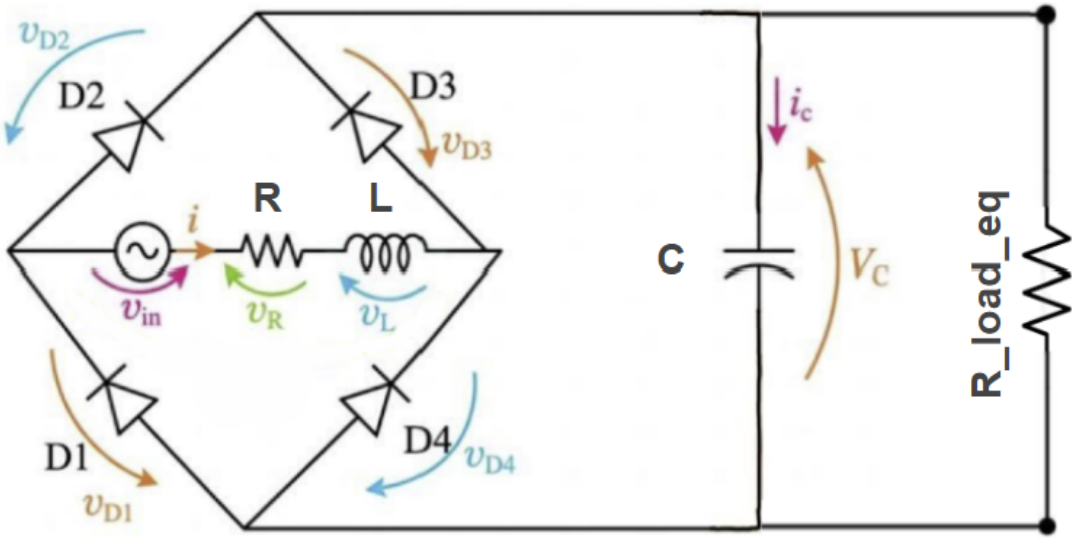

Figure 3.2: Inrush circuit model

Based on this scheme we can define the analytical equations in the three phases

D1-D3 ON D2-D4 OFF

$$\begin{cases} V_{in}(t) = R_{inrush} \cdot i_L(t) + L\,\dfrac{di_L(t)}{dt} + 2 \cdot V_{TH} + v_c(t) \\ C\,\dfrac{dv_c(t)}{dt} = i_L(t) - \dfrac{v_c(t)}{R_{eq\_load}} \end{cases}$$

D2-D4 ON D1-D3 OFF

$$\begin{cases} V_{in}(t) = R_{inrush} \cdot i_L(t) + L\,\dfrac{di_L(t)}{dt} - 2 \cdot V_{TH} - v_c(t) \\ i_L(t) = -\,C\,\dfrac{dv_c(t)}{dt} - \dfrac{v_c(t)}{R_{eq\_load}} \end{cases}$$

D1-D2-D3-D4 OFF

$$\begin{cases} i_L(t) = 0 \\ -\dfrac{v_c(t)}{R_{eq\_load}} = C\,\dfrac{dv_c(t)}{dt} \end{cases}$$

The system is a piecewise system where the state variables are

$$x = \begin{bmatrix} i_L \\ v_c \end{bmatrix}$$

It is now necessary to define a structure capable of determining when to switch between the different sets of equations, i.e., to precisely identify the instants at which the diodes turn on and off.

To achieve this, a simple state machine can be implemented.

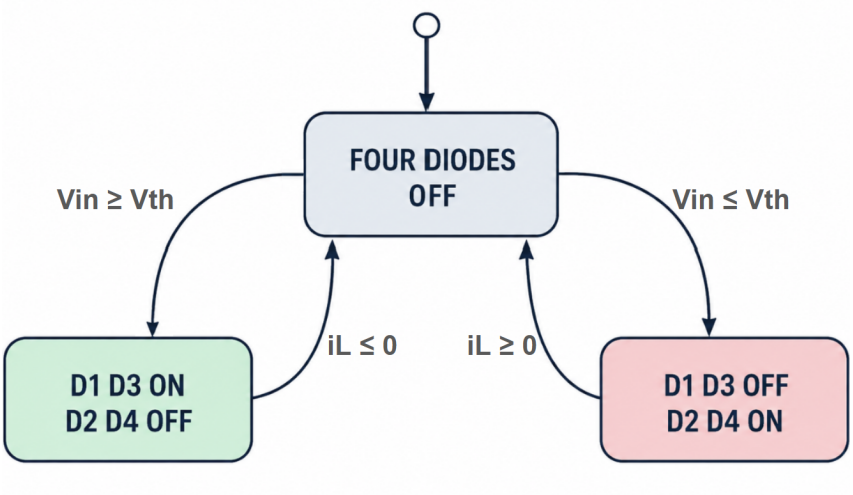

Figure 3.3: Diodes operation state machine

To better understand the reason behind these transitions, it is sufficient to observe that:

- if two diodes are conducting, there will always be an intermediate state in which all four diodes are turned off before the other pair turns on;
- if all four diodes are off, no current flows in the circuit; therefore, turn-on is determined solely by V_{in} with respect to the threshold V_{th}, defined as the sum of the forward thresholds of the two diodes that are going to conduct added to the capacitor voltage;
- for the turn-off condition, even if the voltage changes, the inductor continues to force current in the same direction due to continuity. It would therefore be incorrect to evaluate the voltage drop with respect to a threshold. When the current reaches zero or changes sign, it means that the operating regime has ended and the system transitions to the OFF state.

Through this simple state machine, we are always able to track the operation of the rectifier bridge, and therefore, at each instant, we know which set of equations must be solved.

At this point, we only need to solve the differential equations case by case, while enforcing the continuity of the state variables at each switching event. However, this naturally raises

the following question:

"Which technique do we use to solve a system of differential equations on a microcontroller?"

This question must be addressed carefully, since the computation is performed in discrete time by a microcontroller, and therefore requires a discussion within the framework of numerical methods.

3.1.1. Euler methods for numerical computation

Euler's method represents one of the earliest and most important algorithms developed for the numerical solution of ordinary differential equations. Introduced by Leonhard Euler in the 18th century, it constitutes the theoretical starting point for most modern numerical techniques used in the simulation of dynamic systems [18].

The objective of the method is to determine an approximate solution of an initial value problem described by a first-order differential equation. In many engineering applications, an analytical closed-form solution is not available, or the computational cost required to obtain it is excessively high. It therefore becomes necessary to rely on numerical procedures that approximate the evolution of the solution through a sequence of discrete steps.

The basic idea behind Euler's method is to exploit the information provided by the derivative of the function. Since the derivative represents the rate of change of the considered quantity, it can be used to predict the value of the solution at the immediately subsequent time instant.

Starting from a known initial condition, the method assumes that, within a sufficiently small time interval, the solution can be approximated by the tangent line to the curve at the current point.

From a geometric point of view, the procedure is equivalent to locally replacing the actual curve with a sequence of straight line segments. At each iteration, the slope of the solution is computed at the current point, and this slope is used to estimate the value of the function at the next time step. By repeating this process over the entire integration interval, an approximate trajectory of the solution is obtained.

The accuracy of the method strongly depends on the size of the integration step h. Smaller values of h generally lead to more accurate results, since the linear approximation remains valid over shorter intervals. However, reducing the step size increases the number of

required iterations and therefore the computational cost.

This results in the classical trade-off between numerical accuracy and computational efficiency that characterizes most numerical integration methods.

Forward Euler

The Forward Euler method is one of the simplest numerical methods used to approximate the solution of an ordinary differential equation (ODE).

The method assumes that, over a small interval h, the solution follows the slope of the tangent line at the current point:

$$y(t_n + h) \approx y(t_n) + h\, y'(t_n)$$

Applied iteratively, this relation allows the construction of the entire function $y(t)$.

The key assumption is that, within each interval of width h, the function does not vary significantly and can therefore be approximated by the tangent line at the initial point of the interval.

Iteratively, the error accumulates from step to step. To study stability and ensure that the numerical solution does not diverge, the so-called test equation is typically used, since it represents the simplest linear differential equation and, by assumption, can locally describe even complex systems.

$$y' = \lambda y$$

where λ is a constant. This test does not exactly describe the derivative of a real function, but it allows a straightforward assessment of numerical stability.

We now apply the Forward Euler method using the test equation, obtaining:

$$y_{n+1} = (1 + h\lambda)\, y_n$$

The exact solution of the test differential equation is:

$$y(t) = y_0 e^{\lambda t}$$

which is exactly the function whose derivative is itself multiplied by a constant.

Therefore, numerical stability is guaranteed only if:

$$|1 + h\lambda| < 1$$

As you can see, λ depends on the function, but h is our chosen step. For large steps, i.e., h, there is a risk of instability.

Backward Euler

The difference from Forward Euler lies in the construction of the derivative. In this method, we evaluate the derivative at the next time step, so the iterative structure is:

$$y(t_n + h) \approx y(t_n) + h\, y'(t_n + h)$$

We also apply the test using Euler's formulation and the test equation, obtaining:

$$y_{n+1} = y_n + h(\lambda y_{n+1})$$

which, after collecting terms, becomes:

$$y_{n+1} = \frac{1}{1 - h\lambda}\, y_n$$

For all physically stable problems (i.e., where the real part of $\mathrm{Re}(\lambda) \leq 0$), the method is always stable for any value of $h > 0$, therefore:

- it never diverges
- it never oscillates
- even with very large steps, the system remains damped

Therefore, despite higher complexity and a greater computational cost compared to Forward Euler, we obtain a method that is always stable. Obviously if we use too large steps to describe fast dynamics it will not be accurate but in any case it will be stable.

3.1.2. Numerical Equations for Inrush phase

Given its intrinsic stability, we will adopt the Backward Euler method, with step size h, to solve the piecewise system of the inrush phase.

We now rewrite the continuous equations in discrete form, evaluating them using the

derivative at the next time step, consistently with the formulation of the method.

D1-D3 ON D2-D4 OFF

$$\begin{cases} V_{in,n+1} = Ri_{L,n+1} + L\dfrac{i_{L,n+1} - i_{L,n}}{h} + 2V_{TH} + v_{c,n+1} \\ C\dfrac{v_{c,n+1} - v_{c,n}}{h} = i_{L,n+1} - \dfrac{v_{c,n+1}}{R_{eq\_load}} \end{cases}$$

D2-D4 ON D1-D3 OFF

$$\begin{cases} V_{in,n+1} = Ri_{L,n+1} + L\dfrac{i_{L,n+1} - i_{L,n}}{h} - 2V_{TH} - v_{c,n+1} \\ i_{L,n+1} = -C\dfrac{v_{c,n+1} - v_{c,n}}{h} - \dfrac{v_{c,n+1}}{R_{eq\_load}} \end{cases}$$

D1-D2-D3-D4 OFF

$$\begin{cases} i_{L,n+1} = 0 \\ -\dfrac{v_{c,n+1}}{R_{eq\_load}} = C\dfrac{v_{c,n+1} - v_{c,n}}{h} \end{cases}$$

Rearranging the equations and making the system state variables explicit, we obtain:

D1-D3 ON D2-D4 OFF

$$\begin{cases} \left(R + \dfrac{L}{h}\right) i_{L,n+1} + v_{c,n+1} = V_{in,n+1} + \dfrac{L}{h} i_{L,n} - 2V_{TH} \\ -i_{L,n+1} + \left(\dfrac{C}{h} + \dfrac{1}{R_{eq\_load}}\right) v_{c,n+1} = \dfrac{C}{h} v_{c,n} \end{cases}$$

D2-D4 ON D1-D3 OFF

$$\begin{cases} \left(R + \dfrac{L}{h}\right) i_{L,n+1} - v_{c,n+1} = V_{in,n+1} + \dfrac{L}{h} i_{L,n} + 2V_{TH} \\ i_{L,n+1} + \left(\dfrac{C}{h} + \dfrac{1}{R_{eq\_load}}\right) v_{c,n+1} = \dfrac{C}{h} v_{c,n} \end{cases}$$

D1-D2-D3-D4 OFF

$$\begin{cases} i_{L,n+1} = 0 \\ \left(\dfrac{C}{h} + \dfrac{1}{R_{eq\_load}}\right) v_{c,n+1} = \dfrac{C}{h} v_{c,n} \end{cases}$$

By solving the system, we obtain the new values of the state variables and, step by step, build their waveforms. In the next chapter, this set of equations will be validated respect with Simscape and the implementation on hardware using the HIL board with a h step of $\frac{1}{65kHz} \sim 15us$ consistent in timing for the board will be discussed.

For the moment, we proceed to mathematically model the second phase of our transient.

3.2. Burst phase

The burst phase completes the soft-start process initiated during the inrush phase. Its purpose is to bring the DC bus to its nominal operating condition by fully charging the output capacitor. During this stage, the load is not connected, and almost all the power drawn from the source is transferred to the capacitor, except for the losses mainly caused by the parasitic resistances of the inductor, the capacitor, and the equivalent on-state resistances of the MOSFETs.

This phase is characterized by two distinct events that must be handled separately in order to implement an efficient model on a microcontroller.

- closure of the inrush relay
- activation of the PWM signals in burst mode

As far as the first part is concerned, we can model the new operating state as follows:

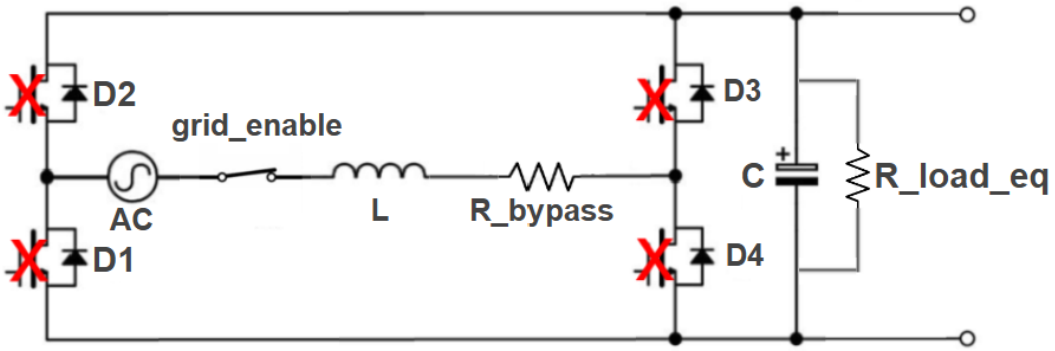

Figure 3.4: Diodes operation state machine

Therefore, by simply changing the value of the resistance with respect to the inrush phase, we are able to model both the closure of the switch and the effect of the parasitic elements. In fact, the bypass resistance is an equivalent resistance computed as the sum of the inductor series resistance and the internal resistance of the MOSFET body diode.

From the model point of view, this event does not introduce significant changes. We simply continue to execute iteratively the model developed for the inrush phase, updating the value of the resistance accordingly. Following the temporal sequence imposed by the controller, after a certain delay the PWM signals are activated in order to charge the output capacitor through short current pulses of constant amplitude until a predefined voltage threshold is reached. In principle, when developing a plant model, the internal operation of the controller should not be relevant, since the behavior of the plant is completely determined by its topology.

"Why, then, do we not use a single unified model instead of separating the operation into different phases?"

It is important to keep in mind that the maximum computation rate of the HIL board is approximately 100 kHz, while the switching frequency of the STEVAL-7BIDIRCB is 65 kHz, according to the datasheet. During the first phase, as previously discussed, the plant evolves naturally with relatively slow dynamics. However, once the PWM signals are activated, they introduce a fast dynamic that becomes comparable to the execution speed of the microcontroller running our model.

In fact, during the first stage we did not model the MOSFET itself, but only its associated body diode. Moreover, we only considered its threshold voltage, neglecting its nonlinear behavior. This approximation is perfectly adequate for modeling the initial transient, but once the PWM signals become active, these assumptions are no longer valid. A new model must therefore be developed to account for the new current paths created by the switching of the MOSFETs.

Consequently, it becomes unavoidable to design a new model capable of accurately reproducing the fast transient behavior imposed by the 65 kHz switching frequency.

This represents the main challenge and hardware limitation of the project, and it is precisely the solution proposed in the next section that gives value to the work presented in this thesis.

Since the model used for the burst phase and for the Power Factor Corrector phase is exactly the same, in the next section we will describe the new model and the technique adopted to represent the high-frequency transient behavior.

3.3. Power Factor Corrector (PFC) phase

Let us now focus on the core challenge of this work, namely:

"How can a microcontroller capable of executing a complete computation cycle at approximately 100 kHz emulate dynamics characterized by switching frequencies of 65 kHz?"

In this phase, the high-frequency PWM signals impose a switching dynamic with a period of approximately 15 μs, which is compatible with the computational capabilities of our board. However, it should be noted that, according to the sampling theorem, a signal should be sampled at least at twice its maximum frequency component.

An ideal PWM signal exhibits infinitely steep rising and falling edges, meaning that an infinitely high sampling frequency would theoretically be required to represent it exactly in a discrete-time framework. In practice, however, we can reasonably assume that dividing the PWM period into approximately 100 intervals provides a sufficiently accurate representation of the switching transients.

Therefore, by carrying out the calculation, we obtain:

$$Switching\ period = \frac{1}{65\,000\ Hz} = 1.5385e-05 \sim 15.38\,us$$

which by discretizing becomes

$$T_{model} = \frac{Switching\ period}{100} \sim 153.8\,ns$$

with a corresponding sampling frequency of approximately $\sim 6.5\,\text{MHz}$.

Such a frequency is clearly not achievable on a microcontroller. It therefore becomes necessary to adopt a mathematical technique capable of describing the high-frequency transient behavior even when using a relatively slow discrete-time sampling rate.

The solution adopted in this work is the *Average Model*.

3.3.1. Average Model

The *average model* is a modeling technique that enables the description of switching power converters by replacing the time-varying dynamics introduced by the switching actions with an equivalent averaged behavior over a switching period.

In this approach, the switching dynamics of the system are substituted by an equivalent continuous-time representation obtained as the weighted average of the state derivatives over the different switching intervals. [19] This formulation relies on the assumption that the state variables evolve slowly with respect to the switching period, allowing their high-

frequency ripple components to be neglected while preserving the dominant low-frequency dynamics of the converter.

$$\frac{dx(t)}{dt} \approx \left\langle \frac{dx(t)}{dt} \right\rangle = duty(t) \left(\frac{dx(t)}{dt} \right)_{ON} + \big(1 - duty(t)\big) \left(\frac{dx(t)}{dt} \right)_{OFF}$$

From a computational point of view, once the derivative is obtained and the switching period is known, the state variables can be updated simply by applying discrete integration via Euler Forward.

$$x(t + T_{\text{switching}}) = x(t) + T_{\text{switching}} \left(\frac{dx(t)}{dt} \right)_{Average}$$

This approach, based on a duty-cycle-weighted average of the ON and OFF configurations, makes it possible to obtain a continuous-time model, significantly simplifying both the analysis and the design of the system.

Our objective remains the development of a faithful plant emulation through the HIL platform, and the signal generation process will be based precisely on the average model. This naturally leads to the following question:

"Why do we use the Average Model?"

As discussed in subsection 1.3.3 of Chapter 1, the HIL board used in this work is inherently constrained by its available computational resources and processing speed.

Consequently, the high-frequency dynamics associated with the switching action of the high-frequency MOSFETs cannot be directly implemented on the HIL platform. For this reason, the average model becomes a fundamental tool. It provides a mathematical framework capable of capturing the overall behavior of the converter while avoiding the need to explicitly reproduce every switching event.

By implementing the averaged equations on the HIL, it becomes possible to emulate the dominant converter dynamics and accurately reproduce its macroscopic behavior. In this way, the effects of switching phenomena occurring at frequencies much higher than the available computation rate can still be represented through their averaged contribution to the system dynamics.

Furthermore, the reason for adopting the Average Model lies in the remarkable accuracy with which it is able to describe this type of transient behavior. As discussed in Section 2.2 of Chapter 2, the steady-state operating condition considered in this work is the Continuous Conduction Mode (CCM), which can be approximated, to a first order, as a sequence of linear transients during the high-frequency switching intervals.

The Average Model naturally fits into this framework. The average state derivative is computed as the weighted sum of the derivatives associated with the different PWM switching states, where the weighting factors are determined by the fraction of the switching period spent in each state, that is, by the duty cycle. Equivalently, after dividing by the total switching period, the average derivative can be expressed simply as the sum of the state derivatives multiplied by their respective duty-cycle coefficients.

The key point to emphasize is that, although this approach does not explicitly track the rising and falling portions of the high-frequency transient, the initial and final values of the state variables over each switching period are mathematically preserved. In other words, the switching ripple within the period is neglected, while the overall evolution of the system is accurately reproduced.

A particularly illustrative example is the inductor current waveform, shown below without a specific scale for conceptual purposes in order to highlight the principle behind the Average Model. This waveform will be presented and discussed in greater detail in the final chapter.

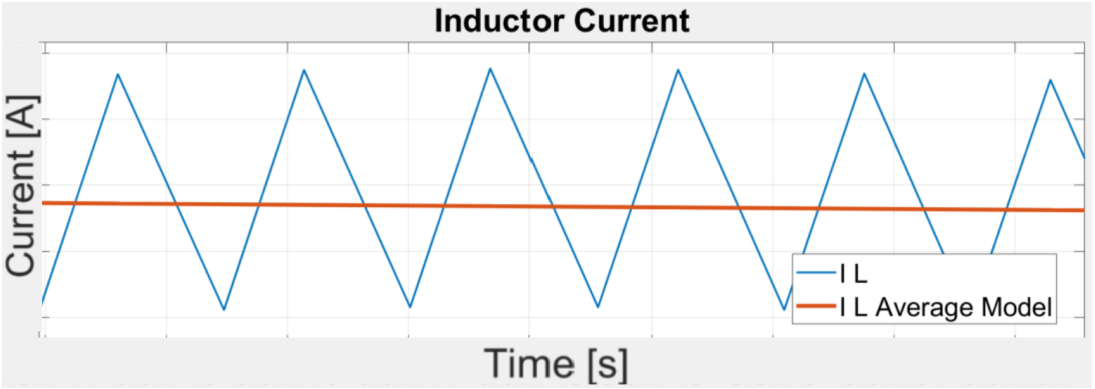

Figure 3.5: Average Model VS CCM inductor current waveform

The waveform shown in purple represents the Average Model, which accurately captures the average behavior of the orange waveform corresponding to the high-frequency switching ripple.

3.3.2. Numerical Equations for the PFC Phase

In Section 2.2 of Chapter 2, the system was modeled as a piecewise linear system. The conventional approach typically adopted in FPGA-based emulation platforms consists of numerically solving these equations by applying the Backward Euler method with a time step on the order of nanoseconds. The procedure is straightforward: the differential

equations are rewritten in discrete form, the appropriate set of equations is selected according to the current operating state, and continuity between different operating modes is ensured through the state variables.

As previously discussed, such an approach is not feasible in our case. Therefore, the first step is to apply the Average Model technique over the switching period, allowing the use of a sampling frequency compatible with the computational capabilities of the microcontroller.

The required steps are:

- identify the operating phases that compose each switching period;
- derive the state-variable derivatives for each phase;
- compute their average over the switching period according to the duty cycle;
- integrate the average derivative using the Forward Euler method.

Although the Backward Euler method offers superior numerical stability properties, as discussed in the previous section, the Forward Euler method is employed in this work to minimize the computational cost associated with the real-time implementation on the microcontroller. The Average Model already incurs a non-negligible computational effort and directly provides the system state derivatives. Therefore, the remaining task consists solely of integrating these derivatives, for which the Forward Euler scheme represents the simplest and most computationally efficient solution.

Furthermore, despite the fact that Forward Euler is only conditionally stable, the risk of numerical divergence is mitigated in the present application. The derivatives are obtained from the Average Model, which provides smooth averaged dynamics by construction. As a result, the state derivatives remain bounded within the operating conditions considered, making the use of Forward Euler suitable for the intended implementation. Recalling the equations derived in Section 2.2 and isolating the state derivatives for each operating phase, we can construct the following table:

In principle, we know that the converter has four possible operating states. However, within a switching period, only the transistors driven by the high-frequency PWM signals actually switch. Therefore, the system can be represented in the following way by considering the PWM sequence.

Phase 1 – Q1 Q4 ON	**Phase 2 – Q1 Q3 ON**
$\frac{di_L}{dt} = \frac{v_{in}(t)}{L}$ $\frac{dv_c}{dt} = -\frac{i_{load}}{C}$	$\frac{di_L}{dt} = \frac{v_{in}(t) - v_c}{L}$ $\frac{dv_c}{dt} = \frac{i_L - i_{load}}{C}$
Phase 4 – Q2 Q3 ON	**Phase 3 – Q2 Q4 ON**
$\frac{di_L}{dt} = \frac{v_{in}(t)}{L}$ $\frac{dv_c}{dt} = -\frac{i_{load}}{C}$	$\frac{di_L}{dt} = \frac{v_{in}(t) + v_c}{L}$ $\frac{dv_c}{dt} = \frac{-i_L - i_{load}}{C}$

Table 3.1: State derivatives in the four switching phases

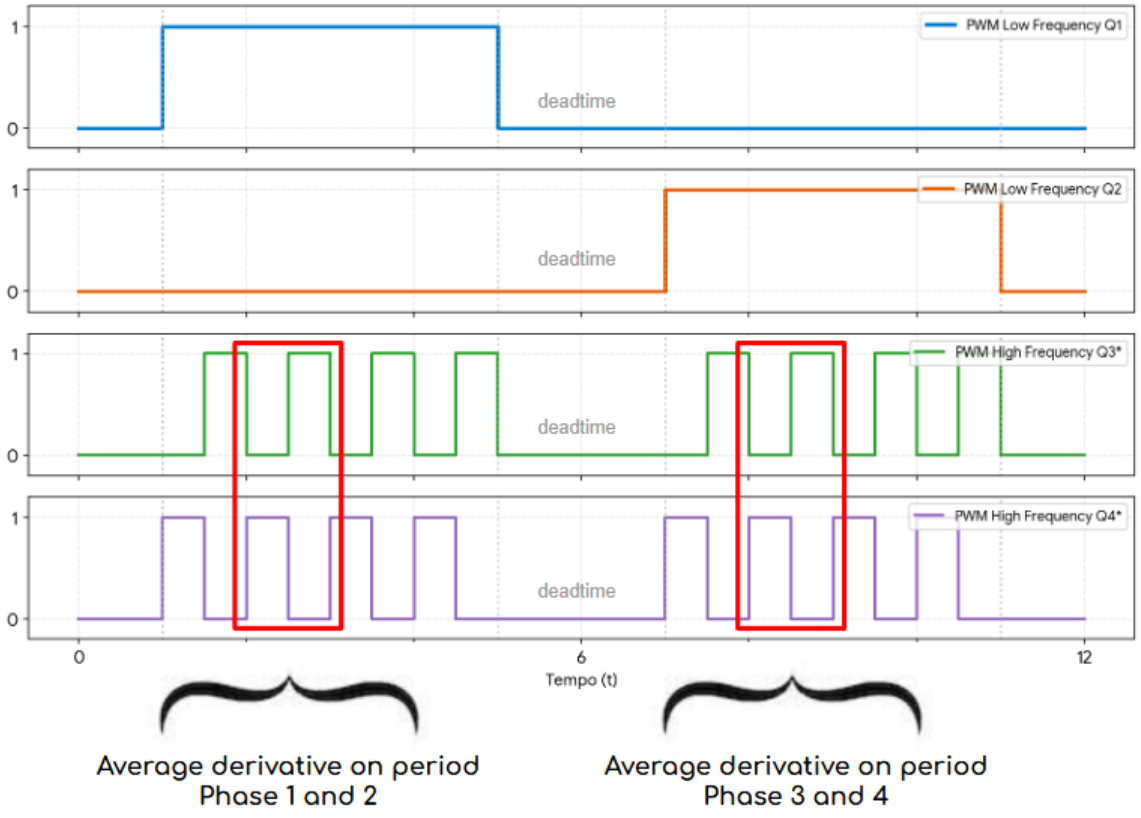

Figure 3.6: Qualitative PWM in PFC mode

As a preliminary remark, it is important to emphasize that the PWM signals are not shown to scale. In fact, if the low-frequency PWM signals with a period corresponding to the 50 Hz mains frequency were represented together with the 65 kHz PWM signals,

the latter would not be graphically visible in the figure. Their switching events would be so densely packed that they would simply appear as a solid block at this zoom level.

Therefore, solely for the purpose of illustrating the structure of the Average Model, the high-frequency switching is represented by only a few commutations. Moving from one switching period to the next, three possible operating conditions can be identified:

- Q1 ON with fast switching of Q3–Q4;
- Q2 ON with fast switching of Q3–Q4;
- **DEADTIME**: all PWM signals are OFF.

Consequently, our model determines the operating condition according to the logical state of the two low-frequency PWM signals and then applies the corresponding set of Average Model equations.

The two average derivatives for the inductor current are

Phase 1 - 2 - Q1 ON

$$\left\langle \frac{di_L}{dt} \right\rangle = \frac{v_{in}(t)}{L}\, duty_{Q4}(t) + \frac{v_{in}(t) - v_c}{L}\, duty_{Q3}(t)$$

Phase 3 - 4 - Q2 ON

$$\left\langle \frac{di_L}{dt} \right\rangle = \frac{v_{in}(t)}{L}\, duty_{Q4}(t) + \frac{v_{in}(t) + v_c}{L}\, duty_{Q3}(t)$$

While for the derivative of the capacitor voltage:

Phase 1 - 2 - Q1 ON

$$\left\langle \frac{dv_c}{dt} \right\rangle = -\frac{i_{load}}{C}\, duty_{Q4}(t) + \frac{i_L - i_{load}}{C}\, duty_{Q3}(t)$$

Phase 3 - 4 - Q2 ON

$$\left\langle \frac{dv_c}{dt} \right\rangle = -\frac{i_{load}}{C}\, duty_{Q4}(t) + \frac{-i_L - i_{load}}{C}\, duty_{Q3}(t)$$

Once the average state derivatives have been defined, they can be integrated using the Forward Euler method with a time step equal to the switching period, $T_s = \text{switching\_period}$.

By iteratively applying this procedure, the current and voltage waveforms can be reconstructed.

$$i_L(k+1) = i_L(k) + T_s \left\langle \frac{di_L}{dt} \right\rangle_k$$

$$v_c(k+1) = v_c(k) + T_s \left\langle \frac{dv_c}{dt} \right\rangle_k$$

As can be observed from Figure 3.6, a third operating region must also be taken into account, namely the dead time between the low-frequency PWM signals.

Since the low-frequency PWM operates at 50 Hz, corresponding to a period of 20 ms, the dead time is not negligible with respect to the high-frequency PWM dynamics, having a duration of approximately 500 μs.

During this interval, a sufficiently large delay is intentionally introduced between the turn-off of one MOSFET and the turn-on of the other in order to avoid extremely dangerous short-circuit conditions. The main risk is that, due to a glitch or timing mismatch, both MOSFETs could become active simultaneously, effectively short-circuiting the entire DC bus capacitor and generating extremely high current peaks.

For this reason, the dead-time interval must also be included in the model.

During this period, the inductor current cannot be interrupted because of the continuity property of the inductor. Instead, it flows through an alternative path provided by the body diodes of the MOSFETs. In practice, for a short interval, the converter operates in diode conduction mode until the next MOSFET is turned on.

Since the recirculation structure through the body diodes is exactly the same as the one used in the inrush modeling described in Section 3.1, the dead-time model in PFC mode is implemented using the same model as the inrush phase.

We now have all the elements required to construct the complete plant model, which will be validated both at software level using Simscape and at hardware level with the HIL board in the next chapter.

4 | Chapter Four: Validation

In the previous chapter, we developed a robust and computationally efficient model for the implementation of the emulation on a microcontroller. However, its actual performance must now be verified through a concrete validation process.

The proposed validation is carried out on two different levels:

- Software validation;
- Hardware validation.

In this chapter, we first compare the complete model, including the inrush phase, burst phase, and PFC phase, against a high-fidelity simulation developed in Simscape. After verifying the correctness of the model at the software level, the same logic is deployed onto the target microcontroller, and the generated waveforms are measured using an oscilloscope. These results are then compared with the waveforms acquired from the physical plant, namely the STEVAL-7BIDIRCB.

The experimental measurements, performed in a laboratory dedicated to high-power applications, are based on the temporal evolution of the plant voltages and currents regulated by the controller implemented on the STM32G474RET6 microcontroller.

It is necessary to include the controller in the validation procedure because testing a power converter in open-loop operation would simply cause the plant dynamics to diverge, defeating the purpose of the converter itself. In the experimental measurements performed on the Totem-Pole converter, the presence of the controller was mandatory, since an uncontrolled divergence of voltages and currents would quickly lead to the destruction of the hardware due to the high power levels involved. Consequently, the validation of the plant model on the HIL board follows the same closed-loop methodology, providing a realistic, reliable, and objective comparison.

4.1. Software Validation

Let us begin by validating the Model through software by first developing a Simulink model that directly compares the high-fidelity model of the topology with our proposed model.

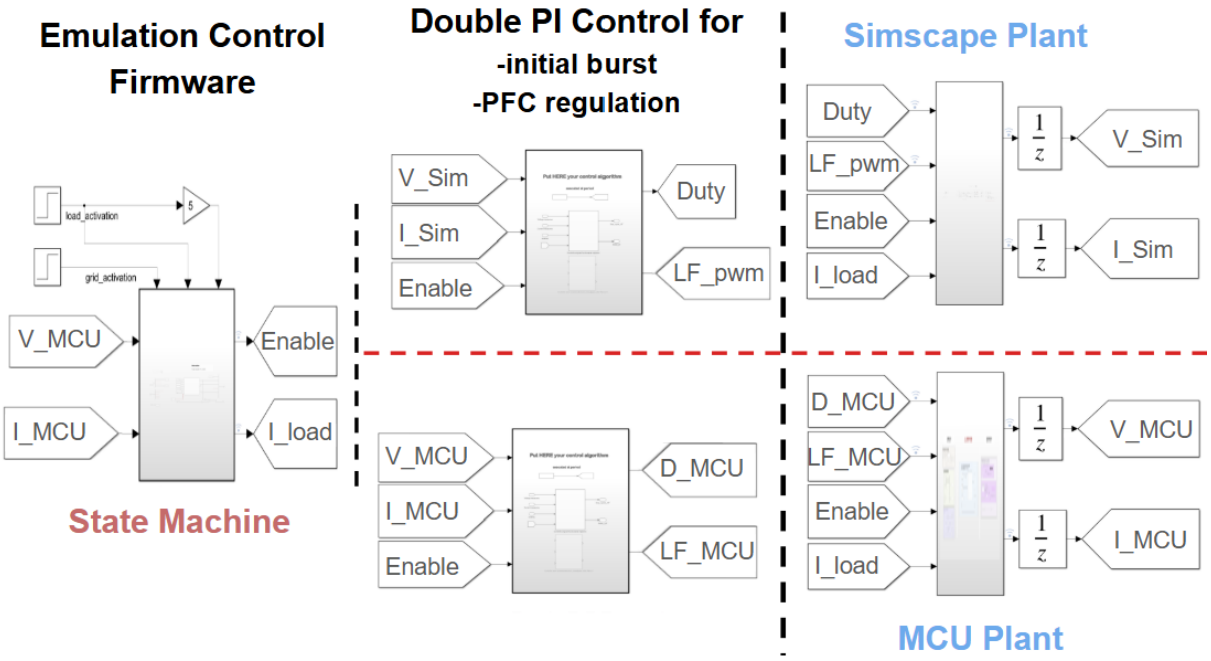

Figure 4.1: Simulink model

This model is composed of five subsystems containing:

- the Simscape plant model (top right);
- the microcontroller-oriented plant model (bottom right);
- the controller state machine (left);
- the PI controllers logics (center).

The combination of the state machine, the control algorithm, and the Simscape plant model defines the high-fidelity system emulation executed in Simscape (upper part of the figure). In contrast, the combination of the state machine, the same control algorithm, and the microcontroller plant model (lower part of the figure) constitutes the testbench used to validate the proposed model.

For the Simscape simulation performed within the Simulink environment, we configure a fixed-step Backward Euler solver with a discretization step $T_{s,\text{Simscape}}$ of 153.85 ns,

corresponding to one hundredth of the switching period. On the other hand, the sampling time of the microcontroller simulation is set equal to the switching period of the topology, namely 15.385 μs, which is also the exact timing that will be adopted for the HIL board implementation.

Since the two subsystems on the right emulate the plant at different execution rates, each of them is provided with its own dedicated controller, both driven by the same state machine in order to guarantee temporal synchronization throughout the simulation.

The outputs of both the Simscape plant simulation and the microcontroller plant simulation are then logged and visualized using the MATLAB Data Inspector, a MATLAB tool specifically designed for displaying and analyzing signals recorded during a simulation.

The signals logged from the Simscape model have already been presented in Figure 2.4 of Chapter 2. The same set of signals is extracted from the microcontroller model to enable a direct comparison.

While the Simscape implementation closely resembles the actual electrical circuit, the Microcontroller Plant Simulation is structured as follows:

Input

- discretized sine wave generator
- circuit parameter
- gate signal information and logical relay signals

Algorithm

The core of the model consists of a MATLAB Function block implementing the complete algorithm. It is developed for the inrush phase and for the first part of the burst mode by means of the state machine and the Backward Euler method, the second part of the burst mode and the PFC mode are instead implemented using the Average Model approach.

All the equations derived in Chapter 3 are therefore organized into a sequence of conditional branches, while continuity between the different operating modes is guaranteed through the use of two persistent variables: the capacitor voltage and the inductor current. At each simulation cycle, determined by the fixed-step solver, the MATLAB Function block is executed and the persistent variables are updated according to the current operating condition.

Output

At the output of the model, we provide the inductor current and the capacitor voltage, both updated step by step through the persistent variables. In addition, the input source voltages with respect to ground, namely `VACL` and `VACN`, are computed at each iteration according to the conduction state of the corresponding diodes or MOSFETs.

The resulting Model-Based Design implementation in the simulation environment is shown below.

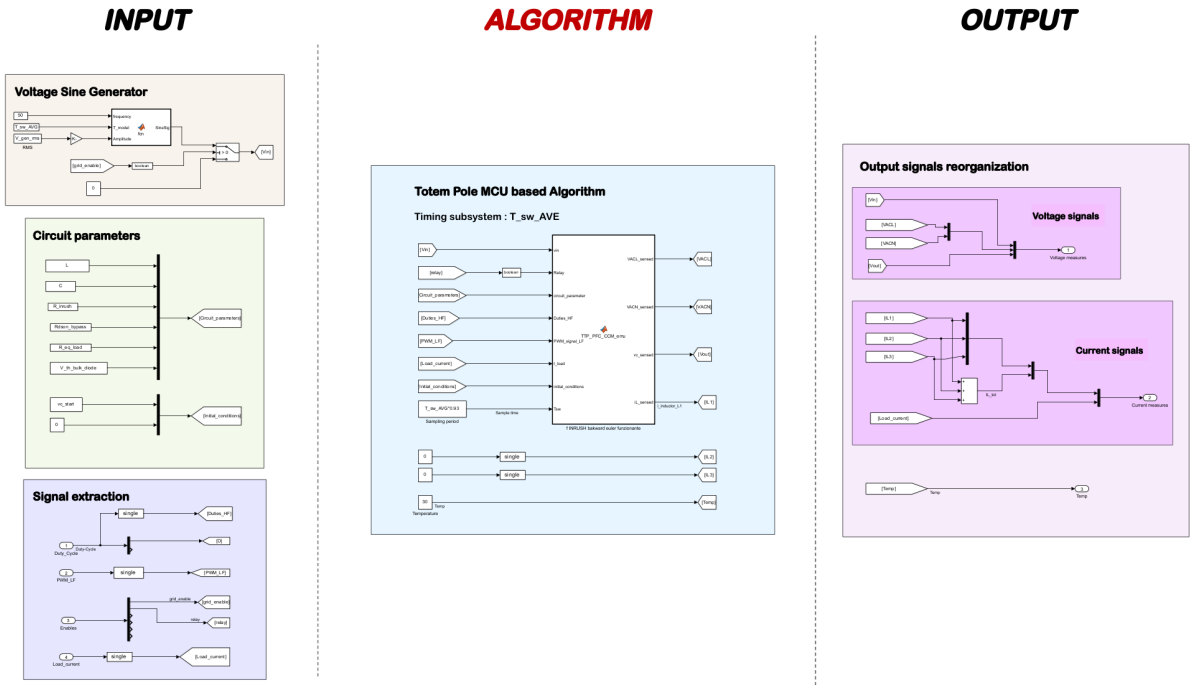

Figure 4.2: Simulink Microcontroller plant model

Before moving on to the simulations, it is important to briefly discuss the controller used in the validation process. While for the hardware validation we will use the firmware implemented in C code described in Section 2.4 of Chapter 2, for the software validation we need to reconstruct its behavior and implement it using a Model-Based Design approach.

In practice, we aim to digitally emulate the controller in order to debug the plant model. Although the original firmware, downloadable from eDesign for the STM32G474RET6 microcontroller, is written in C, it cannot be directly used in MATLAB. The reason is straightforward: while MATLAB supports integration of C code through specific interfaces, the firmware in question also includes hardware abstraction and peripheral management layers, such as HAL (Hardware Abstraction Layer) functions, which are not compatible with a direct simulation environment.

Therefore, wrapping the entire firmware inside a Simulink C Function block is possible in principle, but the code would need to be extensively restructured. The approach adopted instead was to develop a simplified controller in Simulink, which retains fewer safety mechanisms but still manages the main state transitions, namely relay control and PWM generation.

This made it possible to build a controller suitable for debugging and validating the response of the plant model to these input signals.

The main differences can be summarized in two points:

- fault management;
- operating frequency.

Indeed, in the Model-Based Design simulation of the controller, fault handling is not included in the state machine; only the transition chain from WAIT to RUN is implemented, as shown in Figure 2.5. This is because the firmware used in the hardware validation phase will be responsible for managing faults and safely disabling the MOSFETs by controlling the PWM signals.

At this stage, our main objective is to ensure that the plant responds correctly to the input stimuli. Once this is verified, the specific duty cycle value generated by whatever control (e.g., 30% or 70%) becomes irrelevant, since this simplified test already provides confidence that the plant behaves correctly under all operating conditions.

Another important aspect concerns the controller operating frequency. In the real implementation, the controller processes signals from the peripherals using moving averages and high-frequency filtering to avoid aliasing during sampling. In order to avoid this additional complexity, the simulation runs the controller at the same frequency as the plant model.

Consequently, the controller for the Simscape plant operates at 6.5 MHz, whereas the controller for the microcontroller plant operates at 65 kHz. It should be emphasized that this is also a significant simplification, although it is fully appropriate for the validation purposes of this work.

4.1.1. Software Simulation Results

By pressing `RUN`, the complete model is executed. With the two plant models running in parallel, the simulation requires approximately 2–3 minutes to compile and simulate 4 seconds of transient behavior. In contrast, consistently with the different discretiza-

tion step, executing only the Microcontroller Plant model while disabling the Simscape simulation reduces the execution time to approximately one second.

The logged signals are then visualized using the MATLAB Data Inspector.

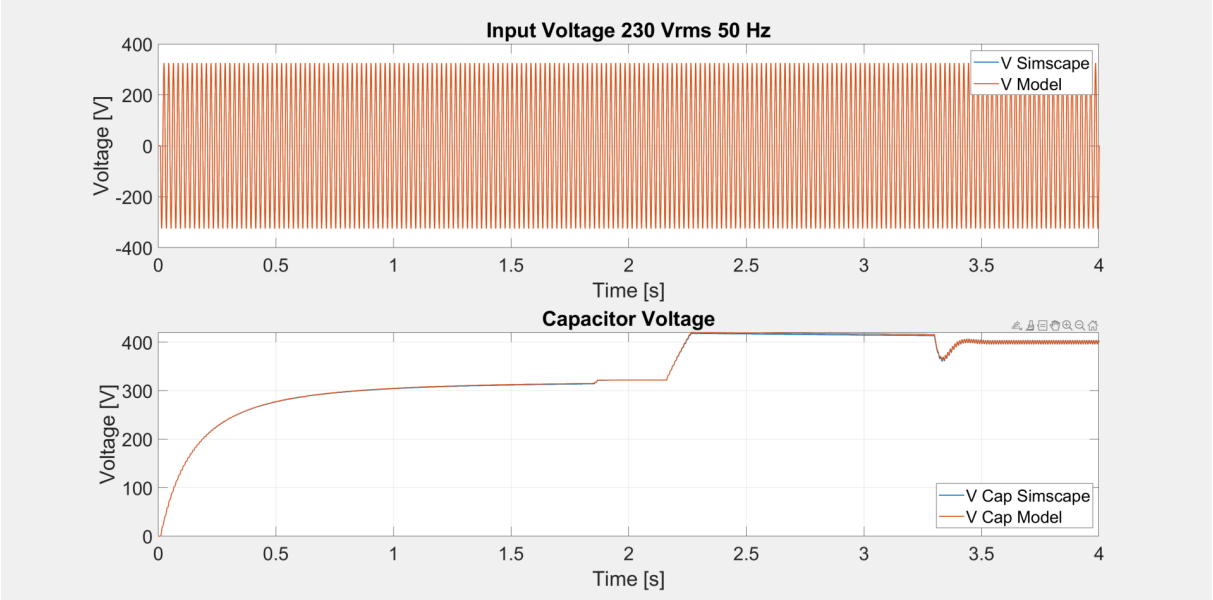

Figure 4.3: Comparison 1/2 between Simscape results and Model results

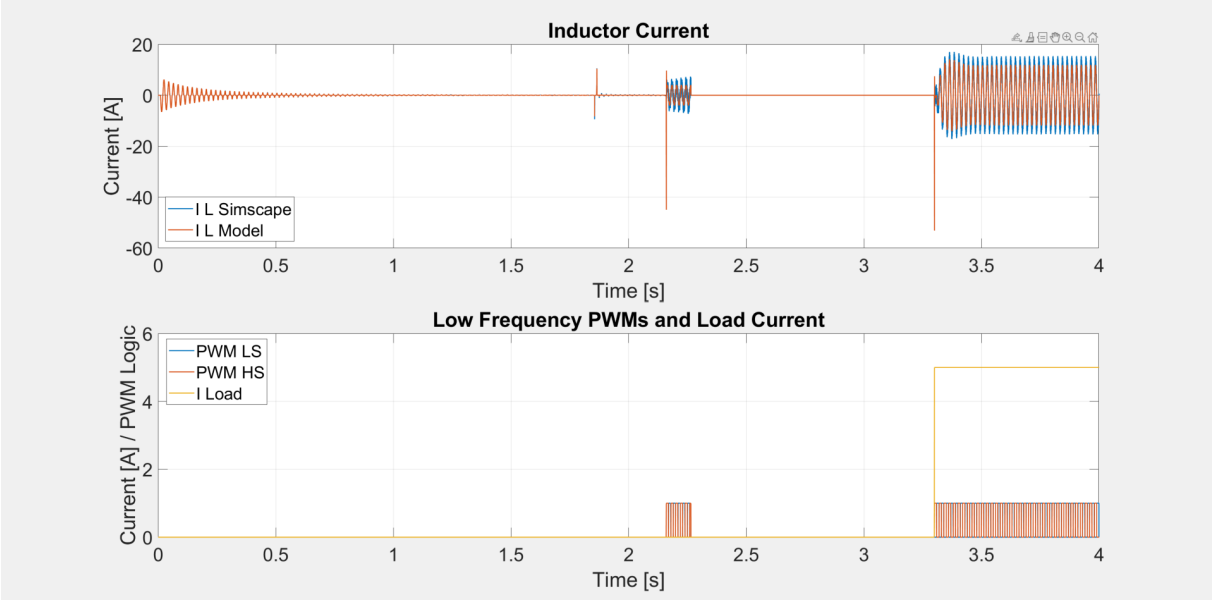

Figure 4.4: Comparison 2/2 between Simscape results and Model results

As can be observed from the figure, the high-frequency Simscape simulation and the proposed Average Model produce almost identical results. In the simulations, the output

capacitance was set to $C = 4 \times 470\ \mu\text{F} = 1.88$ mF, while the inductance was chosen as $L = 211\ \mu\text{H}$.

From this overall view, we can clearly identify

- the initial exponential charging phase of the output capacitor through the body diodes ($R_{\text{inrush}} = 47\ \Omega$);
- the small voltage step caused by the closure of the relay that bypasses the inrush resistor;
- the burst-mode ramp, obtained by injecting a constant current (2 A RMS) into the output capacitor through appropriate duty-cycle regulation;
- the slow discharge of the capacitor after the burst phase due to the modeled parasitic effects (for the Simscape model: $R_{\text{ds\_ON}} = 40\ \text{m}\Omega$ and $R_{L,\text{series}} = 50\ \text{m}\Omega$; for the Microcontroller model: $R_{\text{bypass}} = 100\ \text{m}\Omega$ and $R_{\text{load\_eq}} = 100\ \text{k}\Omega$);
- the application of a 5 A load step.

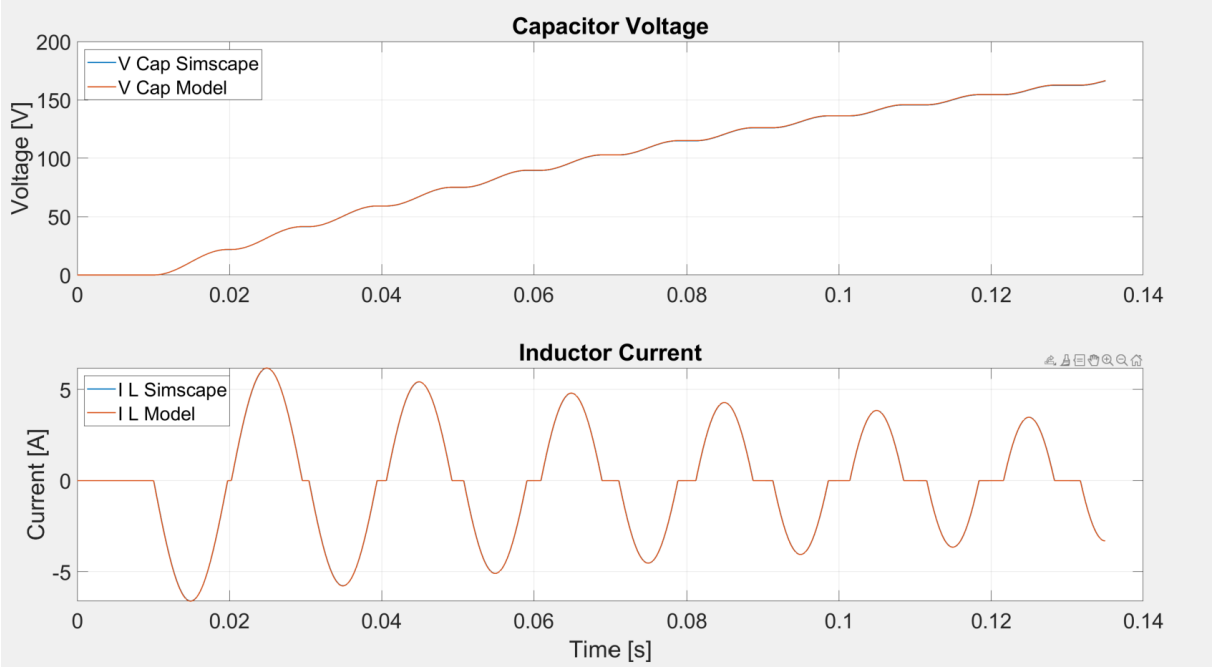

Figure 4.5: Start Inrush Comparison

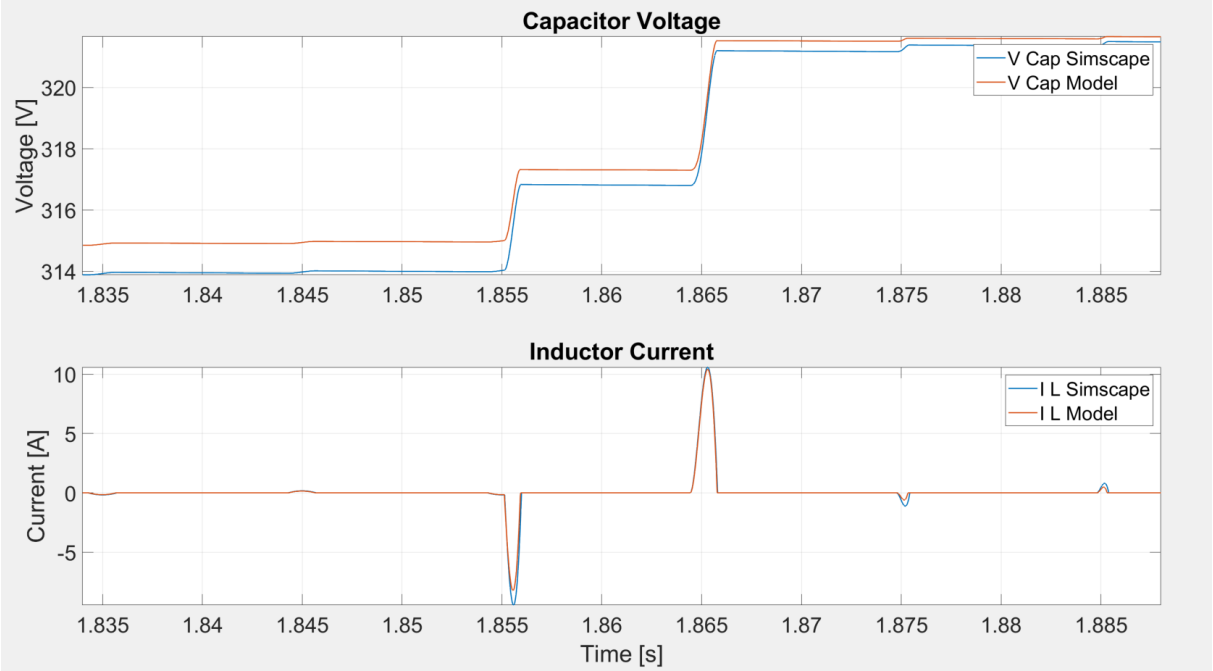

Figure 4.6: Burst phase 1 Comparison

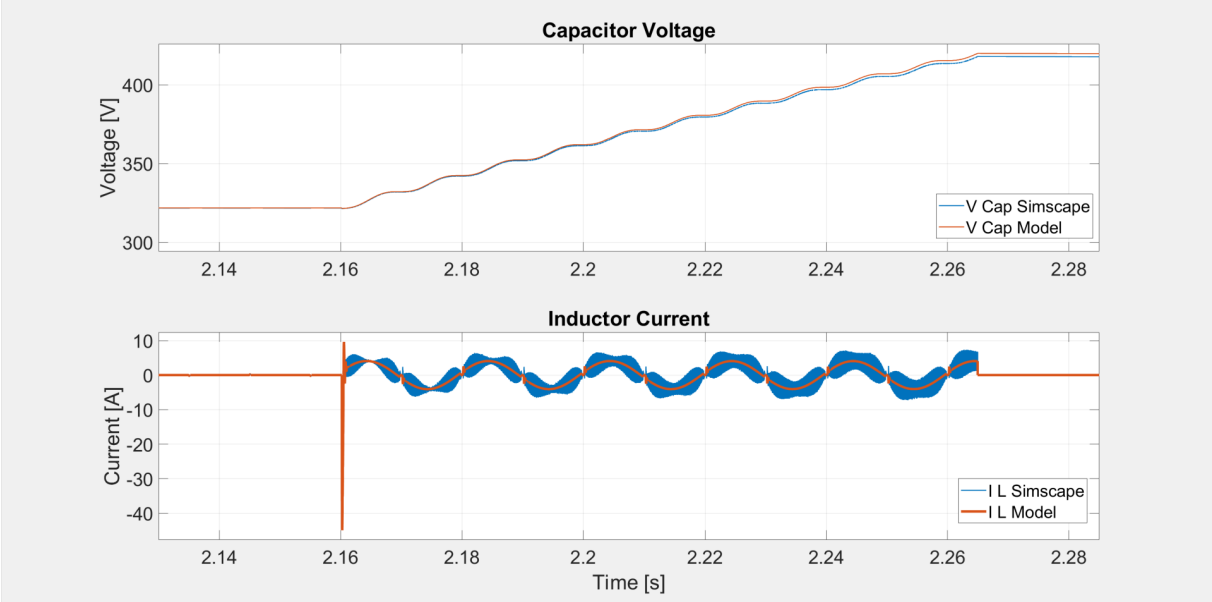

Figure 4.7: Burst phase 2 Comparison 1/2

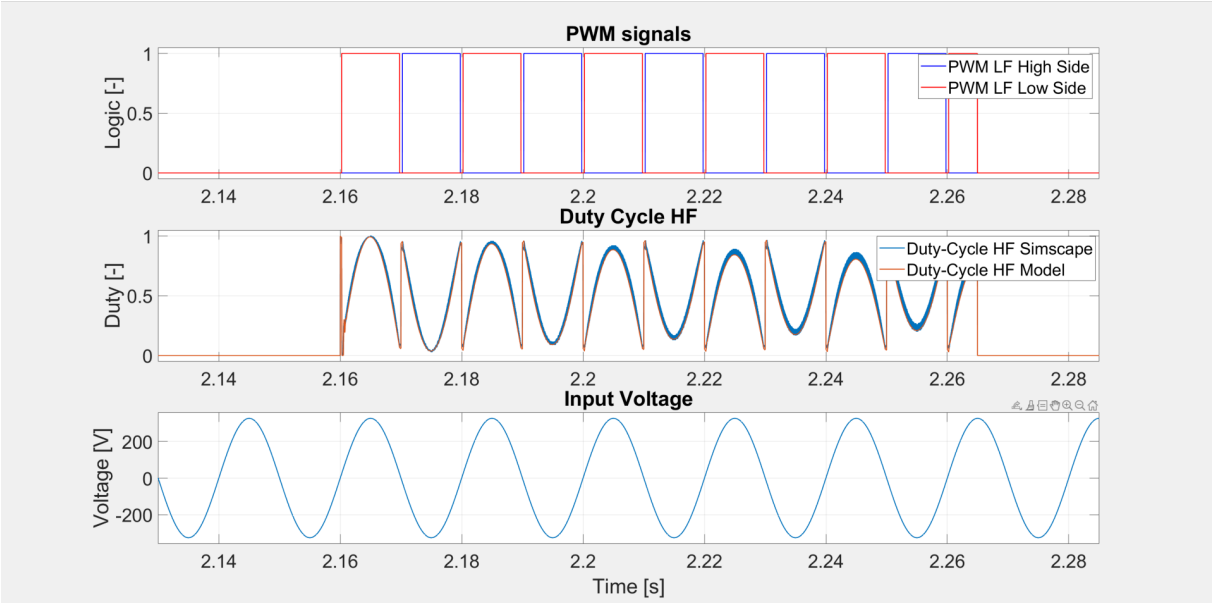

Figure 4.8: Burst phase 2 Comparison 2/2

PFC Mode - Load step

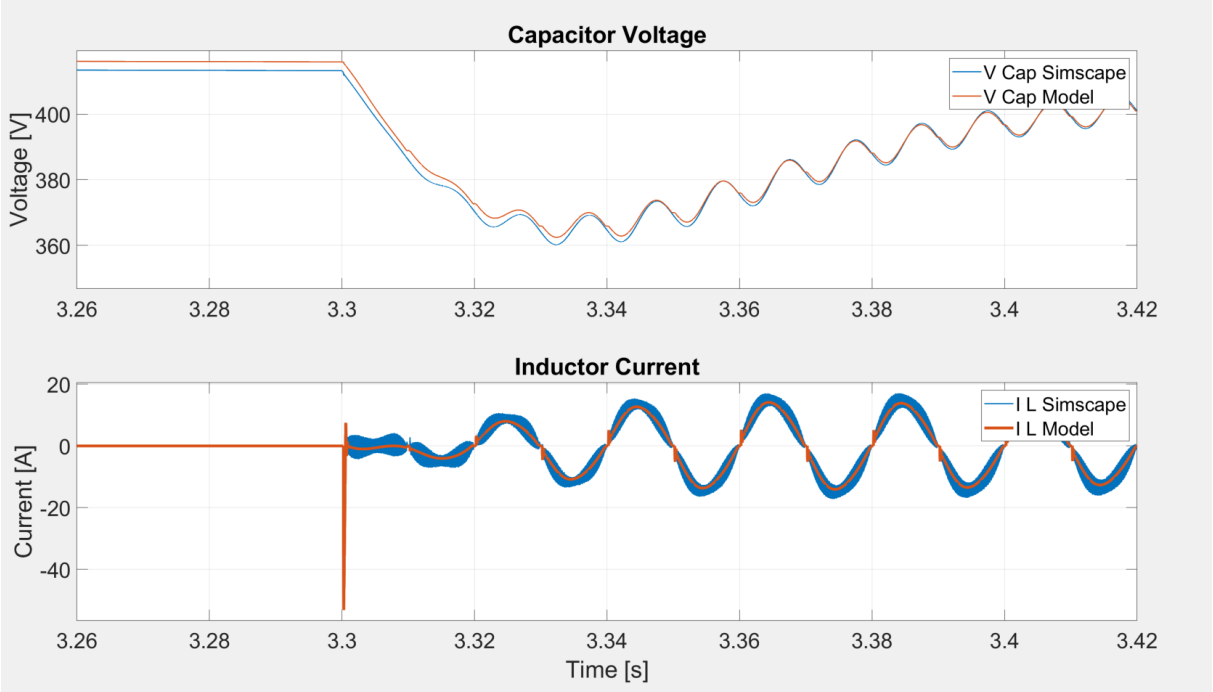

Figure 4.9: PFC Step phase Comparison 1/2

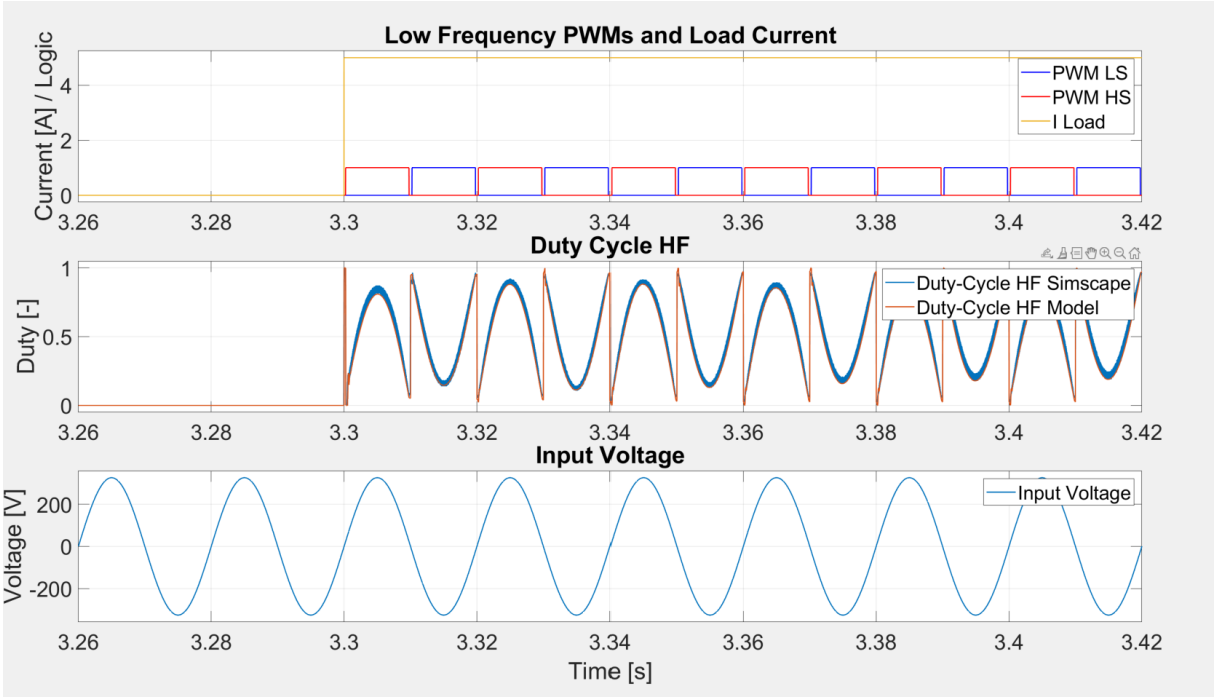

Figure 4.10: PFC Step phase Comparison 2/2

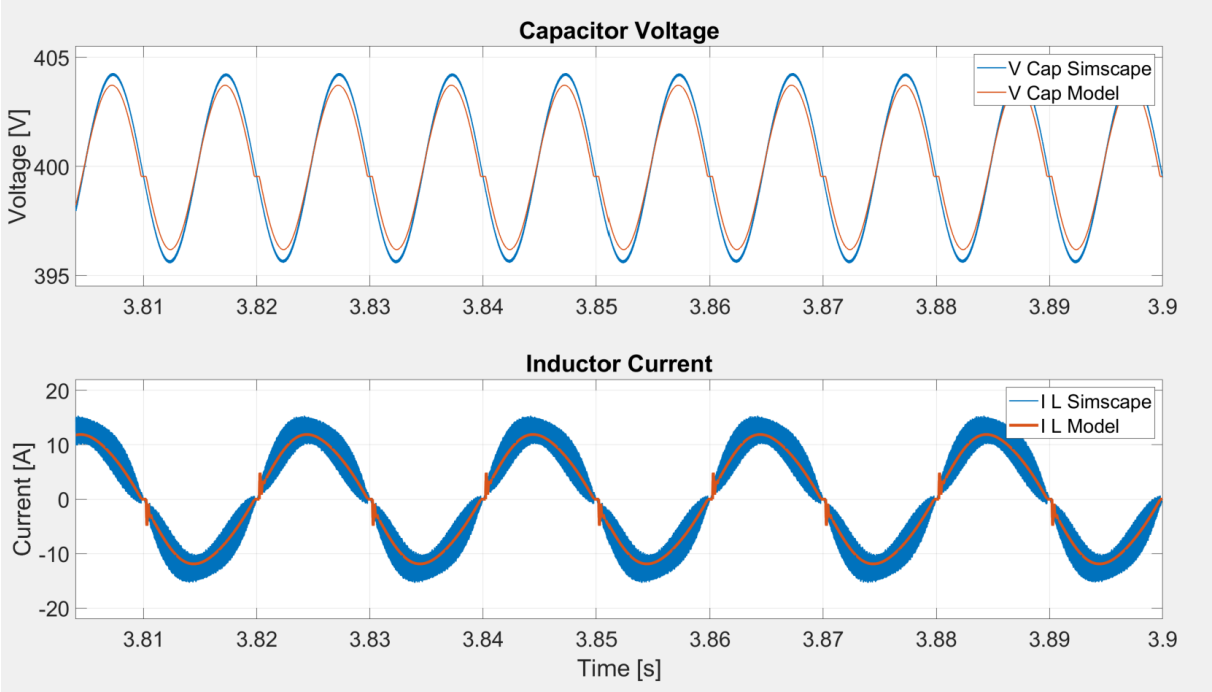

Figure 4.11: PFC Steady-State phase Comparison 1/2

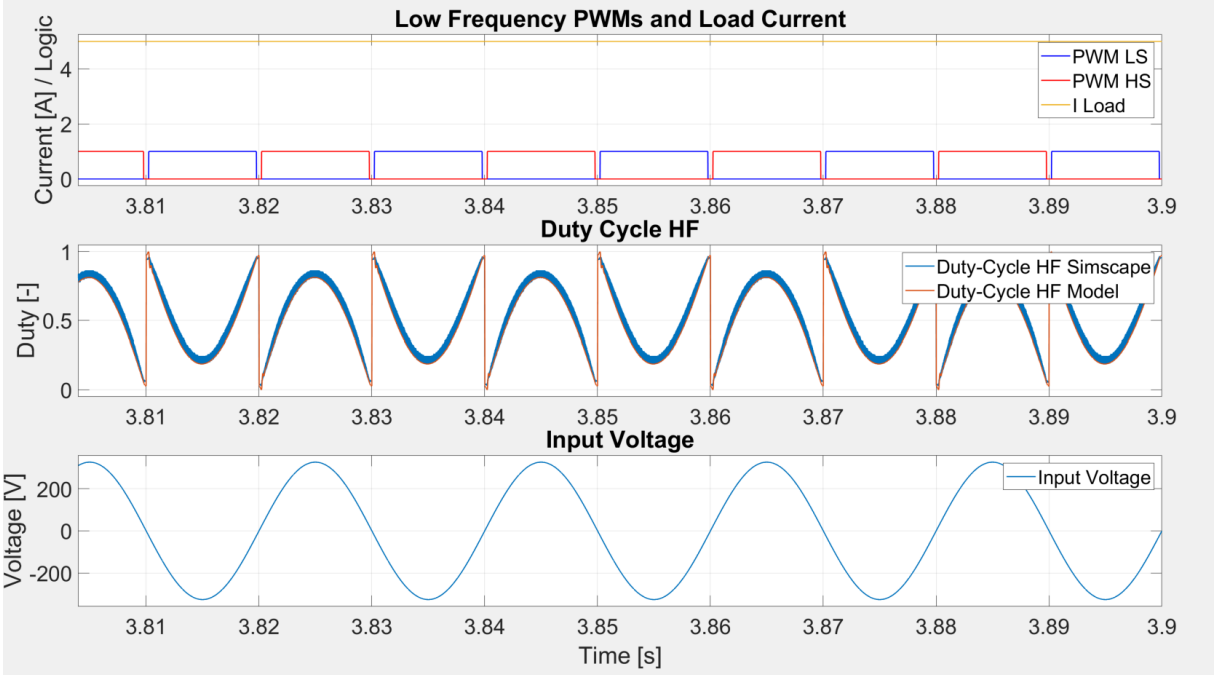

Figure 4.12: PFC Steady-State phase Comparison 2/2

By analyzing the different operating phases, it can be observed that an almost perfect overlap between the two models is achieved throughout the entire simulation. During

the inrush phase, the state machine consistently identifies the correct diode conduction states, as demonstrated by the excellent agreement in both the magnitude and direction of the current waveform. Furthermore, no numerical divergence appears during the various state transitions, confirming the stability guaranteed by the Backward Euler integration method.

At the end of the first inrush stage, a small discrepancy of approximately one volt can be observed between the two capacitor voltage waveforms. This difference is mainly due to minor parasitic effects that are locally modeled in Simscape, whereas in our simplified implementation they are lumped into the equivalent resistances R_{bypass} and $R_{\text{load\_eq}}$. Nevertheless, the deviation is less than 1% with respect to the approximately 300 V bus voltage reached during this phase and can therefore be considered negligible. Passing to the stage in which both PWM signals become active, it can be observed that our model accurately reproduces the behavior of the Average Model. While the voltage ripple is almost negligible due to the attenuation provided by the large output capacitance, the inductor current clearly highlights the averaging effect: the orange waveform generated by our model follows the average value of the blue current waveform obtained from the high-fidelity Simscape simulation.

As discussed in the previous chapter, the switching ripple is so fast that the blue waveform appears almost as a solid trace at this time scale. However, by zooming in, the characteristic behavior shown in Figure 3.5 of Chapter 3 can be clearly recognized. The dead time associated with the low-frequency switching is also correctly averaged, without introducing artificial spikes or numerical discontinuities.

It can also be observed that the high-frequency duty-cycle signals generated by the two controllers are almost identical, further confirming the equivalence of the control actions applied to the two plant models. Similarly, the response to the applied load step is accurately reproduced, demonstrating that the proposed model is able to capture the dynamic behavior of the converter even under transient operating conditions.

Looking at the inductor current, which corresponds to the input current drawn from the grid, it is in phase with the input voltage. This confirms that the controller correctly performs the Power Factor Correction (PFC) function.

Furthermore, once the system reaches steady-state operation, a power balance can be carried out to verify the consistency of the simulation results. As previously described, the simulation includes a load step corresponding to 5 A, a regulated DC bus voltage of 400 V, an AC input voltage of 230 V_{rms}, and a sinusoidal inductor current with a peak value of approximately 12.3 A, corresponding to 8.69 A_{rms}.

By performing the power balance, we obtain

$$P_{in} = 230[V\_rms] \cdot 8.69[A\_rms] \sim 2000[W] = 400[V] \cdot 5[A] = P_{out}$$

In this case, the most dominant apparent error is the initial current spike observed in the inductor when the PWM is enabled for the first time [Figure 4.9], both during the burst phase and during the load step transition. However, it is important to note that the same spike is present in both the Simscape simulation and our proposed model, which confirms that the two simulations remain fully consistent.

This indicates that the source of the discrepancy is not the plant model itself, but rather the controller implementation. In particular, a more refined control strategy would require the inclusion of a PWM soft-start technique in order to avoid these current peaks during transients. At this stage, however, this improvement is not relevant, since it pertains to controller optimization rather than plant validation.

Our objective is to verify the correctness of the plant behavior, and this strong overlap, despite the use of a simplified controller, confirms that goal.

We can therefore conclude that the microcontroller-implementable plant model correctly reproduces the system dynamics, with only minor and negligible discrepancies. The next step is the hardware implementation of the model.

4.2. Hardware Validation

The hardware validation represents the final stage of the verification process, allowing us to demonstrate that the proposed model, once implemented on a microcontroller, is capable of accurately emulating the real-time behavior of the Totem-Pole topology.

To perform this validation, the following experimental setup is adopted:

- an HIL board running the microcontroller-oriented plant model;
- an ST Nucleo development board featuring the STM32G474RE microcontroller, programmed with the original control firmware.

The HIL board used for the emulation is the one included in the ST *STEVAL-HILE1G4K1* kit, while the controller is implemented using an ST Nucleo board, specifically the *STM32 Nucleo-64 development board with STM32G474RE MCU*. This choice is motivated by the fact that the control board included in the kit is based on a different microcontroller and is therefore not fully compatible with the porting of the control firmware used in this

work.

On the other hand, the Nucleo board exposes all the microcontroller pins through its connectors, making it particularly suitable for our application. The only required step is to correctly map the Nucleo board I/O pins to the corresponding signals available on the 64-pin connector of the HIL board, ensuring proper communication between the controller and the emulated plant.

4.2.1. Nucleo–HIL Porting and Wiring

To define the port mapping, we first analyze the `.ioc` file associated with the control firmware, which specifies the configuration and usage of the microcontroller pins.

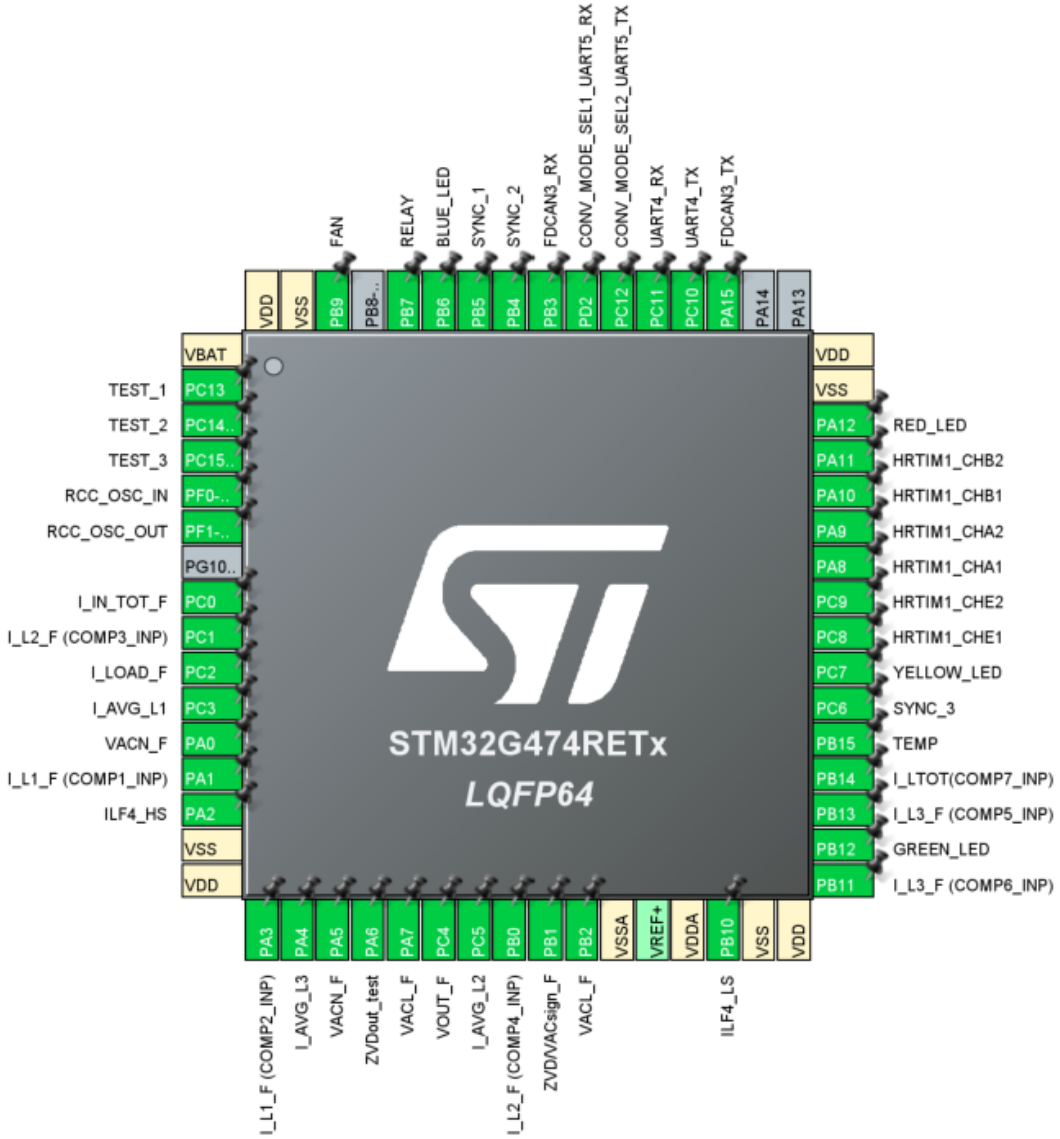

Figure 4.13: Control Firmware IOC

Starting from this information, all the pin assignments are extracted into an Excel spreadsheet and grouped into the following categories:

- **Control Firmware inputs**, which must be connected to the outputs of the HIL board;
- **Control Firmware outputs**, which must be connected to the input channels of the HIL board;
- **Internal control connections**, namely signals that are internally routed within the controller itself in order to support and verify the correct operation of the control firmware.

At this point, using the Nucleo board datasheet [20], we identify the physical connector pins corresponding to the various microcontroller I/O ports.

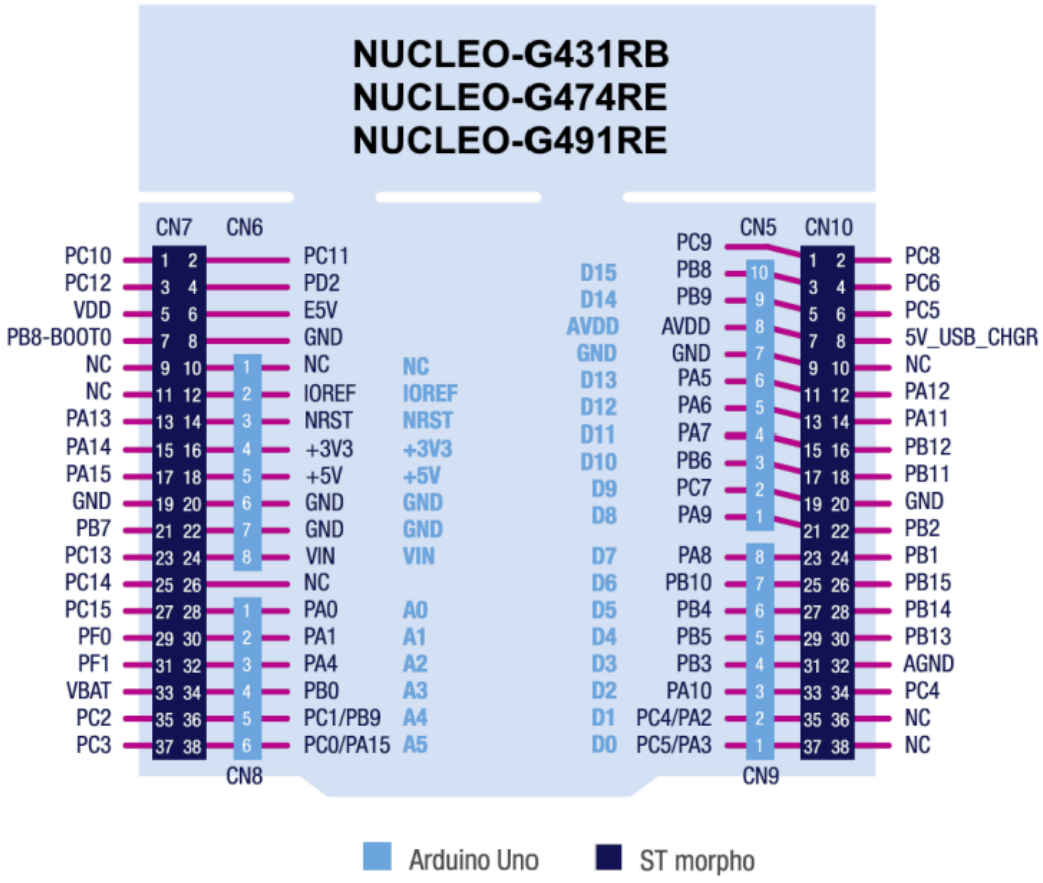

Figure 4.14: NUCLEO-G474RE Pinout

Similarly, from the HIL board datasheet, we extract the pinout of the exposed 64-pin connector.

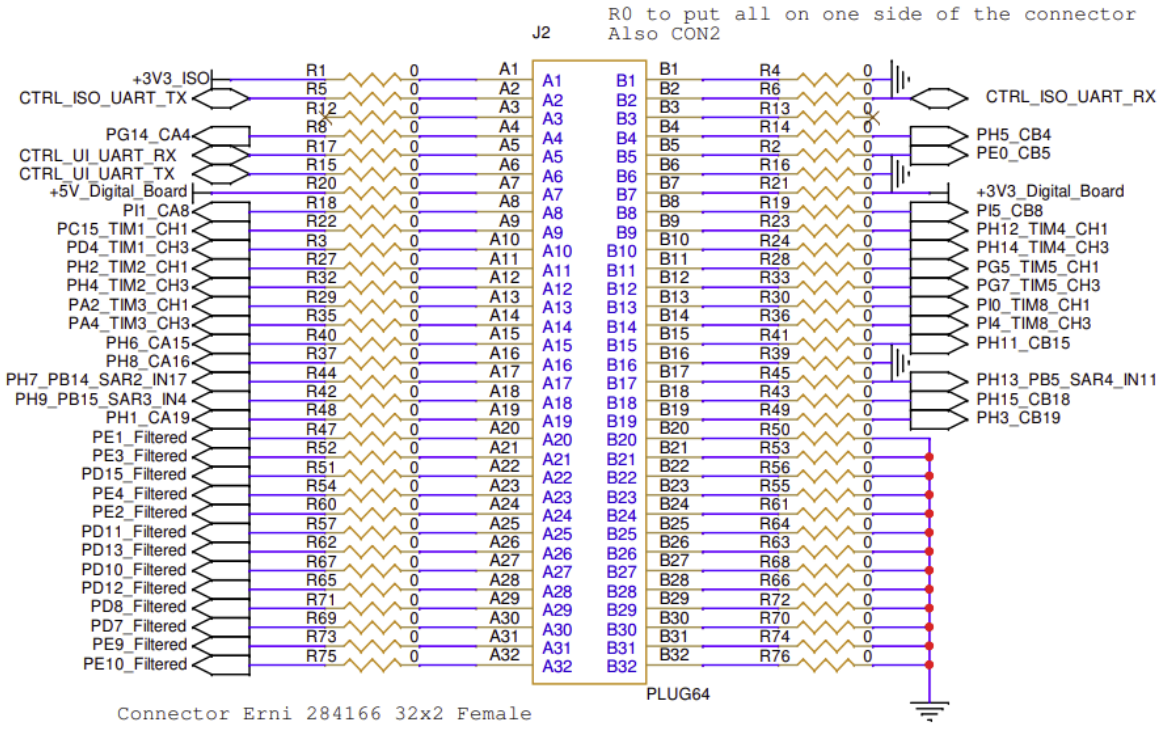

Figure 4.15: HIL-board 64-pin Connector Pinout

While the I/O assignment on the controller side is fixed by the firmware implementation, the HIL board offers complete flexibility, allowing us to choose which physical pins are used to expose the various input and output signals according to the most convenient wiring configuration.

The complete mapping adopted for this work is summarized in the following three Excel tables.

Input Control - Output HIL

Name	LEGENDA	CONN NUCLEO	HIL
PA0	VACN_F	CN7	A23
PA5	VACN_F	CN10	
PA7	VACL_F	CN10	A24
PB2	VACL_F	CN10	
PC4	VOUT_F	CN10	A25
PA1	I_L1_F	CN7	A26
PA3	I_L1_F	CN10	
PC3	I_AVG_L1	CN7	
PC1	I_L2_F	CN7	A27
PB0	I_L2_F	CN7	
PC5	I_AVG_L2	CN10	
PB11	I_L3_F	CN10	A28
PB13	I_L3_F	CN10	
PA4	I_AVG_L3	CN7	
PB14	I_LTOT	CN10	A29
PC0	I_IN_TOT_F	CN7	
PC2	I_LOAD_F	CN7	A30
PB15	TEMP	CN10	A31

HIL OUT -> IN Nucleo	
A1	B1
A2	B2
A3	B3
A4	B4
...	...
A17	B17
A18	B18
A19	B19
A20	B20
A21	B21
A22	B22
A23	B23
A24	B24
A25	B25
A26	B26
A27	B27
A28	B28
A29	B29
A30	B30
A31	B31
A32	B32

POSITION INPUT NUCLEO

CN7		CN10	
1	2	1	2
3	4	3	4
5	6	5	6
7	8	7	8
9	10	9	10
11	12	11	12
13	14	13	14
15	16	15	16
17	18	17	18
19	20	19	20
21	22	21	22
23	24	23	24
25	26	25	26
27	28	27	28
29	30	29	30
31	32	31	32
33	34	33	34
35	36	35	36
37	38	37	38

Figure 4.16: Control port (LEFT) - HIL board OUT port - Nucleo board IN port (RIGTH)

Output Control - Input HIL

Name	Label	CONN NUCLEO		HIL
PA9	HRTIM1_CHA2	CN10	21	A13
PA8	HRTIM1_CHA1	CN10	23	A14
PA2	ILF4_HS	CN10	35	A15
PB10	ILF4_LS	CN10	25	A16
PB7	RELAY	CN7	21	A17

POSITION OUTPUT NUCLEO

CN7		CN10	
1	2	1	2
3	4	3	4
...	...	...	...
21	22	21	22
23	24	23	24
25	26	25	26
27	28	27	28
29	30	29	30
31	32	31	32
33	34	33	34
35	36	35	36
37	38	37	38

OUT Nucleo -> HIL IN	
A1	B1
A2	B2
...	...
A12	B12
A13	B13
A14	B14
A15	B15
A16	B16
A17	B17
A18	B18
...	...
A30	B30
A31	B31
A32	B32

Figure 4.17: Control port (UP LEFT) - Nucleo board OUT (DOWN LEFT) - HIL board IN (RIGTH)

Internal Connections

Name	Label	CONN NUCLEO			Name	Label	CONN NUCLEO
PA6	ZVDout_test	CN10	13	⟷	PB1	ZVD/VACsign_F	CN10
PB5	SYNC_1	CN10	29	⟷	PA8	HRTIM1_CHA1	CN10

Figure 4.18: Control port (UP LEFT) - Nucleo board OUT (DOWN LEFT) - HIL board IN (RIGTH)

By following the wiring scheme described above and assigning the corresponding I/O ports in the plant model implemented on the microcontroller, we can proceed with the experimental validation.

4.2.2. Hardware Validation Results

Once all the electrical connections have been completed, the only remaining step is to deploy the proposed model onto the HIL board. At this stage, the advantages of the Simulink development environment become particularly evident. In fact, it is sufficient to create a new model defining the required I/O mapping, copy the plant model previously debugged through software validation, and the Totem-Pole emulator is immediately ready for deployment.[21]

Through the Hardware Support Package (HSP), the peripherals of the HIL board are configured and managed in order to physically generate the output waveforms produced by the plant model, properly scaled according to the corresponding sensor gains, as described in Section 2.3 of Chapter 2.

At this point, the resulting waveforms can be directly observed and analyzed using an oscilloscope.

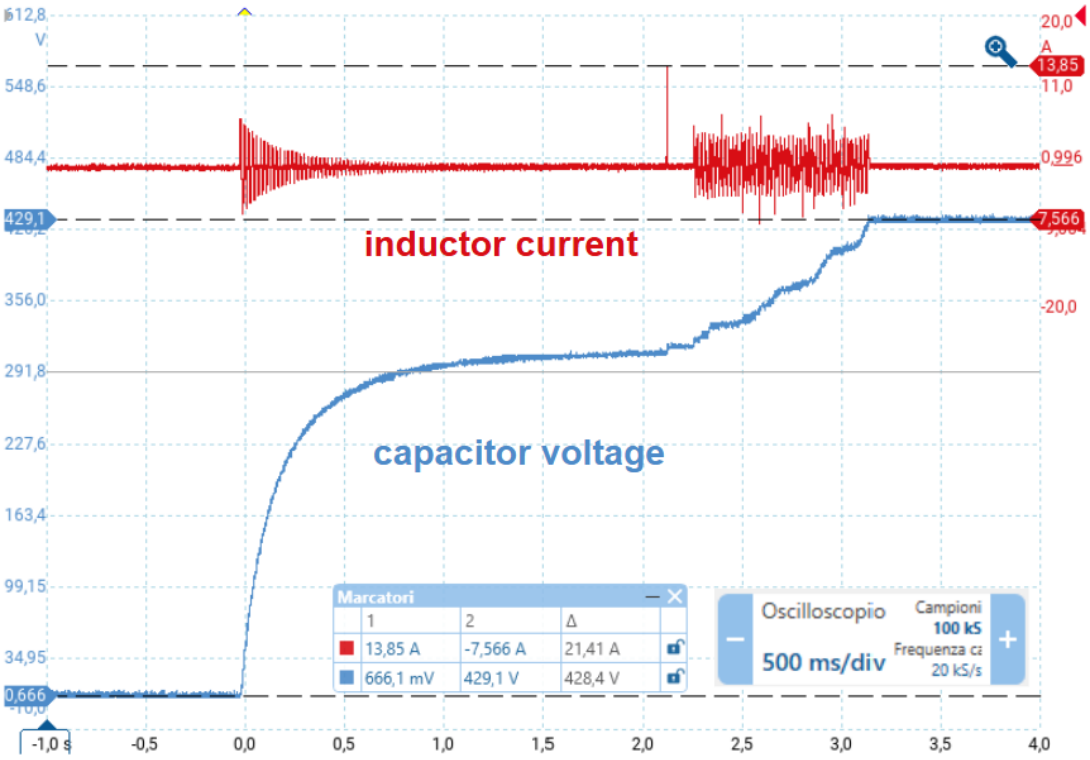

Figure 4.19: Oscilloscope Validation

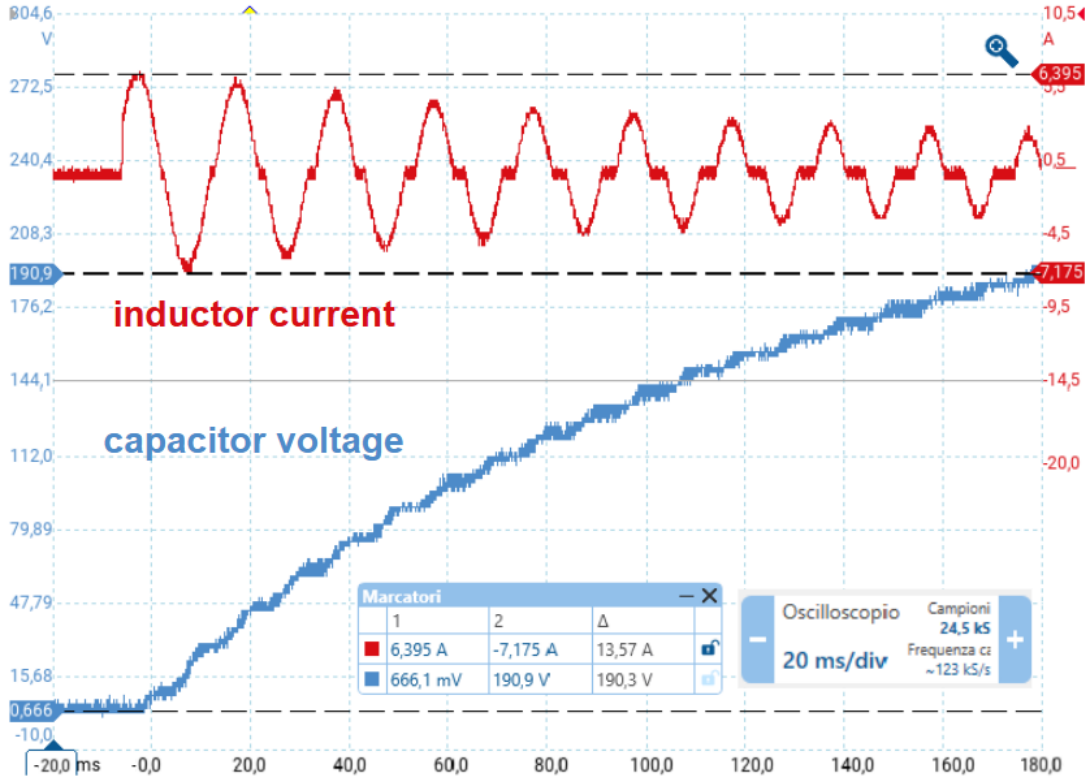

Figure 4.20: Start Inrush

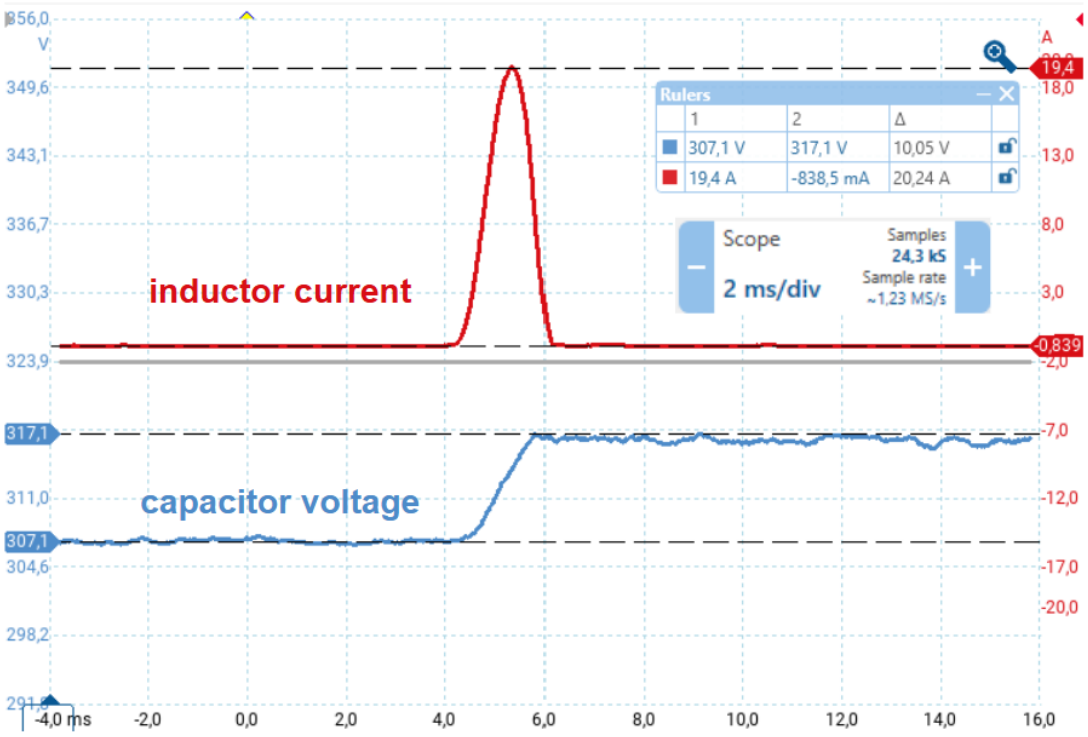

Figure 4.21: Burst phase 1

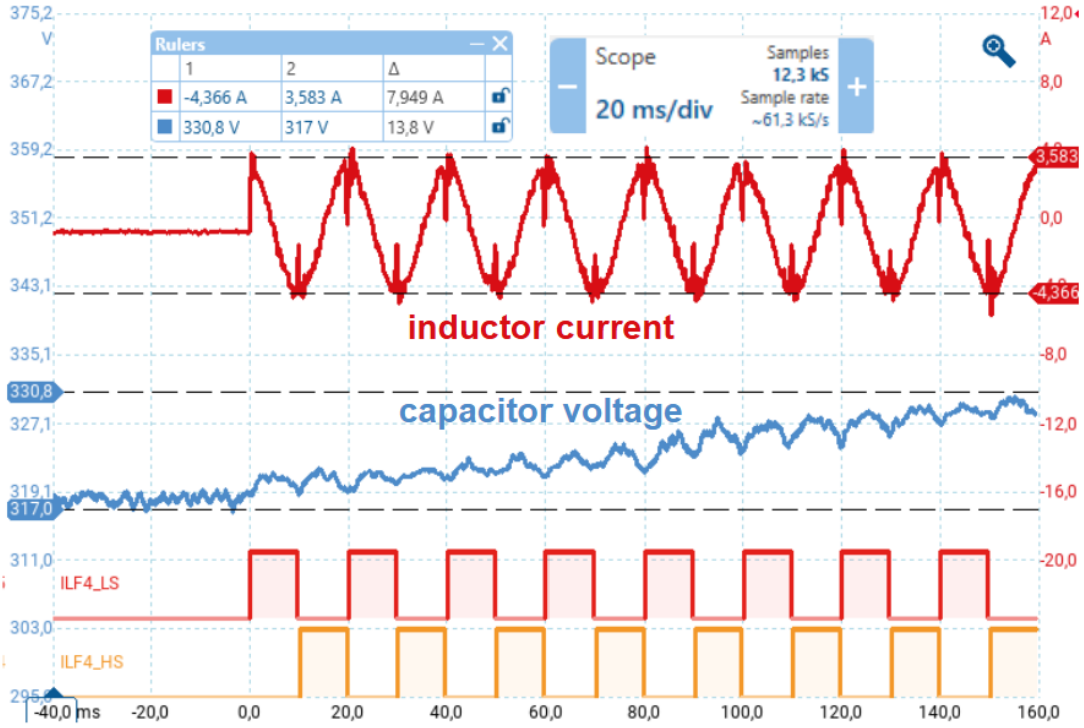

Figure 4.22: Burst phase 2

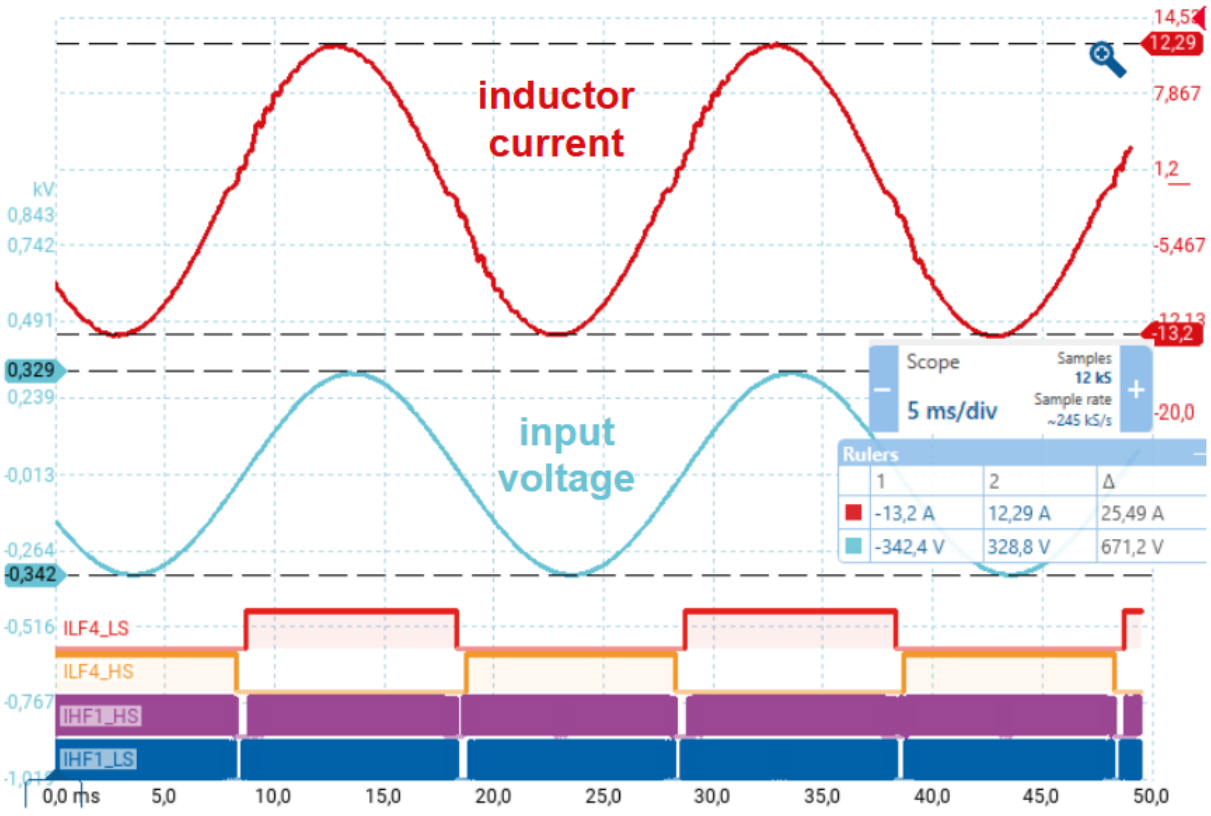

Figure 4.23: PFC Mode - Steady State

As can be observed across the different operating phases, the first stage once again confirms the high stability and robustness of the model, mainly ensured by the Backward Euler formulation. In this region, the model behaves correctly even in the hardware implementation. In the first burst phase, there is only one peak, compared to the simulation where there are two. This is simply a consequence of the different closing point of the relay; if it trips at a capacitor voltage higher than our control, the first current injection is sufficient to bring it to steady state.

While regarding the second burst phase, some numerical glitches can be observed in the current emulation. However, the controller monitored via STM32CubeIDE does not enter in a fault state and, as evidenced by the correct voltage ramp behavior, the system remains operational, indicating that the model is still valid but requires further refinement.

A more detailed analysis shows that the glitch originates specifically during the dead-time interval of the Low Frequency PWM. This is due to the fact that the current implementation does not explicitly include an additional dead-time contribution associated with the PWM soft start-up behavior. During this interval, the low-frequency PWM signals are turned off, while only one of the two high-frequency PWM signals is active. In this

case, the controller implemented on the Nucleo board expects a PWM soft-start phase; however, since this behavior is not included in the model, it has no corresponding effect in the emulation. Consequently, when the PWM signals switch to full operation, the controller has accumulated an error that results in a current spike call, producing the observed glitch.

This represents the only current limitation of the model; however, it can be addressed by introducing a dedicated Average Model formulation that explicitly includes this soft-start behavior.

Finally, the model performs correctly in steady-state operation and, as shown in the last figure, the power balance is consistent with the results obtained in simulation. It can also be concluded that the system responds properly to both load steps and load variations.

For completeness, I also report the measurements actually performed on physical Totem-Pole which attest to the quality of the emulation.

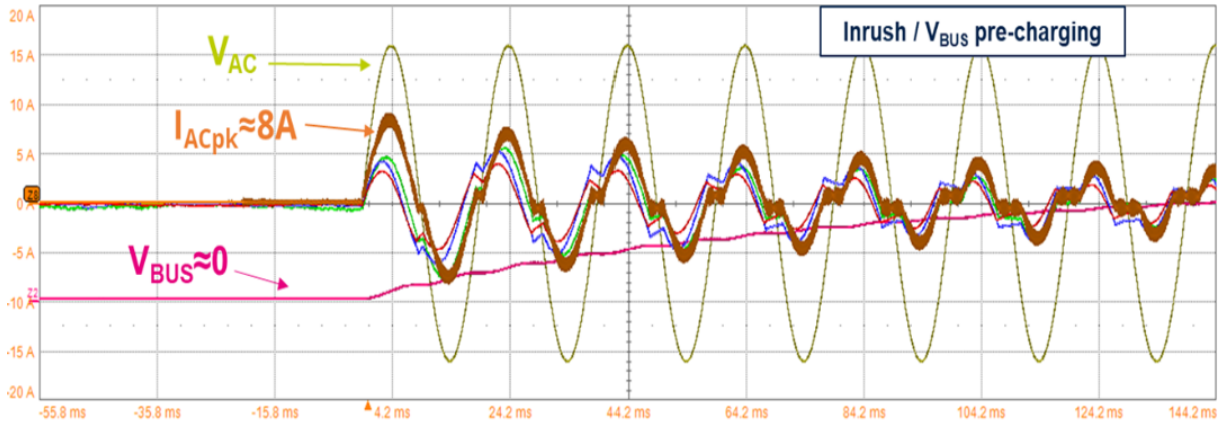

Figure 4.24: Physical board measurements

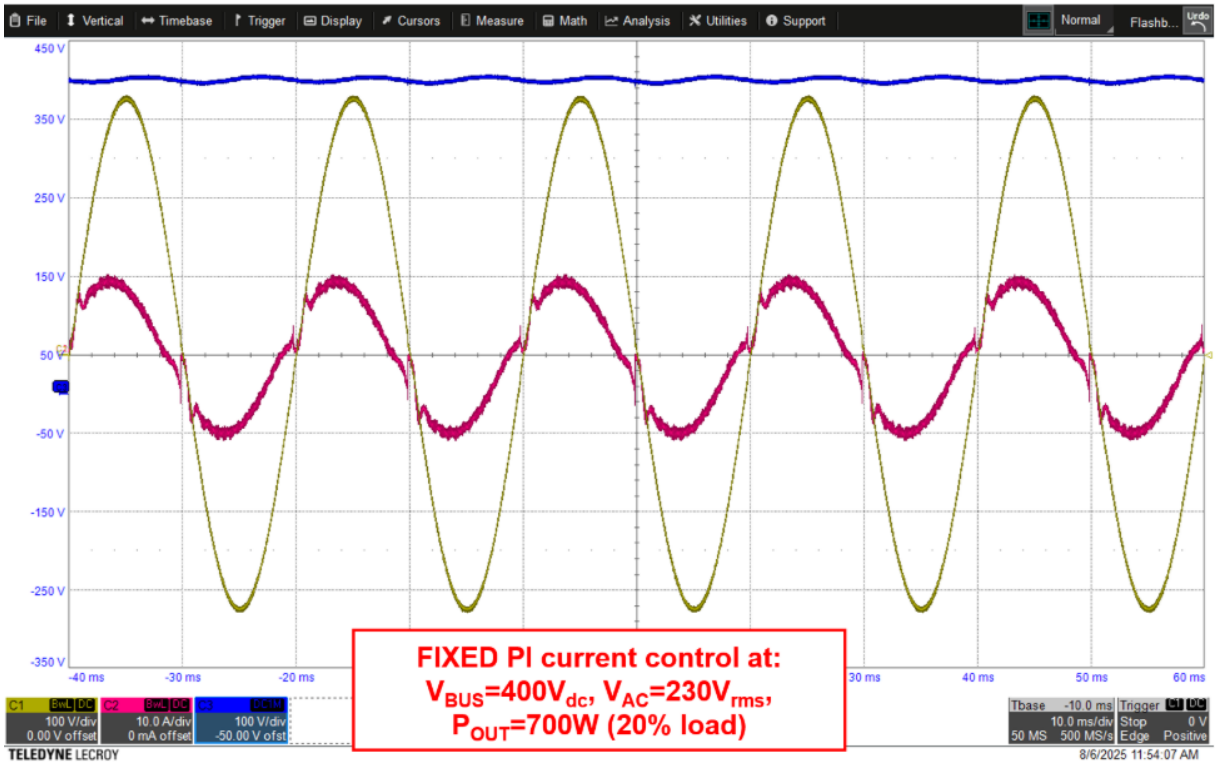

Figure 4.25: Physical board measurements

And finally a photo of the complete hardware setup to give a concrete idea of the work

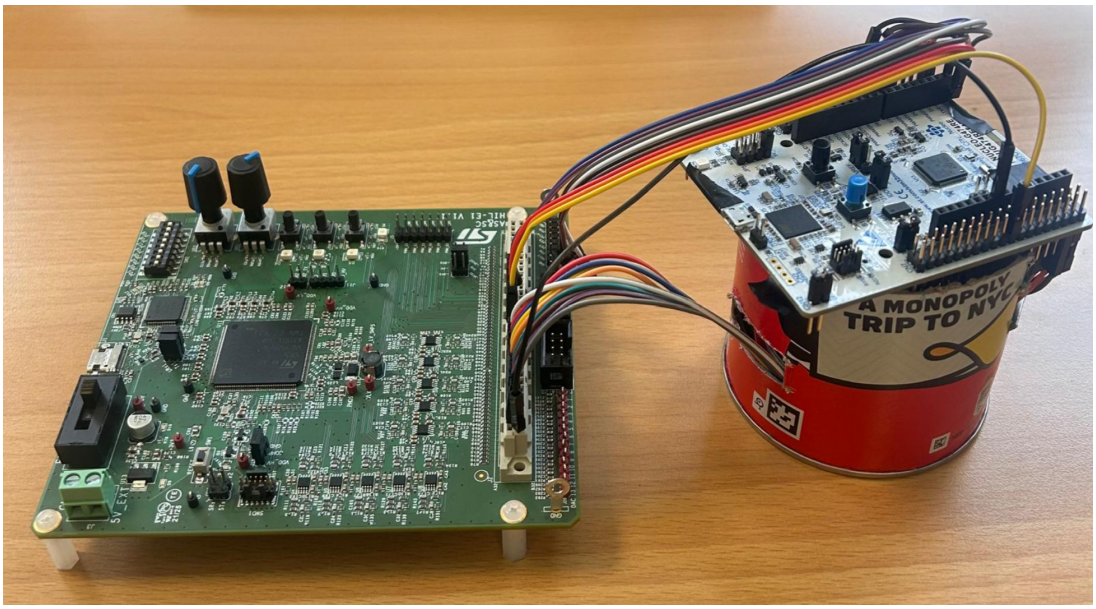

Figure 4.26: Hardware Setup

5 | Conclusions and future developments

We have now reached the conclusion of this thesis after a long development process that led us from the initial idea of emulating a power converter topology to its full implementation. The requirements of safety and simplicity in the Hardware-in-the-Loop approach have been addressed through an intensive modeling effort, which ultimately led to satisfactory results.

As shown in the hardware validation, which is generally the most challenging and representative compared to purely software-based simulations, the system exhibits a correct and consistent behavior. As previously discussed, the mitigation of the glitches caused by PWM soft-start effects would further improve the overall performance; however, this extension is already well structured within the current modeling framework and can be readily implemented by extending the existing Inrush and PFC models.

If in the abstract it was stated that

> *"to develop a highly computationally efficient digital model of a power electronic circuit, with high fidelity such that it is capable of reproducing the behavior of the analog system even at high frequencies, while using a low-cost microcontroller as the emulation platform."*

This objective can be considered achieved.

The proposed HIL board is therefore suitable for real-time, low-cost emulation for interested users. Moreover, the Model-Based Design approach is strongly recommended not only for those developing emulation systems, but also for those implementing microcontroller-based control strategies. In fact, through the Hardware Support Package, both design and implementation become significantly more user-friendly and streamlined.

Finally, I would like to thank the reader for their attention, and I hope that this thesis may be useful for future work in this field. For any questions, clarifications, or suggestions regarding related topics, I can be contacted at: `giulio.salbego@gmail.com`

Bibliography

[1] Martin Schlager, Wilfried Elmenreich, and Ingomar Wenzel. Interface design for hardware-in-the-loop simulation. In *IEEE International Symposium on Industrial Electronics (ISIE)*, 2006. doi: 10.1109/ISIE.2006.295703.

[2] Bin Lu, Xin Wu, Hernan Figueroa, and Antonello Monti. A low-cost real-time hardware-in-the-loop testing approach of power electronics controls. *IEEE Transactions on Industrial Electronics*, 54(2):919–931, 2007. doi: 10.1109/TIE.2007.892253.

[3] Handy Fortin Blanchette, Tarek Ould-Bachir, and Jean-Pierre David. A state-space modeling approach for the fpga-based real-time simulation of high switching frequency power converters. *IEEE Transactions on Industrial Electronics*, 59(12):4555–4567, 2012. doi: 10.1109/TIE.2011.2182021. URL `https://doi.org/10.1109/TIE.2011.2182021`.

[4] Typhoon HIL Inc. *HIL604 Real-Time Power Electronics Simulator Brochure*, 2026. URL `https://www.typhoon-hil.com/doc/products/Typhoon-HIL604-brochure.pdf`. Product brochure, FPGA-based real-time simulator.

[5] OPAL-RT Technologies. *OP5707XG Real-Time Simulator System Description*, 2026. URL `https://opalrt.atlassian.net/wiki/spaces/PHDGD/pages/1592950858/OP5707XG+System+Description`. Hybrid CPU-FPGA real-time simulation platform.

[6] Plexim GmbH. *PLECS RT Box 1 Product Overview*, 2026. URL `https://www.plexim.com/products/rt_box/rt_box_1`. FPGA-based real-time simulation platform.

[7] Speedgoat GmbH. *Performance Real-Time Target Machine*, 2026. URL `https://www.speedgoat.com/products-services/real-time-target-machines/performance-real-time-target-machine`. Simulink Real-Time CPU/FPGA platform.

[8] National Instruments. *PXIe-7822R R Series Multifunction RIO Device Datasheet*, 2026. URL `https://www.artisantg.com/info/National_Instruments_PXIe_`

7822_Datasheet_2018327133738.pdf. FPGA-based RIO platform for real-time control and HIL.

[9] STMicroelectronics. *STEVAL-HILE1G4K1: HIL cost effective emulation tool for digital power converters*, February 2026. URL https://www.st.com/resource/en/data_brief/steval-hile1g4k1.pdf.

[10] MathWorks. *STM32 Hardware Support from Simulink and Embedded Coder*. MathWorks, 2026. URL https://www.mathworks.com/hardware-support/stm32.html.

[11] MathWorks. *Model-Based Design*. MathWorks, 2026. URL https://www.mathworks.com/help/simulink/gs/model-based-design.html.

[12] MathWorks. *Simscape Documentation*. MathWorks, 2026. URL https://www.mathworks.com/help/simscape/index.html.

[13] STMicroelectronics. *SR5 E1 line of Stellar electrification MCUs – 32-bit Arm Cortex-M7 automotive MCU 2x cores, 300 MHz, 2 MB flash, rich analog, 104 ps 24-channel high-resolution timer, HSM, and ASIL D*. STMicroelectronics, rev. 8 edition, November 2025. URL https://www.st.com/resource/en/datasheet/sr5e1e3.pdf.

[14] Laszlo Huber, Yungtaek Jang, and Milan M. Jovanovic. Performance evaluation of bridgeless pfc boost rectifiers. *IEEE Transactions on Power Electronics*, 23(3): 1381–1390, 2008. doi: 10.1109/TPEL.2008.921107.

[15] STMicroelectronics. *Introduction to Low-Voltage Totem Pole Power Factor Correction on STM32 MCUs Using X-CUBE-DPOWER*. STMicroelectronics, October 2025.

[16] STMicroelectronics. *STEVAL-7BIDIRCB: 7 kW bidirectional AC-DC converter for ESS and industrial charger, full SiC-based*, October 2025. Data Brief, Rev. 4.

[17] Luka Stanić, Zeljko V. Despotović, Milan Pajnić, and Miodrag Skender. Digital control challenges in a single-phase ccm totem-pole pfc rectifier with gan devices. In *2023 IEEE Conference on Industrial Electronics and Applications*, pages 1–6, 2023. doi: 10.1109/EE59906.2023.10346108. URL https://ieeexplore.ieee.org/document/10346108/.

[18] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, 2 edition, 2007. ISBN 978-3-540-49857-1.

[19] R. D. Middlebrook and Slobodan Ćuk. A general unified approach to modelling

switching-converter power stages. *IEEE Power Electronics Papers (various proceedings / foundational work)*, 1977.

[20] STMicroelectronics. *STM32 Nucleo-64 boards (NUCLEO-XXXXRX, NUCLEO-XXXXRX-Q, NUCLEO-XXXXRX-P) Data Brief*. STMicroelectronics, February 2026.

[21] MathWorks. *Build and Run Executable on ARM Cortex-M Processors*. MathWorks, 2026. URL `https://www.mathworks.com/help/ecoder/armcortexm/ug/build-and-run-executable-on-arm-cortex-m-processors.html`.

List of Figures

List of Tables