An Open Processor-in-the-Loop Framework for Power Converter Control

Marco Guerreiro, Wesley Becker, Pedro dos Santos and Steven Liu
Chair of Control Systems, University of Kaiserslautern-Landau, Kaiserslautern, Germany
Corresponding author: marco.guerreiro@rptu.de

***Abstract*—Controllers for modern power converters are complex embedded systems. Processor-in-the-loop (PiL) schemes are one way to test and verify the behavior of the controller when executed on the target hardware. PiL schemes can be employed at low cost and can be used to verify if control algorithms are executed correctly and in real-time. Although PiL is often supported by power electronics simulation tools, the range of supported controllers is limited. This paper proposes a framework where PiL schemes can be implemented for a wide range of software tools and controllers. The framework is implemented as a C library with external interfaces that are made configurable and can be adapted to different devices and simulation tools. The C library is open-source and available online. To demonstrate the framework, a PiL scheme is implemented with PLECS as the software tool and a Zynq-7000 system-on-a-chip as the controller.**

***Index Terms*—power converters, control, processor-in-the-loop**

I. Introduction

As the current electric grid transitions to the next generation power grid, the role of power converters becomes evermore crucial [1], [2]. Power converters enable many of the main features of the future grid, such as distributed generation, bidirectional power flow and high-voltage DC (HVDC) transmission. These features, however, come at the cost of increasingly complex converter topologies, control, and operation. One example is the modular multilevel converter (MMC), a topology used for HVDC applications. Its operation requires several control loops, such as submodule voltage balancing, circulating current suppression, output power control and energy control [3]. Moreover, as grid-connected converters are operated following strict grid codes, advanced control strategies are required in order to ensure that the converter is able to quickly react to changes in setpoints or to reject disturbances. An example is model predictive control (MPC), an optimization-based technique that allows controlling multiple-input multiple-output systems while imposing constraints on system states and inputs [4]. For power converters, this ensures their operation within safe current and voltage levels, without distortions due to overmodulation.

One of the main challenges of developing such complex controllers is their implementation and verification on hardware. Control strategies such as MPC often require the execution of numerical methods in real-time. Because power converters have sampling periods in the sub millisecond range, implementing these controllers requires embedded systems with high computational capacity [5], [6]. In practice, these embedded controllers are realized through a combination of a digital-signal processor (DSP) with a field-programmable gate array (FPGA), or by using a system-on-a-chip (SoC) that integrates a multi-core processor and an FPGA on the same device. However, verifying the control algorithm on these controllers is challenging. In addition to verifying that the control algorithm is properly implemented, it is also necessary to verify that the FPGA and processor are properly integrated and that all timing requirements are met.

Although the controller could be verified in closed-loop with the actual power converter, this is not always possible or recommended. If the closed-loop system is unstable because the controller is malfunctioning, the power converter could be damaged. Moreover, running test cycles on the converter may be time consuming due to startup and shutdown procedures, and it might not be possible to test scenarios such as faults.

A popular approach to verify controllers is to use hardware-in-the-loop (HiL) [7], [8]. In HiL testing, a mixed-signal platform emulates the power converter in real-time, providing analog and digital signals that represent, for example, sensors and actuators. The embedded controller is then attached to the HiL platform, allowing it to be verified in a safe and controlled environment. After HiL testing, the controller could be directly deployed to the actual converter. Although HiL is an established and valuable tool, HiL testing in power electronics is not always possible or feasible. Models with a high number of switches, such as the MMC, or with many subsystems, such as microgrids, require expensive high-end HiL platforms, or it might not even be possible to emulate such models in real-time. However, these are cases that would benefit the most from verifying the implementation of the controller in a controlled closed-loop environment.

An alternative to HiL is processor-in-the-loop (PiL) testing [9], [10]. In PiL testing, the converter is simulated by a digital platform without real-time requirements, and the control loops are executed on the embedded controller. Measurements and control signals are exchanged between simulation and embedded controller through a communication channel. PiL testing is illustrated in Fig. 1, which shows that the power converter, measurements and actuators are represented in a simulation environment, and the control algorithm is implemented on hardware.

PiL testing has many features that are interesting for the development and implementation of embedded controllers for

 DOI 10.1109/IECON51785.2023.10312423

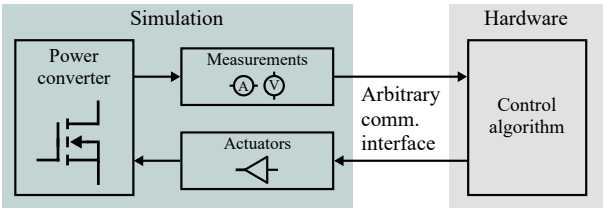


Fig. 1: Processor-in-the-loop scheme.

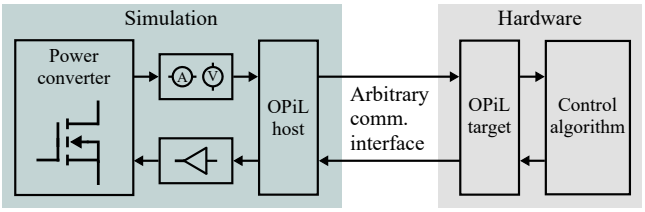


Fig. 2: Processor-in-the-loop with host and target modules.

power electronics. As with HiL, PiL testing provides a safe and controlled environment to verify the controller. However, unlike HiL, PiL testing has no real-time requirements. This allows to implement non-real-time controllers which can be then optimized to become real-time. This is especially important for the implementation and verification of complex control strategies such as MPC, where the implementation of numerical methods is often iterated on hardware in order to meet real-time constraints. Moreover, the range of converters and systems that can be simulated is larger than can be emulated with HiL platforms. Thus, the implementation of controllers for complex converters such as the MMC, or for large systems such as microgrids, can be verified. Another important feature of PiL is that it allows to verify the coordination between FPGAs and processors. If the control algorithm is implemented partially on the FPGA and partially on the processor, their integration can be verified with PiL testing.

The main disadvantage of PiL is that it cannot be used to verify input and output peripherals of the embedded controller (such as analog-to-digital converters (ADCs) and pulse-width modulators (PWMs)). After PiL testing, it is necessary to integrate the control algorithm and input and output peripherals, which requires additional testing. However, because of how PiL is implemented, there are clear boundaries between the control algorithm and the hardware peripherals, and their integration is simplified. This also simplifies the process of changing the embedded controller. For example, moving to a different DSP requires updating only the interface to the peripherals, while the control algorithm verified in PiL remains the same. Moreover, PiL testing can be deployed at much lower cost, since only a simulation software is required.

PiL schemes are a valuable tool for embedded control practitioners and researchers. The limiting factor in their use is support by software tools, which usually have a limited range of supported controllers. Moreover, previous proposals of PiL testing in the power electronics community have also been tied to specific software and controllers [9]–[11]. To overcome this issue, this paper proposes a framework that allows extending the use of PiL for a wide range of simulation tools and controllers. The proposed framework can be of great value for verifying the real-time implementation of complex control strategies for complex converter topologies. as well as for large systems. An open-source implementation of the framework using the C language is discussed, since executing C code is usually possible in simulations tools and embedded controllers. The implementation is available online [12].

To illustrate the framework, two examples are presented. The first example shows verification of an MPC controller for a grid-connected inverter, where the controller is partially implemented on an FPGA and on a processor, using an SoC. The second example shows verification of a high-level controller for a grid-connected nine-level MMC, highlighting the verification of an embedded controller for a complex converter topology. Next, Section II of this paper discusses the proposed framework, and the two examples are shown in Section III. Section IV concludes this paper.

II. OPiL: An Open Processor-in-the-Loop Framework

Fig. 1 illustrates the general PiL scheme. A simulation environment (host) simulates the power converter, signal acquisition and actuators. At each sampling period, measurements are sent to the embedded controller (target), which processes the measurements and sends back control signals.

These steps indicate several factors that need to be implemented in order for a PiL scheme to work. There must be an interface between the simulation and the controller, i.e. a communication medium, and the content and flow of data between the two parts has to be coordinated. Moreover, the controller and simulation must know what kind of information to expect from each other. The sequence of events is also important, i.e. the simulation sends measurements and waits for control signals, while the controller first waits for measurements and replies with control signals.

One way to achieve this coordination is through a component that is attached to both the simulation and controller. This is illustrated in Fig. 2, which shows the PiL scheme but the simulation and controller are augmented with OPiL host and target blocks. These blocks decouple simulation and controller, which is a key aspect that allows running PiL schemes for different software tools and controllers [13].

The operation of the OPiL blocks can be represented by the flowcharts indicated in Fig. 3. The host initially connects to the target and, while the simulation is running, gets measurement data from the simulation, sends it to the target, and waits for control signals. After receiving the control signals, the host updates the actuators on the simulation. This main process repeats at each sampling instant and lasts until the simulation ends. On the target side, the target initially waits for a connection from the host. Then, while the host is connected, the target receives simulation data, executes the control algorithm and sends the control signals back to the host. The next sections

 DOI 10.1109/IECON51785.2023.10312423

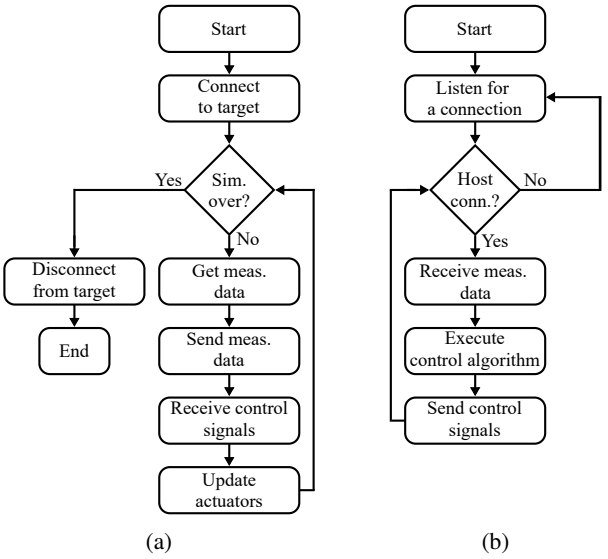


Fig. 3: Operational flowchart of (a) the host and (b) the target.

Meas. size N 4 bytes	Meas. N bytes	Sim. data size M 4 bytes	Sim. data M bytes

(a)

Control size K 4 bytes	Control signals K bytes	Con. data size L 4 bytes	Con. data L bytes

(b)

Fig. 4: Data packets sent by (a) the host and (b) the target.

discuss the data exchange protocol between the host and target, the design of the framework and its implementation.

A. Data exchange

The simulation and controller need to exchange two sets of data, namely measurements and control signals. However, it may be necessary to exchange and monitor signals that are not necessarily related to measurements or control. For example, the simulation may update reference signals on the controller, modify controller gains, etc. On the controller side, it is interesting to monitor data such as control execution time, fault flags and additional signals for debugging purposes.

The OPiL framework proposes a data exchange protocol between the simulation and controller that consists of four sets of data. The first two sets of data are the measurements and simulation data, and are sent from simulation to controller. The remaining two sets of data are the control signals and controller data, sent from controller to simulation. OPiL imposes no specific structure on these data sets. Instead, OPiL establishes only the format of the data packets, shown in Fig. 4.

B. Design

Fig. 2 suggests some of the interfaces that the OPiL host and target blocks require. The host block needs an interface to the simulation tool, such that it can get measurements and update actuators. Moreover, it needs access to a communication medium, such that it can send and receive data. In a similar way, the target block needs an interface to the control algorithm and to a communication medium.

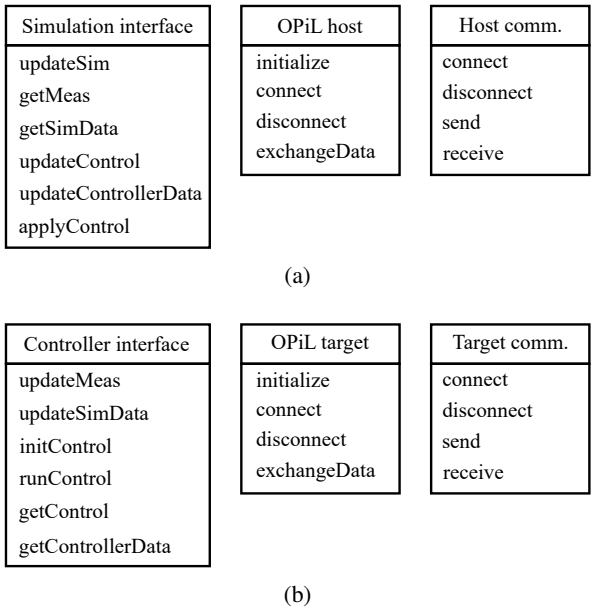


Fig. 5: Module representation of (a) the host and (b) the target.

This structure motivates the design of the OPiL framework based on modules. On the host side, three modules are identified. At the higher level, an OPiL host module is responsible for executing the steps illustrated in Fig. 3a. The execution of these steps is aided by two additional modules: a simulation interface and a communication module. The simulation interface provides functionalities such as getting data from the simulation, updating actuators, etc. The communication module provides functionalities such as connecting to the target, sending and receiving data, etc.

In a similar way, three modules can be identified on the target side. At a higher level, an OPiL target module executes the steps indicated in Fig. 3b. Two additional modules, a controller interface and a communication module, provide functionalities to the OPiL target module. The controller interface provides an interface to the control algorithm, while the communication module provides functionalities related to receiving a connection from the host, sending and receiving data, and detecting disconnection from the host.

Fig. 5 shows schematically the representation of the modules related to the host and target. The main advantage of this segmentation is that it is straightforward to adapt the framework to different software, controllers, and communication media. For example, using a different software environment requires implementing only the simulation interface, while the remaining modules remain unchanged.

C. Implementation

Since most of the embedded controllers are implemented using C language, and most software tools for power converter

 DOI 10.1109/IECON51785.2023.10312423

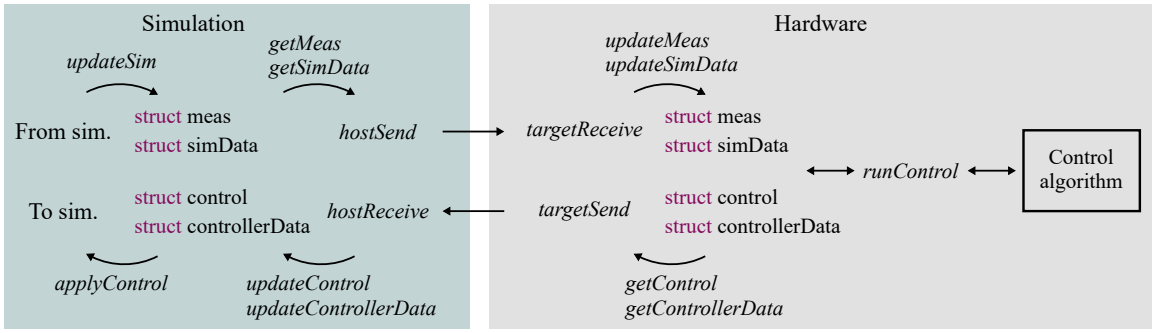


Fig. 6: Representation of the C implementation based on data structures.

control allow inclusion of blocks capable of executing C code, the C language was chosen to implement the OPiL framework.

The main concept behind the C implementation is based on operation on data structures, and is schematically shown in Fig. 6. Both the target and the host modules keep a copy of four data structures, which are related to the measurements, simulation data, control signals, and controller data. At each sampling step, the OPiL host library uses the *updateSim* function to get signals from the simulation environment and update the host measurements and simulation data structures. Next, these structures are sent to the target. Upon receiving the new data, the OPiL target library uses the functions *updateMeas* and *updateSimData* to update the measurements and simulation data structures on the target. With the updated measurements and simulation data, the control algorithm is executed and the control signals and controller data on the target are updated. After update, these two structures are sent to OPiL host, which uses the functions *updateControl* and *updateControllerData* to update these structures on the host. After update, OPiL host uses the functions *applyControl* to apply the control signals and controller data to the simulation.

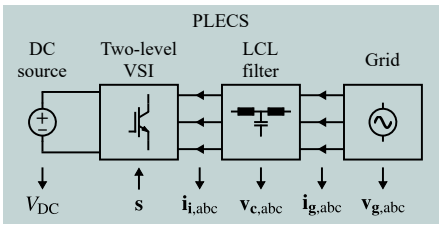
The implementation of the main core of the framework is available online [12].

III. Application Examples

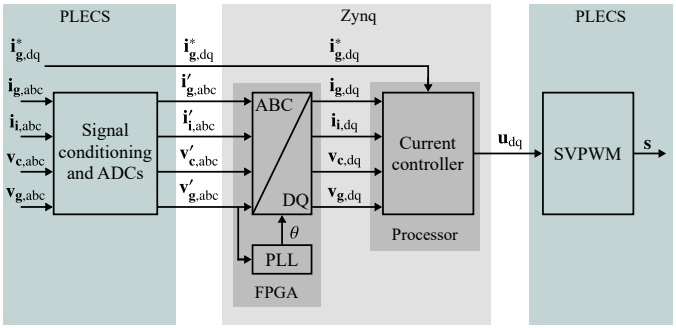
This section highlights the use of PiL testing for two cases. In both cases, PLECS running on Windows is used as simulation software, a Zynq-7000 SoC is used as embedded controller and socket programming with TCP/IP over Ethernet is used as the communication medium. The first example highlights the use of PiL testing to verify the implementation of a complex controller, whereas the second example highlights the use of PiL for a complex converter.

A. Simulation of a grid-connected three-phase inverter

This example highlights the use of PiL testing to implement a continuous-control-set MPC (CCS-MPC) controller for a grid-connected inverter. The setup is shown in Fig. 7. The inverter is connected to the grid through an LCL filter. PLECS simulates the diagram shown in Fig. 7a and, additionally, signal conditioning, ADCs, and a space-vector PWM (SVPWM) block, shown in Fig. 7b. The parameters used for the simulation are shown in Table I, and were based on [14].



(a)



(b)

Fig. 7: PiL scheme of grid-connected inverter highlighting (a) the power stage and (b) signal conditioning and control.

TABLE I: Parameters used for the simulation of the grid-connected inverter.

Parameter	Value	Parameter	Value
V_{DC}	650 V	L_i, R_i	3.4 mH, 0.1 Ω
V_{AC}	162.5 V	L_g, R_g	1.8 mH, 0.1 Ω
ω	$2\pi 50$ rad/s	C_f, R_c	20 µF, 0.06 Ω
f_s	7.5 kHz	ADC res.	12 bits

The CCS-MPC controller controls the grid-side current in the DQ frame, while imposing constraints on the inverter-side current and on the control signals. The overall structure of the control loop is shown in Fig. 7b, which shows that the DQ transform and the phased-locked loop (PLL) are executed in the FPGA, while the main CCS-MPC algorithm is executed in the processor. In this case, PiL testing allowed to verify several important aspects. It was possible to validate, in closed-loop, the correct behavior of the numerical method used to solve the MPC problem. It also allowed to verify that the DQ

 DOI 10.1109/IECON51785.2023.10312423

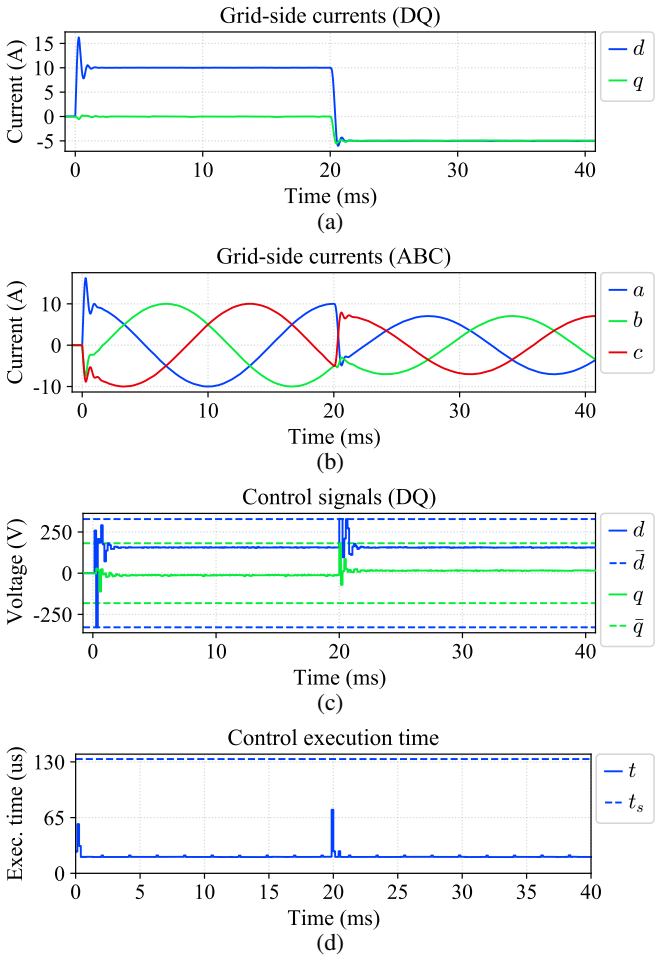


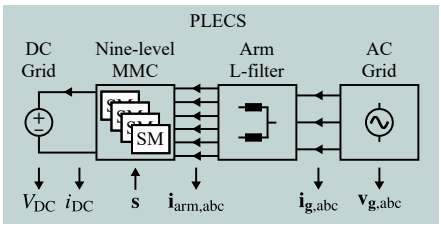
Fig. 8: Simulation results of the grid-connected inverter.

transform and PLL were correctly implemented on the FPGA, and that the processor and FPGA were properly integrated. More importantly, the PiL scheme allowed to tune the MPC algorithm such that it was possible to execute it in real-time. In this example, the prediction horizon of the MPC algorithm was set to four, while constraints were enforced only on the first two steps of the horizon. The design was based on the discussion in [15], [16].

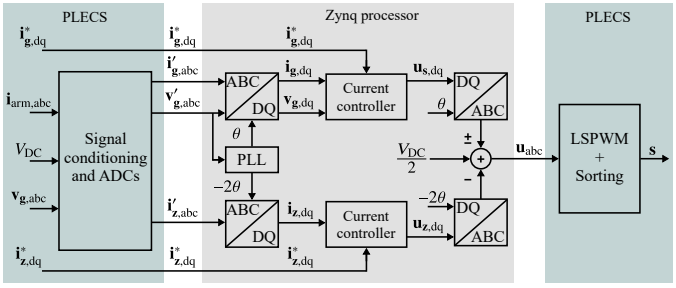
Fig. 8 shows results from a closed-loop simulation. Figs. 8a and 8b show that the controller is able to track the reference for the grid-side currents, which is initially set to $\mathbf{i}^*_{\text{g,dq}} = \begin{bmatrix}10 & 0\end{bmatrix}^\intercal$ A and changed to $\mathbf{i}^*_{\text{g,dq}} = \begin{bmatrix}-5 & -5\end{bmatrix}^\intercal$ A at 20 ms. Fig. 8c shows the control signals, which are within the predefined boundaries. Finally, Fig. 8d shows the execution time of the complete control loop, already including the interaction between FPGA and processor. The total execution time is well below the targeted sampling period t_s, which shows that the control algorithm meets real-time constraints.

B. Simulation of a medium voltage grid-connected MMC

This example illustrates the use of PiL testing for a medium voltage, nine-level MMC. Because of the high number of



(a)



(b)

Fig. 9: PiL scheme of a nine-level MMC highlighting (a) the power stage and (b) signal conditioning and control.

TABLE II: Parameters used for the simulation of the MMC.

Parameter	Value	Parameter	Value
V_{DC}	6 kV	L_{arm}, R_{arm}	3.19 mH, 47 mΩ
$V_{\text{AC-LL}}$	3.3 kV	C_{sm}, R_c	2.3 µF, 0.05 Ω
ω	2π50 rad/s	N_{sm}/Arm	8 Half-Bridge SM
f_s	2 kHz	ADC res.	16 bits

switches in the model, using HiL testing might not be possible or would be very expensive. However, PiL allows verifying a higher-level controller on the actual switched model of the converter. Moreover, using PiL testing further allows to verify the implemented control algorithm against harsh scenarios, such as faults or grid unbalances.

The PiL setup is shown in Fig. 9, and the parameters used for the simulation are shown in Table II. The grid-connected MMC, along with conditioning circuits, ADCs, low-level controllers and level-shifted PWM (LSPWM) are simulated in PLECS. The overall control loop consists of low-level and high-level controllers [17]. The low-level controller is responsible for PWM generation and for balancing the submodules (SMs). The high-level controller is responsible for tracking the desired current and energy references, and is implemented as in [18]. The controller was first simulated in PLECS, and then a code generation tool was used to generate the code that was implemented in the Zynq processor.

The simulation results are showin in Fig. 10. A current step is commanded at 1 s, when the active power reference changes from 0.1 MW to 0.5 MW. The references are well tracked and the submodule voltages remains balanced, proving a successful deployment of the generated C-code from the simulation model. Although, the generated code was only deployed to the processor, the overall method provides a quick validation in the controller hardware of the designed algorithm.

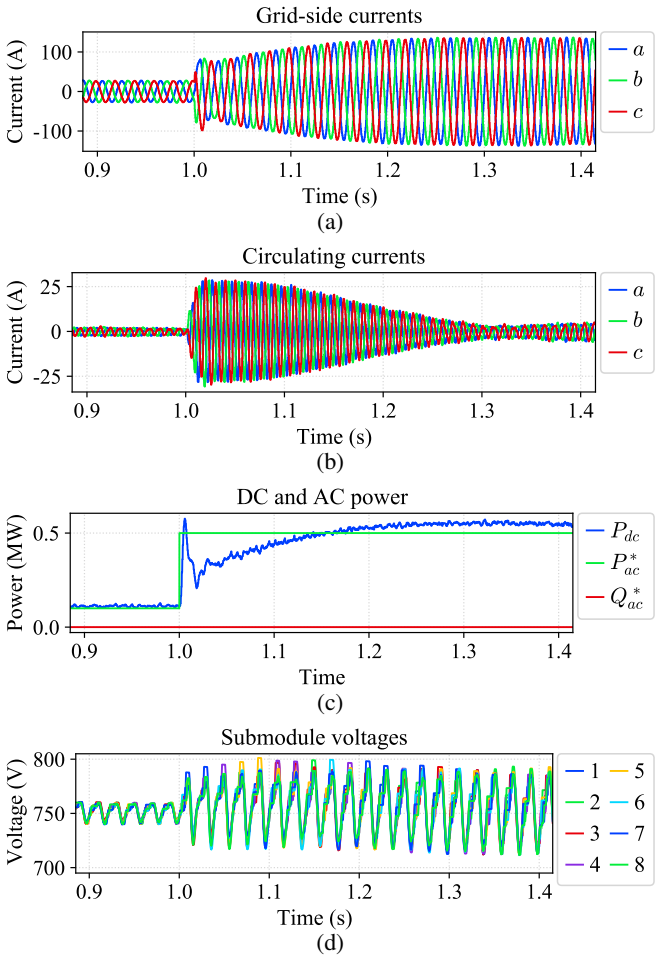


Fig. 10: Simulation results of the grid-connected MMC.

IV. Conclusions

This paper presented a framework that allows executing PiL schemes on a wide range of simulation tools and controllers. This is a valuable tool for the development of embedded controllers for power converters, since it allows verifying the controller in closed-loop without risks. This is especially useful when the controller combines devices such as an FPGA and a processor, because their integration can be also verified.

The framework discussed in this paper is based on the high level coordination of controller and simulation with the aid of low-level interfaces. Changing between software tools or controllers reduces to changing the corresponding interfaces, allowing to quickly adapt the framework to different cases. As an example, the framework is used to implement a PiL scheme where PLECS is used to simulate the power converter and a Zynq-7000 SoC, not supported by PLECS, is used as the controller.

References

[1] M. E. T. Souza Junior and L. C. G. Freitas, "Power electronics for modern sustainable power systems: Distributed generation, microgrids and smart grids – a review," *Sustainability*, vol. 14, no. 6, 2022.

[2] B. K. Bose, "Power electronics, smart grid, and renewable energy systems," *Proc. IEEE*, vol. 105, no. 11, pp. 2011–2018, 2017.

[3] J. Li, G. Konstantinou, H. R. Wickramasinghe, and J. Pou, "Operation and control methods of modular multilevel converters in unbalanced AC grids: A review," *IEEE J. Emerg. Sel. Top. Power Electron.*, vol. 7, no. 2, pp. 1258–1271, 2019.

[4] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young, "Model predictive control: MPC's role in the evolution of power electronics," *IEEE Ind. Electron. Mag.*, vol. 9, no. 4, pp. 8–21, 2015.

[5] M. Guerreiro, S. Kharade, P. Dos Santos, and S. Liu, "Power electronics control with system-on-a-chip-based platforms," in *2022 IEEE 20th Int. Power Electron. and Motion Control Conf. (PEMC)*, 2022, pp. 353–359.

[6] S. Wendel, A. Geiger, E. Liegmann, D. Arancibia, E. Durán, T. Kreppel, F. Rojas, F. Popp-Nowak, M. Diaz, A. Dietz, R. Kennel, and B. Wagner, "UltraZohm - a powerful real-time computation platform for MPC and multi-level inverters," in *IEEE PRECEDE*, 2019, pp. 1–6.

[7] F. Mihalič, M. Truntič, and A. Hren, "Hardware-in-the-loop simulations: A historical overview of engineering challenges," *Electronics*, vol. 11, no. 15, 2022.

[8] L.-A. Grégoire, K. Al-Haddad, and G. Nanjundaiah, "Hardware-in-the-loop (HIL) to reduce the development cost of power electronic converters," in *2010 IICPE*, 2011, pp. 1–6.

[9] J. Galarza, "Software and processor-in-the-loop execution for a grid connected modular multi-level converter," in *2019 IEEE XXVI INTERCON*, 2019, pp. 1–4.

[10] H. Vardhan, B. Akin, and H. Jin, "A low-cost, high-fidelity processor-in-the loop platform: For rapid prototyping of power electronics circuits and motor drives," *IEEE Power Electron. Mag.*, vol. 3, no. 2, pp. 18–28, 2016.

[11] J. Mina, Z. Flores, E. López, A. Pérez, and J.-H. Calleja, "Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in FPGA," in *2016 13th CIEP*, 2016, pp. 172–177.

[12] (2023) OPiL: an open processor-in-the-loop framework. [Online]. Available: https://gitlab.rhrk.uni-kl.de/lrs/opil

[13] P. Johnston. (2023) Mediator pattern. [Online]. Available: https://embeddedartistry.com/fieldmanual-terms/mediator-pattern/

[14] P. Falkowski and A. Sikorski, "Finite control set model predictive control for grid-connected AC–DC converters with LCL filter," *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 2844–2852, 2018.

[15] M. Guerreiro, P. Dos Santos, and S. Liu, "On the design of a model predictive controller for a DC-DC buck converter," in *EPE'23 ECCE Europe, to be published*, 2023.

[16] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[17] J. Ismail, Y. Wan, H. K. Iqbal, P. L. dos Santos, and S. Liu, "Experimental evaluation of a high power medium voltage converter for a DC grid connected agricultural machine," in *PCIM Europe digital days*, 2020, pp. 1–7.

[18] J. Pan, Z. Ke, M. Al Sabbagh, H. Li, K. A. Potty, W. Perdikakis, R. Na, J. Zhang, J. Wang, and L. Xu, "7-kV 1-MVA SiC-based modular multilevel converter prototype for medium-voltage electric machine drives," *IEEE Trans. Power Electron.*, vol. 35, no. 10, pp. 10 137–10 149, 2020.

 DOI 10.1109/IECON51785.2023.10312423