HOW2POWER TODAY
Your Power Design Newsletter

Exclusive Technology Feature

ISSUE: August 2021

# Developing A 25-kW SiC-Based Fast DC Charger (Part 5): Control Algorithms, Modulation Schemes And Feedback

*by Oriol Filló, Dionisis Voglitsis, Karol Rendek, Stefan Kosterec and Rachit Kumar, onsemi, Phoenix, Ariz.*

In parts 1 through 4 of this series,[1-4] we've shared and extensively described the development of a 25-kW EV charger from a hardware perspective. Fig. 1 represents the system discussed until now. However, this part 5 dives into a different dimension of the charger design as we explore and provide practical insights on the implementation of the control strategy and algorithms for such a system.

Rather than discussing control theory, our purpose is to provide firsthand details on the beneficial approach to control hardware and software development that the development team has taken, which helps speed up the firmware development and the validation process. This applies to both the state-machine on the ARM controller and to the main control algorithm on the FPGA, which we'll soon say more about.

At the same time, the particular development process described here ensures that errors are minimized and detected early on, even before prototype hardware becomes available or is being designed. In the following sections, we will describe the steps and tools (MathWorks and Xilinx) to implement such an approach, the state machine of and algorithmic blocks for the power factor correction (PFC), and the main algorithmic blocks of the DAB converter.

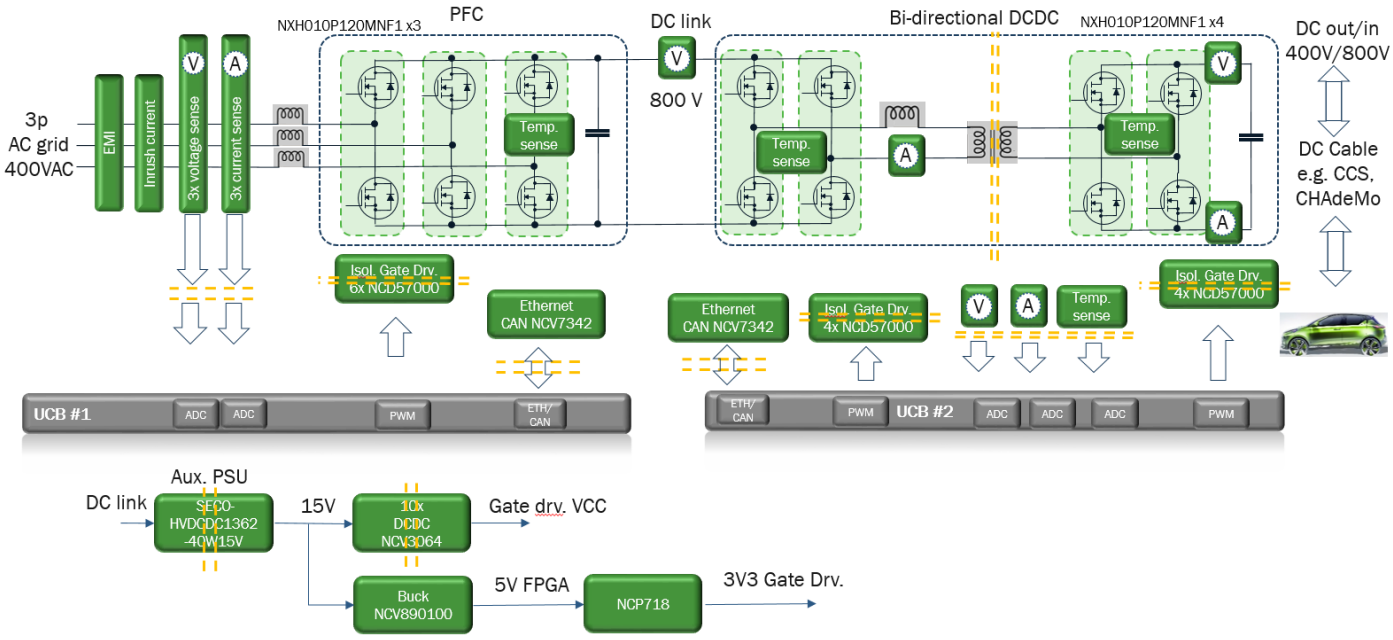


*Fig. 1. A high-level block diagram of the 25-kW EV dc charger.*

## Development Process For the Control Strategy

The overview architecture of the control software for the PFC is illustrated in Fig. 2. At the heart of this design is Xilinx's Zynq 7000 SoC, which contains both ARM cores and an FPGA core. The Zynq 7000 is mounted to the universal controller board (UCB), which also contains peripherals, ADCs, multiple memory boards and the necessary power-tree for the SoC and other components.[5]

First, the ARM cores run the state-machine, which is the higher-level routine in the firmware, as well as other ancillary tasks, such us communication protocols, protection functions, etc. Secondly, the FPGA serves as the powerhouse of the main control algorithm, running the control loops that drive the converter to handle the

power as desired and implement the ac-dc conversion, PFC and voltage boost to the desired dc link level. Thus, the "main algorithm" on the FPGA is a particular state—the steady state we could say—of the state-machine. The DAB converter has the same distribution of tasks between ARM cores and FPGA.

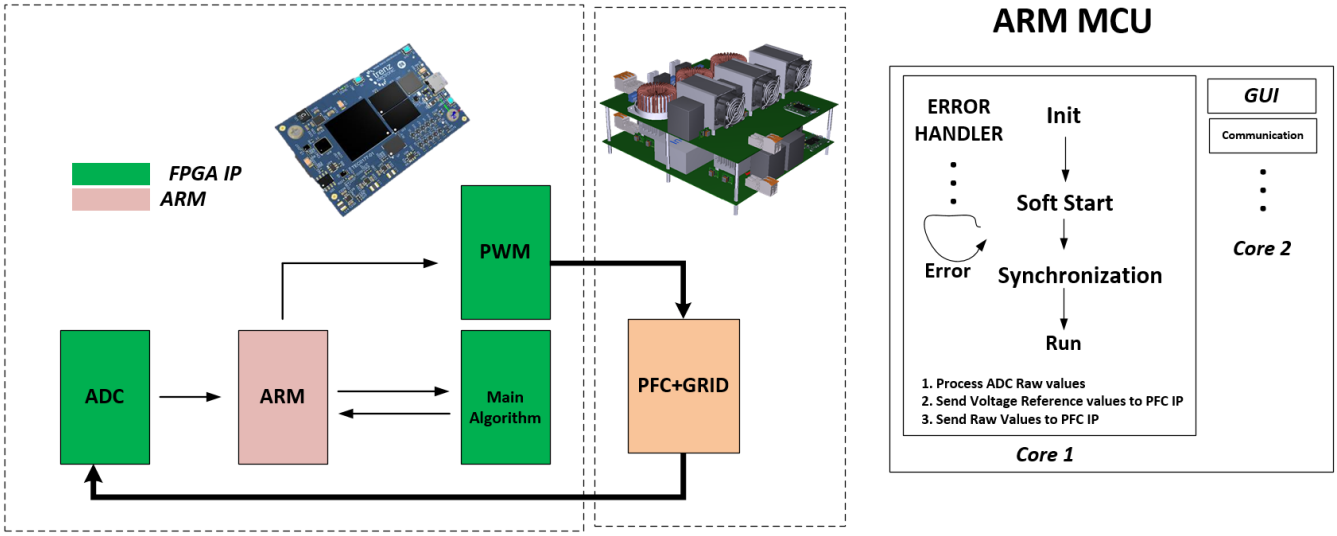


*Fig. 2. Overview of the 25-kW PFC converter control architecture. Tasks are distributed between the FPGA and ARM MCUs in the Xilinx Zynq 7000 on the UCB. The control architecture overview will be similar for the DAB converter.*

## Model-Based Testing To Unveil Errors In Control System

Fig. 3 illustrates a typical distribution of how errors arise and are detected along a project's development chain. One may notice that most of the errors are introduced in the early stages of the specification and design phase; however, the majority of them are discovered in the late stage of testing.

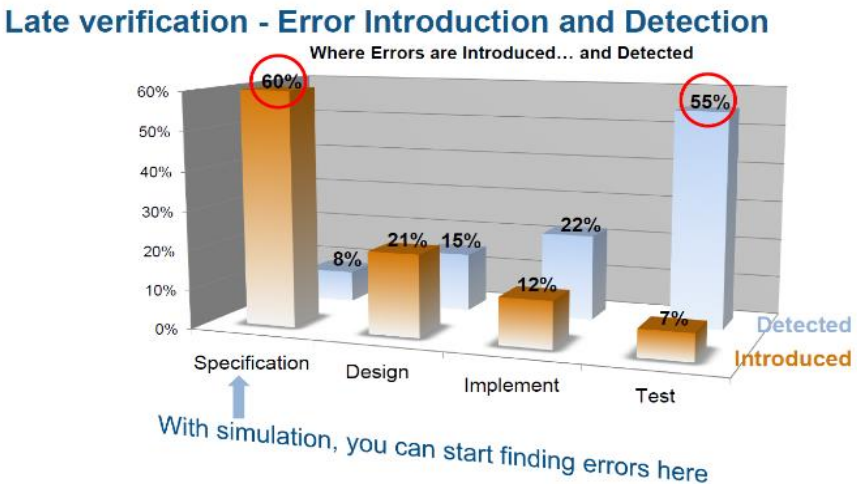


*Fig. 3. Errors introduced versus errors detected. (Source: Clive Maxfield and Kuhoo Goyal's book, "EDA: Where Electronics Begins.")*

In order to tackle the phenomena presented in Fig. 3, we adopted a development process that aims to detect the majority of these errors during the early stages of development. If well implemented, this approach brings a few advantages from a project resources and timeline perspective, such as

- Minimizing the risk of additional necessary hardware iterations.
- Optimizing the control system and converter performance to a great extent, before hardware is on hand.
- Accelerating the hardware-evaluation phase and minimizing the necessary tuning to be done on the hardware. Significant work will have already been carried out while prototype boards were in production.

To do that, the onsemi firmware and control engineers utilized a model-based testing methodology that leverages the MATLAB tools and ecosystem.[6] There are four key pillars underpinning the successful implementation of this approach and that developers need to address:

- Representative models that ensure a close match between the simulated and actual system response with viable simulation times. A similar tradeoff between model accuracy and simulation time as presented in part 3 for the power simulations of the PFC.
- Compilation and validation of our firmware C code (state-machine) within our simulation process and within our simulation model. Thus validation happens at the simulation stage—and not at the hardware evaluation stage.
- Automated synthesizable generation of FPGA IP cores from the verified models. This eliminates manual coding errors and allows high-level optimizations to minimize FPGA core area while meeting timing constraints.

To accelerate the implementation of these features, we capitalized on the advantages of the tools described in Table 1.

Table 1. Development and simulation tools used by the onsemi engineering team to develop, simulate, deploy and test the firmware for the 25-kW fast dc EV charger design.

| MathWorks tools<br><br>(Development, simulation and partial tuning of the complete control algorithm and state-machine. Automated generation of the FPGA core IPs) | Xilinx tools<br><br>(Deployment of the state-machine and control algorithm to the FPGA and ARM cores) |
|---|---|
| • *Simscape Electrical* to model and simulate electrical power systems<br>• *Simulink* for graphical modeling and simulation in the MATLAB environment<br>• *Fixed-Point Designer* to optimize conversion to fixed-point<br>• *HDL Coder* to generate synthesizable register-transfer level (RTL) from the model<br>• *HDL Verifier* to co-simulate the RTL back-to-back with the Simulink model<br>• *MATLAB Compiler* to compile handwritten C code for simulation in Simulink | • Vivado (Development Ecosystem for FPGA)<br>• Vitis (Integrated Development Environment ) |

**Taking A Step At A Time. How To Develop The Simulation Models?**

Fig. 4 depicts the simplified flow chart of the development and implementation process of the firmware, which is split into three main phases summarized in Table 2. Within the scope of this article, only the simulation model development is discussed in-depth, which is the most significant.

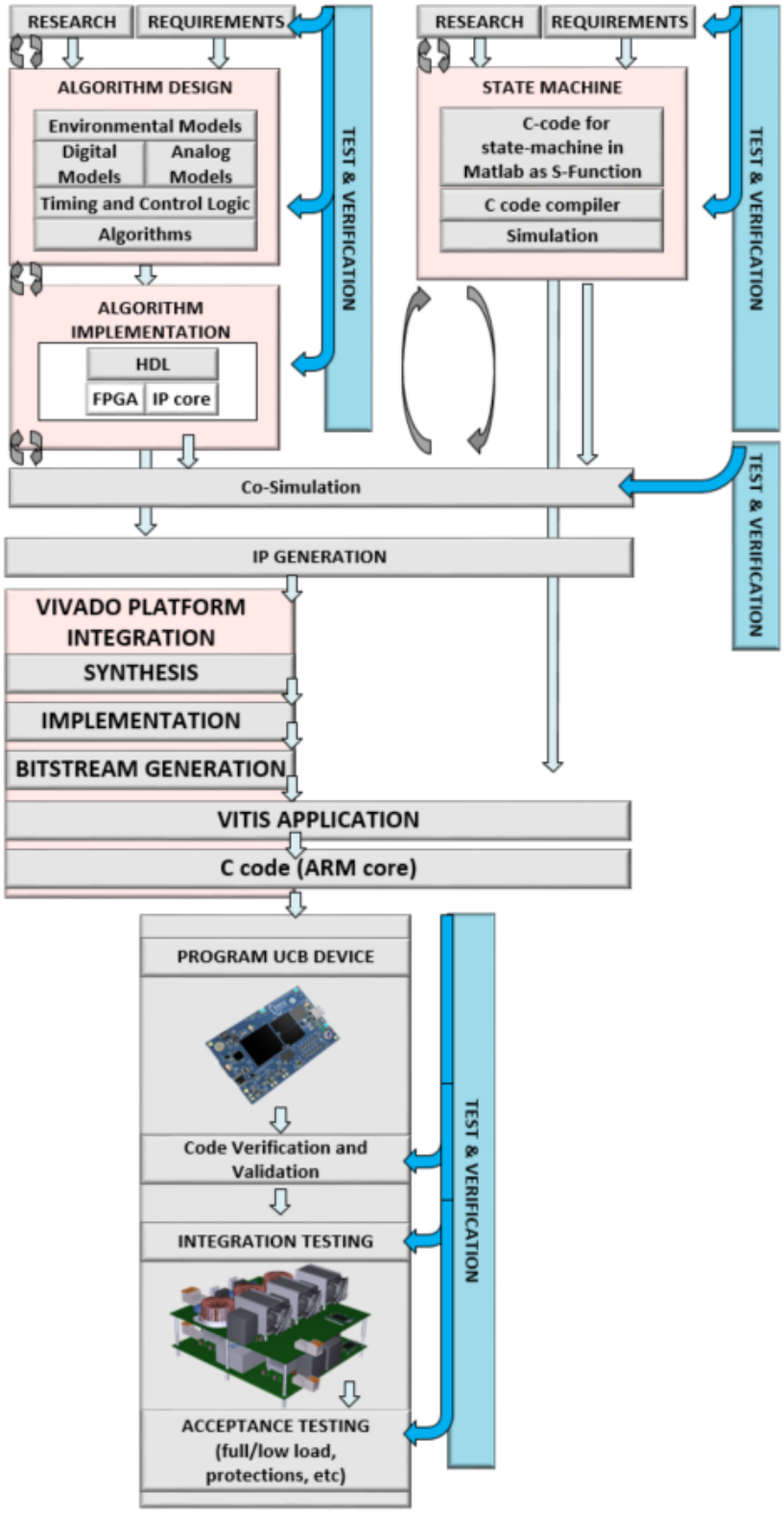


*Fig. 4. Firmware development flowchart for the 25-kW fast dc EV charger.*

Table 2. Phases of the firmware development process.

| Stage | Includes |
|---|---|
| Simulation model development | Main control algorithm<br>• C-code (state-machine) imported with S-function block<br>• Power converter block (hardware power model)<br>• Interface block (hardware ADC model)<br>• Ac and dc plant blocks |
| Deployment into Xilinx environment | • Integrates the FPGA IP generated in Mathworks<br>• Programming of the ARM core with the state-machine and required ancillary tasks. |
| Testing and verification | Verification of the model throughout the complete development process and within the multiple stages. At the end, deployment of the firmware on the hardware, bring up and evaluation of functionality and control requirements. |

The simulation model development stage encompasses the development of the simulation models (or simulation blocks) that will be used to verify the control algorithm of our system. The foremost blocks included in this project are:

- The C-code (state-machine) that will run on the ARM cores, imported for simulation via an S-function block
- Control algorithm of the converter (control loops)
- Power converter, which models the hardware
- Hardware interface, which models the ADC circuitry in the hardware
- Plant blocks, and ac plant for the PFC and a dc plant for the DAB.

At this stage of development we use "light" models (representative models that exclude fine details), which allow us to run multiple cases/scenarios under various conditions (grid impedances, current commands—depending on output-power-level variations—and other conditions) validating our controller response against many different scenarios. Switching models should therefore be avoided at this stage, as these are very detailed and take too much time to run—we learned that as well in part 3 of this series for the power simulations.

As an alternative, we are using switching average equivalent models,[7] which allows building simulation blocks with FPGA IP cores. At the same time, we preserve all the important/impactful features of the HW to ensure simulation integrity, such as the converter voltage-drop effect, noise measurement, PWM transport and analog to digital delays, etc.

**Steps Toward IP Generation Using MATLAB**

Diving into the details, this section explains the key steps to implement the particular simulation model and approach leveraging the capabilities of the MATLAB environment presented. Fig. 5 shows a simplified representation of a generic energy conversation system with the elements introduced in Table 1.

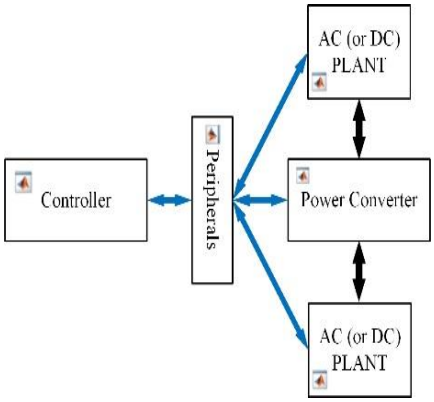


*Fig. 5. A simplified representation of a generic energy conversion system (not specifically the 25-kW dc charger).*

The "power converter" is the central element of the model (the representation of our hardware), and the "controller" is the main algorithmic block of interest and the one that we are developing and evaluating. Eventually this algorithmic block will be turned into an FPGA IP core itself by using the automated tools provided by the MATLAB simulation ecosystem.

Our team uses a series of six steps during the model development stage, which takes us all the way to the final IP generation. These steps are summarized in the simplified flow-chart in Fig. 6 and are briefly explained below.

Step 1: We develop our model in double-precision floating point, while we are using an average model for the power converter. As discussed in the previous section, at this stage, the developed models play an important role and should be as light as possible to allow reasonable simulation runtimes, but also accurate enough to reflect the actual behavior of the system.

Step 2: We are using automation tools provided by MATLAB to generate a fixed-point equivalent model of our system. The tool that we are using for this task is the MATLAB Fixed Point Designer.

Step 3: Having changed from double precision to a fixed-point one, we run a validation simulation, to ensure that fixed-point conversion has not affected the system's operational behavior.

Step 4: After validation, we include the state-machine that will be running in the ARM core of our UCB controller. The tool that allows simulating handwritten C-code in a Simulink model is the S-Function. At this point, we should be able to test our controller against numerous cases and under various conditions within reasonable simulation runtimes. Within this process, various important subtasks may take place. For example, the validation of proportional integral controller gains, the evaluation of the load-step response of the controller, the overcurrent reaction of the state-machine, error-handles, etc.

Step 5: Prior to the FPGA IP core generation, we highly recommend running a few simulations for selected cases/scenarios replacing the average model of the converter with the switching one. This process being time consuming should be repeated for very few simulation cases. It is, however, important to ensure that our controller is immune against the nonlinearities that are introduced by the switching behavior of the converter.

Step 6: Having reached the required confidence level for the developed algorithm, we can now use the automation tools for FPGA IP cores generation. This process significantly reduces programming errors, and achieves synthesizable and area-optimized RTLs that meet the timing constraints.

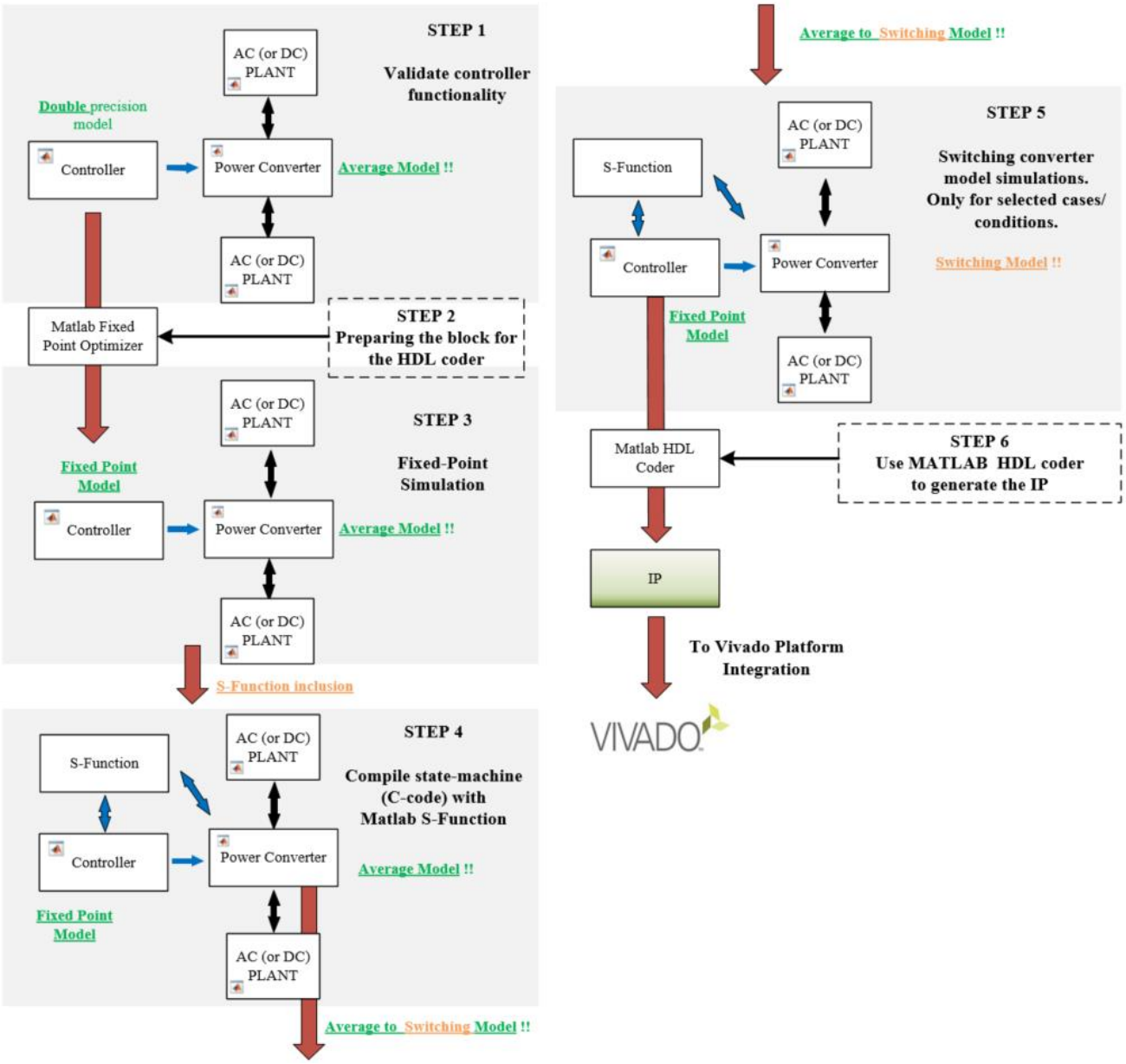


*Fig. 6. Flow chart of the six steps in the simulation model development stage. For ease of representation, the "peripherals" block in Fig. 5 has been omitted from this flow chart. It would be present in the same position and connections with the other blocks as in Fig. 5.*

## *PFC Control Strategy: State Machine And Control Loops*

This section will elaborate on the control strategy of the PFC, including the state-machine as well as the control algorithm (control loops). The state machine runs on the ARM cores of the UCB, and the control algorithm is run during the "DC BUS VOLTAGE_CONTROL" state of the state-machine, and implemented on the FPGA silicon.

In upcoming sections, more details are provided of both the state-machine and algorithmic functionality. Fig. 7 provides an overview of the PFC state-machine, with the "DC BUS VOLTAGE_CONTROL" state highlighted in green, where the control loops and FPGA would take over the control and run the main algorithmic functionality.

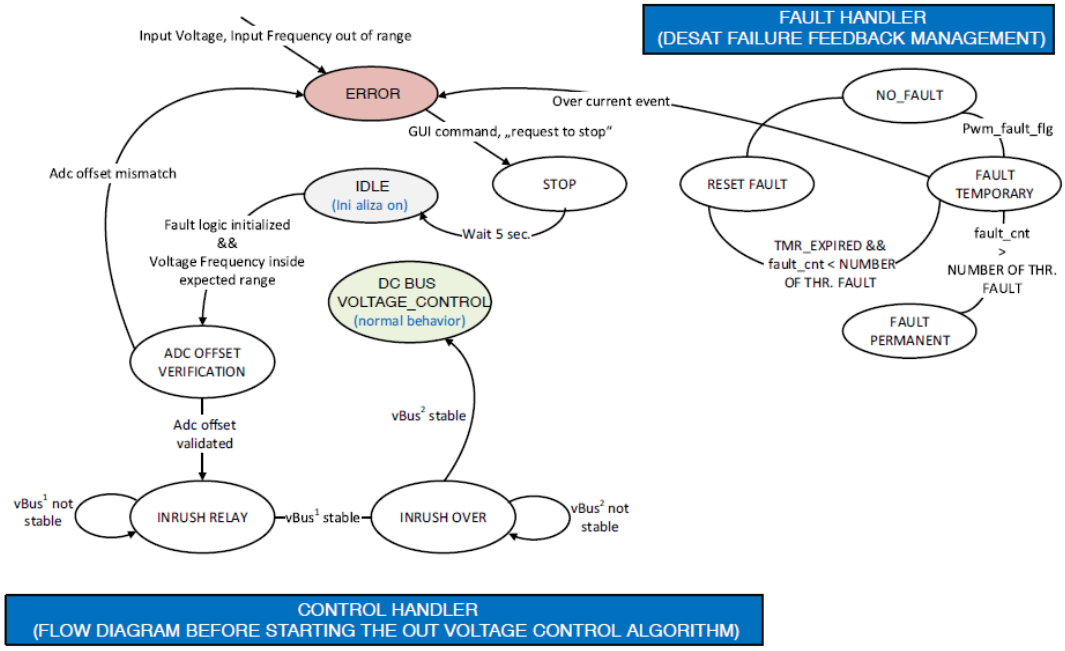


*Fig. 7. Overview of the state machine for the PFC converter.*

When a three-phase voltage at 50 Hz is provided to the input connector of the charger, the output bus capacitor voltage will rise because of the nature of the PFC topology. A bridgeless PFC with MOSFETs guarantees a current path from input to output because of the parasitic freewheeling diode present on each MOSFET.

When the MOSFETs are all off, the board simplifies to a three-phase diode bridge. The rectified input ac voltage will set to a defined level depending on the supply voltage amplitude and on the forward voltage of the MOSFET body diodes. However it is expected that at least a minimum ac voltage is provided at the input. For this reason, a resistor on two different lines serves as inrush current limiter.

Once the bus voltage reaches 230 V, the main ancillary power supply starts to operate. This power supply along with a series of dc-dc regulators generate the other voltage levels needed for powering up the digital and analog circuitry. More details about the PFC functionality can be found in onsemi's AND9957/D application note for an on-board charger PFC converter[8] as the same strategy has been implemented in this 25-kW dc charger project.

## PFC State-Machine Implementation On The ARM Core

As described above, the state-machine of the PFC runs on the ARM cores of the UCB. Its sequence begins with the IDLE block illustrated in Fig. 7, moving into the verification of the offset voltages in the ADC channels and the monitoring and sensing of the input voltages. These are used to determine the frequency and the angle of the phase of the three voltages. The angle will be the reference of the system to achieve power factor correction.

When the dc bus voltage becomes flat and stabilized, the PFC controller issues a command to the relay to bypass the inrush resistor and allow a further boost in the output bus voltage. However, the voltage increment is going to be something lower than the rectified input voltage amplitude, $\sqrt{6}\cdot V_{PH}$ rms. The PFC controller will wait until the bus voltage becomes flat again in order to start controlling the bus voltage to reach the targeted value of 800 V. The target value does not change in a single step, it follows a smooth ramp generator that brings up the bus value following a parametrized slope to the final 800 V.

The PFC implements only one hardware protection, against overcurrent events leveraging the DESAT functionality of the NCD57000DWR2G gate drivers. However, the DESAT hardware protection can combine with the software protection to generate a single input to a NAND gate, which provides a hardware stop to the PWM generation.

A reset of the failure conditions is possible only via a reset command sent through a GUI or via a power-down/power-up sequence, which represents a HW/SW reset. More details about the PFC functionality are found in reference 8 as it describes the same strategy implemented for this 25-kW dc charger project.

## PFC Main Algorithm And Control Loops On The FPGA

Fig. 8 illustrates the PFC control block as part of the complete simulation model. The PFC algorithm utilizes seven inputs and three outputs (summarized in Table 3). As part of the project, we will run and test different modulation strategies to assess which one produces better results, in terms of efficiency and harmonic distortion. This control strategy is the same as described in reference 8.

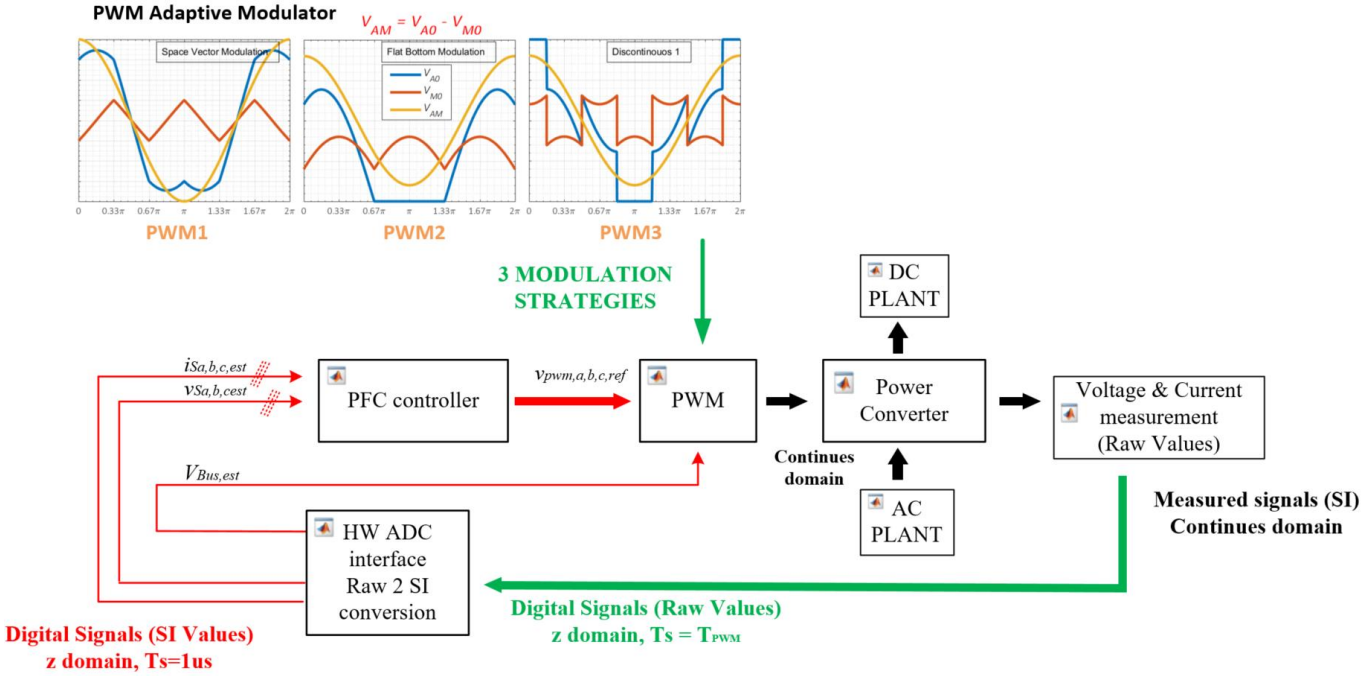


*Fig. 8. High-level block diagram of the PFC control algorithm.*

Table 3. Input and output parameters of the PFC control algorithm.

| PFC control algorithm | |
|---|---|
| Inputs (sampled quantities) | $V_{PHASE}$ (x3) |
| | $V_{LINE}$ (x3) – as no neutral point is available |
| | $V_{DC\_BUS}$ (x1) |
| Outputs | $V_{REF}$ (x3) |

Taking a deeper look, Fig. 9 shows in detail the blocks and relationships that compose the PFC algorithm. The $V_{LINE}$ voltages serve to determine the actual position of the ac voltage phasor. The angle, theta, is then used for regulating the current phase delay to 0° which is the main target of the PFC. The voltage position is used for switching from the stationary ABC system reference into the rotating DQ frame by means of Clarke and Park transformation (for the PFC, D axes means the amplitude of the phase voltage phasor).

As the angle theta is known, all the electrical quantities can be then expressed in the DQ system; such simplification will ensure the usage of simple proportional integral (PI) regulators. The gain tuning of the PIs depends on the transfer function of the plant to be regulated. PI regulators indeed can effectively regulate an error to zero when a constant is provided as reference quantity, while these regulators are not capable of regulating ac reference quantities.

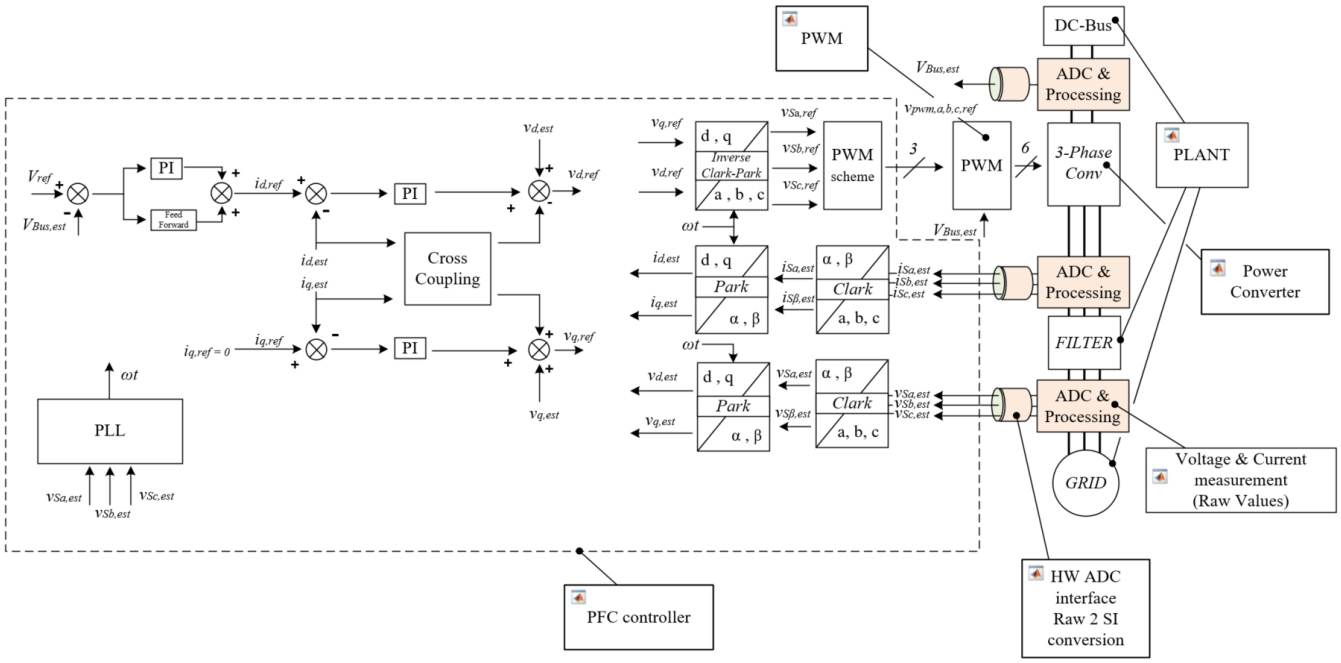


*Fig. 9. Detailed block diagram of the PFC control algorithm.*

In any case, the PI regulators need some sort of calibration to ensure a proper system stability. It is typical to expect fast response for the current loop (internal) and slower response on the external loop (voltage). At this point it is worth noting that the current control loop is running synchronously with the PWM. The synchronization procedure ensures that the ADC peripheral can be triggered at the exact time instance of the PWM carrier to ensure the natural filtering of switching ripple in the measured current quantities.

A side note is that the PWM frequency is not completely independent of the control frequency, because of the intrinsic ADC measuring delay, which should be small enough to guarantee the in-time execution of the PFC algorithm within the switching period. As the FPGA PFC controller latency is very low, around 150 nsec, the main limiting factor for the PWM frequency is the ADC sampling and conversion time. Once the ADC quantities are available, the control implementation is straightforward.

The main functionality of the PFC has been extensively tested with MATLAB, as described in the section "Steps Toward IP Generation Using MATLAB". The main Simulink model used is shown in Fig. 10 (the only part missing from this model is the S-function to test the state machine of the firmware). The blocks used are explained within the figure.

Note that the model at this stage consists mainly of Simulink blocks, including the average model of the three-phase power converter. The grid and the interconnection filter of the PFC utilize physical models in the

Simscape Electrical library, while the dc load and the capacitor (dc-plant) are modeled with the help of Laplace Simulink blocks. The model being light, allows reasonable simulation time using conventional laptops, achieving simulations of 0.1 sec in less than 1 min.

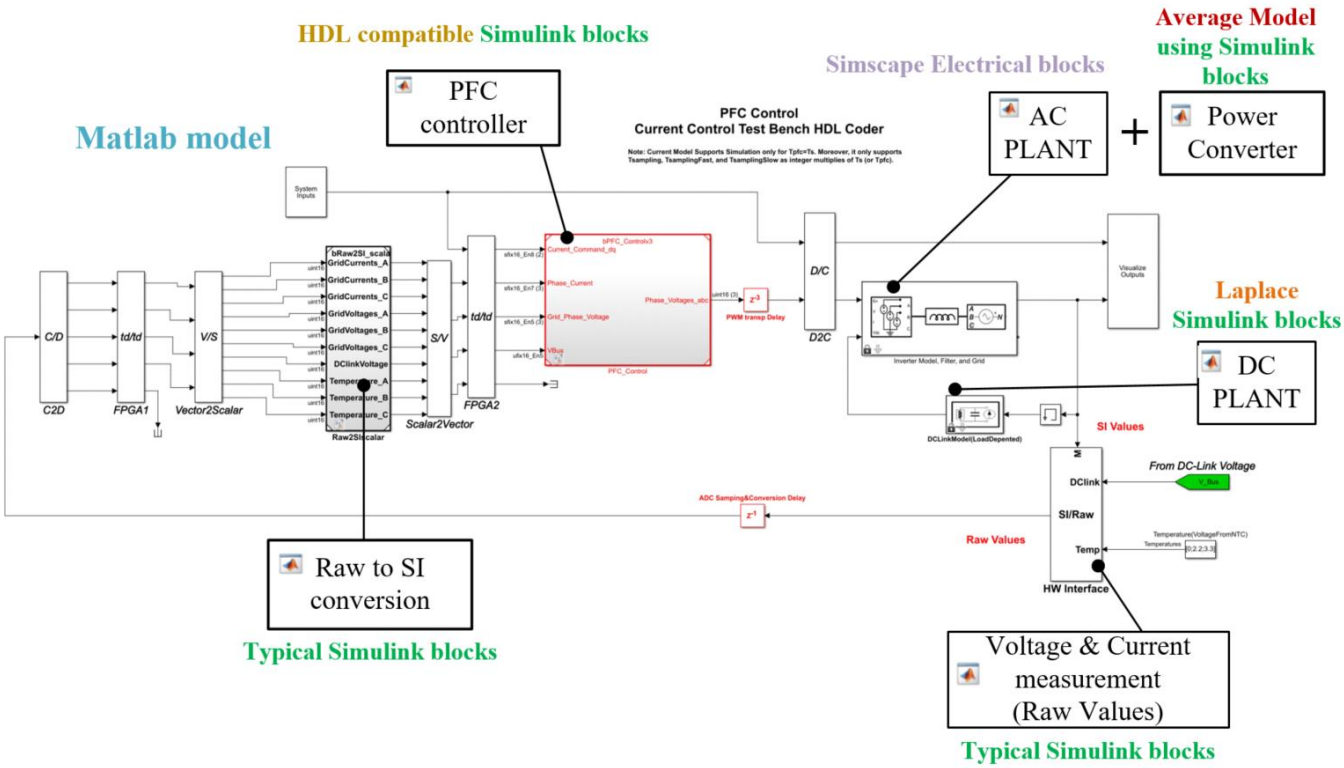


*Fig. 10. Main PFC controller Simulink model. The dc plant block (simple resistor and capacitor) is used as a load to test the PFC algorithmic functionality, and does not represent a model for the actual DAB converter.*

## *DAB Converter Control Strategy And Flux-Balancing Technique*

The implementation of the control strategy for the DAB converter follows a process similar to that of the PFC. In this section, we'll discuss the control algorithm for the converter as well as the flux-balancing technique. At the time of this writing, the Simulik model of the converter needs to be reworked to be prepared for the HDL Coder and the average models for the DAB are not finalized (we are in Step 6 in Fig. 4).

Starting off with the control algorithm, among the control techniques that may be applied, perhaps the most well-known are the fixed-frequency phase-shift techniques. Fig. 11 presents a classification of those techniques, with single phase-shift (SPS) being the simplest one. The simplicity of the controller is in fact the main benefit of this technique, but it comes at the cost of an increased current circulation in the converter and a tighter operating range wherein zero voltage switching (ZVS) is possible. These two downsides might certainly impact the efficiency of the system.

Two alternatives which are based on the SPS, the extended phase-shift (EPS) and dual phase-shift (DPS) techniques, achieve a better utilization of the converter, reducing the circulating currents and extending ZVS operation. But the penalty for these improvements is the additional control complexity that is added to the system.

Lastly, the triple phase-shift (TPS) technique is the unified version of the SPS, EPS and DPS. From this perspective, the SPS, EPS and DPS can be all derived from the TPS, and as such, are considered as special cases or subcases of TPS. Figs. 12-14 illustrate the operational principle of the SPS, EPS, and DPS respectively.

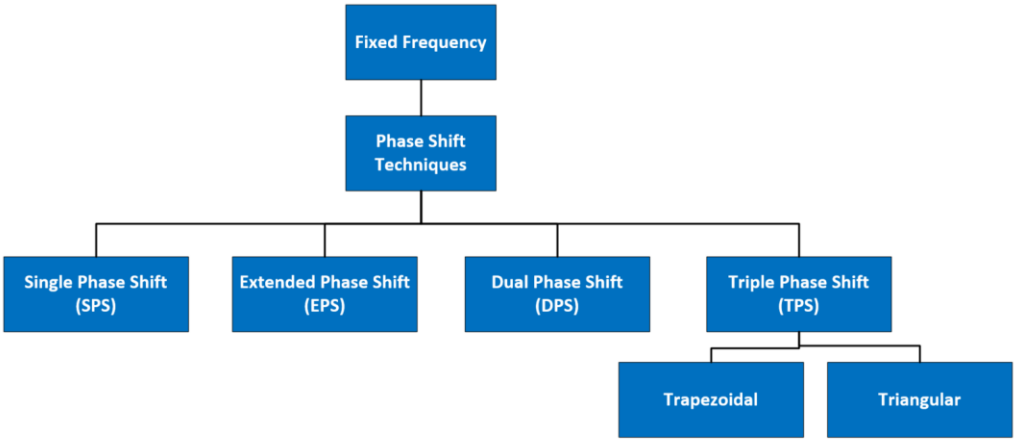


*Fig. 11. Classification of different phase-shift techniques. Triple phase-shift (TPS) is a unified version of the other techniques, and each of these can be considered as a subcase of TPS.*

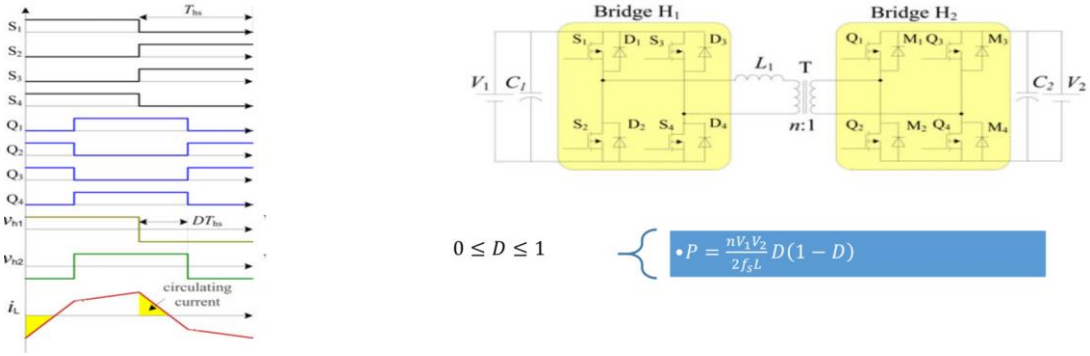


*Fig. 12. Single phase-shift (SPS) technique. (Source: "Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System"[9])*

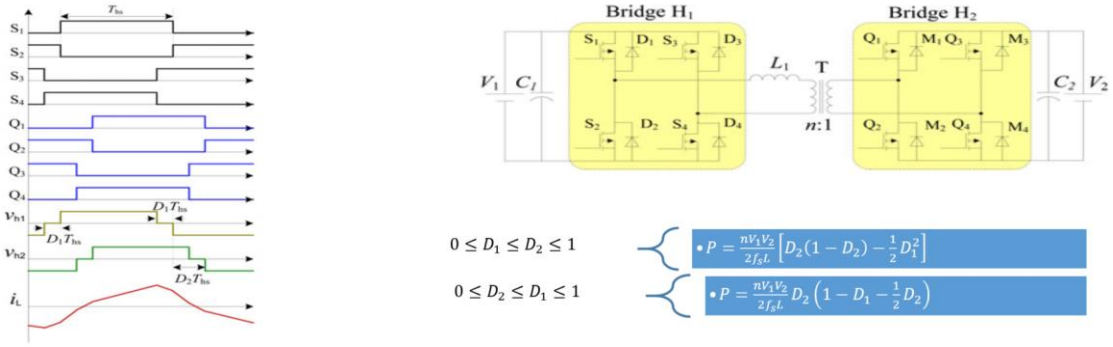


*Fig 13. Dual phase-shift (DPS) technique. (Source: "Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System"[9])*

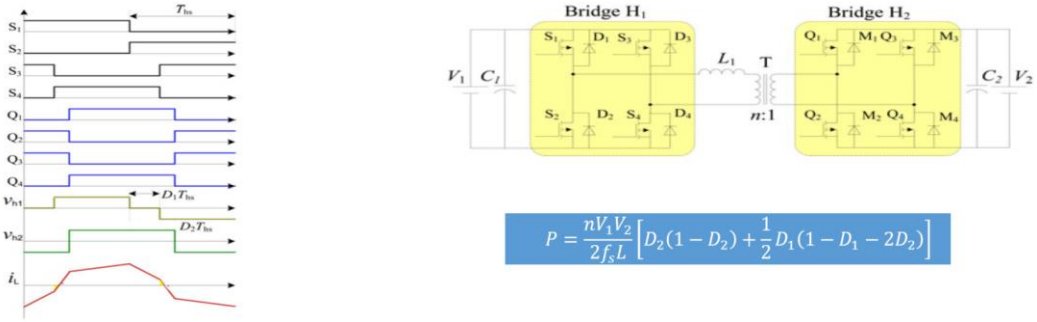


*Fig. 14. Extended phase-shift (EPS) technique. (Source: "Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System"[9])*

In part 4 of the series, it was noted that the power simulations for the DAB converter had been carried out with SPS. Later, we will test the more advanced techniques on the hardware prototypes and assess the possible benefits of each one.

The foremost possible improvement could be the higher efficiency of the converter. Furthermore, a reduction of the magnetizing peak current ($I_M$) in the transformer might also be possible, which could allow for the use of more compact transformer and inductor.

## Control Algorithm And Flux-Balancing Model Blocks

The control principle of the DAB is depicted in Fig. 15. The goal of the controller is to generate the desired output voltage or current for the battery.

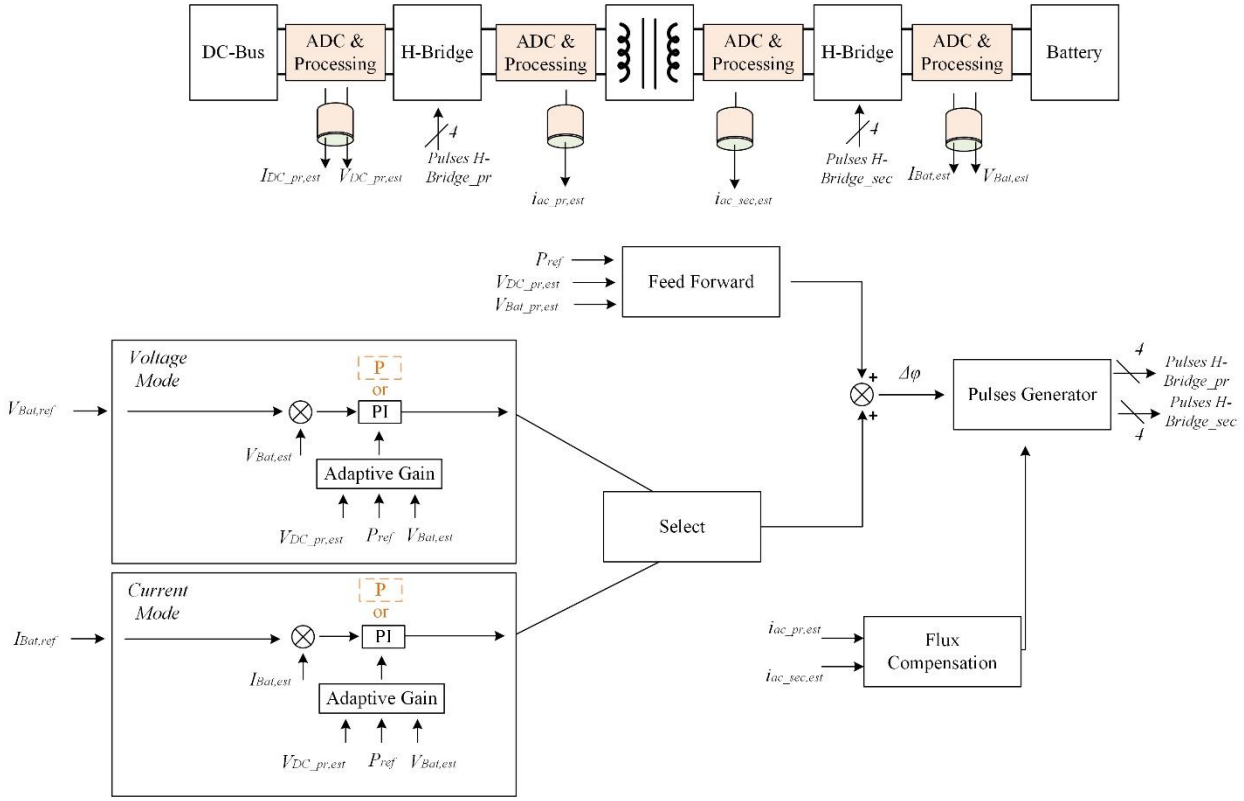


*Fig. 15. Block diagram of the DAB control algorithm. The transformer flux-balancing algorithm is included in it as well.*

The basic concept is simple: The measured output voltage (or current) and the target value, are both fed into a PI controller. The PI controller output tries to eliminate the error between them by generating the required ΔΦ, angle phase difference between the primary ac- voltage and secondary ac-voltage of the DAB, to drive the PWM of the primary and secondary sides. The control loop is slow because of the output capacitor, however considering the slow dynamics of battery charging, this isn't a problem.

A side point is the importance of adaptive PI gains to compensate for the steep Vout/ΔΦ slope. A P-only controller, (as opposed to PI) might be used as an alternative. On this front however, the engineering team needs to carry out further investigations.

An interesting part of the DAB control algorithm that's worth elaborating on is the flux-balancing feature. This technique, which was introduced in part 4, compensates any dc-component in the converter transformer, preventing the accumulation of magnetizing currents and the saturation of the core.

Fig. 16 shows the Simulink model used to realize the flux-compensation concept in the 25-kW DAB transformer. The block has three inputs and a single output. Primary and secondary transformer currents and synchronization (sync.) pulses are the inputs to the block. The output of the block is used to adjust the duty cycle of the PWM on the primary side of the transformer.

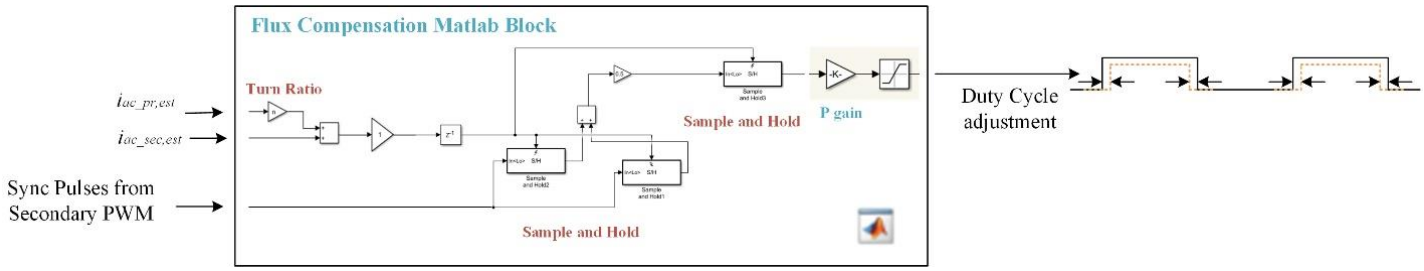


*Fig. 16. Flux compensation block diagram.*

The flux-balancing block is triggered by the transformer's secondary PWM sync. pulses (at the switching frequency of the converter), implying that the required sampling frequency of the ac currents fed into the block should be (at least) double the switching frequency of the converter. In particular, the sync. pulses are those of the PWM that drives the high-side switch of the first leg on the secondary side. The magnetizing peak-to-peak current of the DAB transformer is then computed with the aid of the sampled input currents using simple computations.

Subsequently, a sample-and-hold (S&H) circuit, triggered by the same sync. pulses, computes the switching average of the replicated magnetizing current. Finally, the estimated magnetizing average current is fed to a P controller, which will generate the command to adjust the duty cycle of the primary side PWM. Fig. 17 demonstrates the functionality of the flux-balancing algorithm implemented in the simulations.

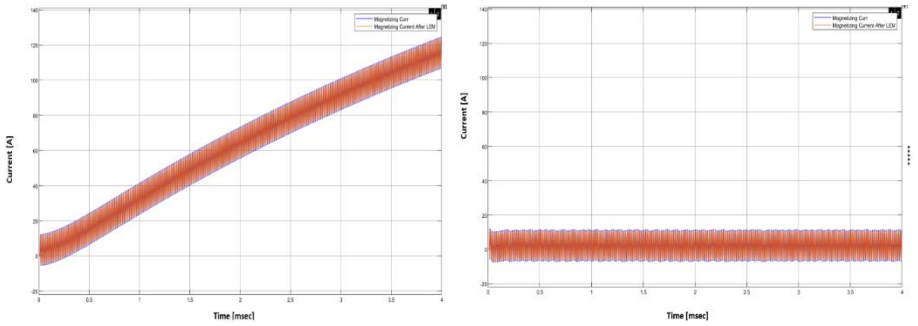


*Fig. 17. Simulated transformer magnetizing current ($I_M$) with the flux-compensation algorithm inactive (left) and active (right) of the DAB. When no flux-balancing technique is running, the residual dc current is building up during each switching cycle, which will eventually saturate the core.*

A critical aspect for an effective implementation of the flux-balancing technique is the required current acquisition bandwidth. As described above, the current to be measured and acted on is switching at 100 kHz, therefore the system should be capable of measuring at 200 kHz at least. Therefore, it is worth running a simulation in order to ensure that the selected current sensor will not introduce significant measuring errors that would undermine the flux compensation implementation.

The selected current sensors (LEM) specify 300 kHz of bandwidth. It must be considered that as the sampling frequency approaches 300 kHz, a decay in the gain appears and as in any acquisition system, phase lags might arise. Hence, although 300 kHz might seem to provide enough room at first glance, running simulations is recommended. The sampling current with and without the limited LEM bandwidth is illustrated in Figs. 18 and 19. (Note that in this example, we have not yet activated the flux-compensation, therefore the magnetizing current grows very large.)

In Fig. 19 a very small error both in magnitude and phase can be observed, but it is almost negligible. The dual-sampling methodology (current is measured two times per switching period) included in the algorithm might also have contributed to mitigating the error. In any case, we've already seen in Fig. 17 that the flux-balancing was working properly.

The simulation presented below should be run in advance of or in parallel with the results presented in Fig. 17. As such, a conventional LEM sensor with a bandwidth of around 300 kHz can be used. Fig. 20 illustrates the estimated switching average current, the actual magnetizing current ($I_M$), as well as the synchronization pulses.

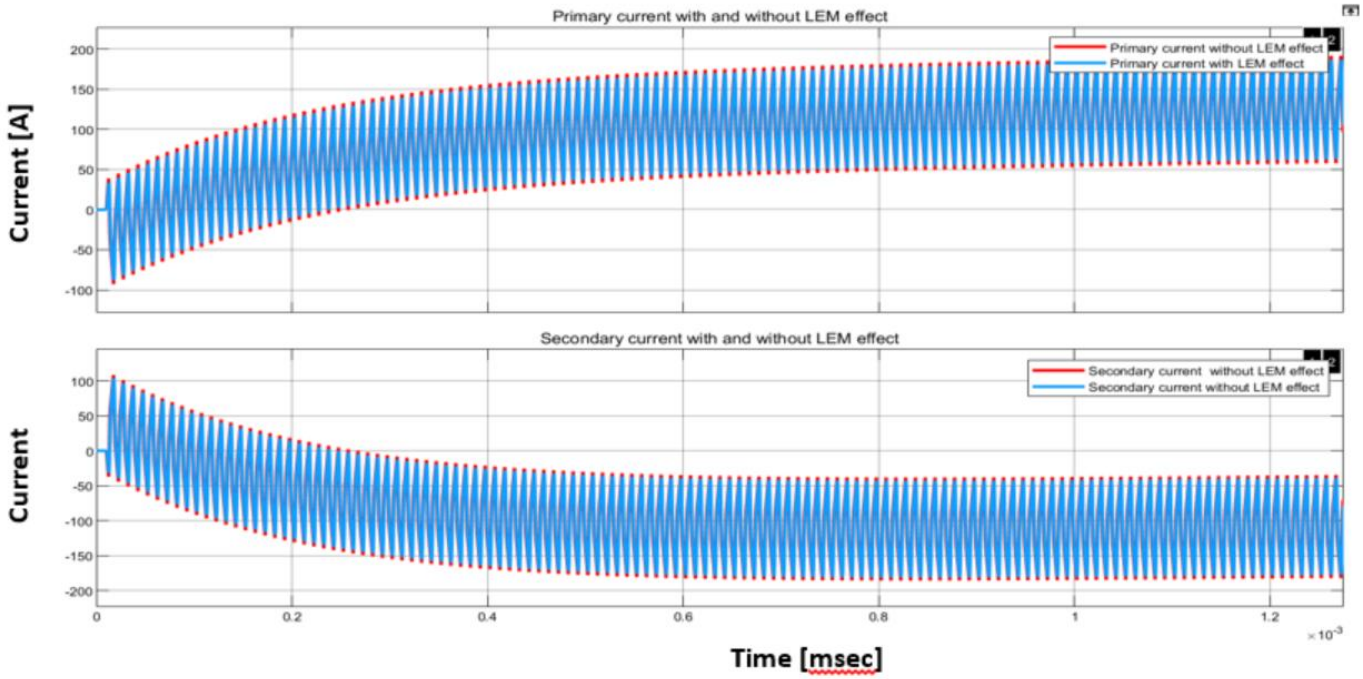


*Fig. 18. Primary and secondary current measurement with and without the LEM current sensor effect. The flux-balancing algorithm was not active in this simulation.*

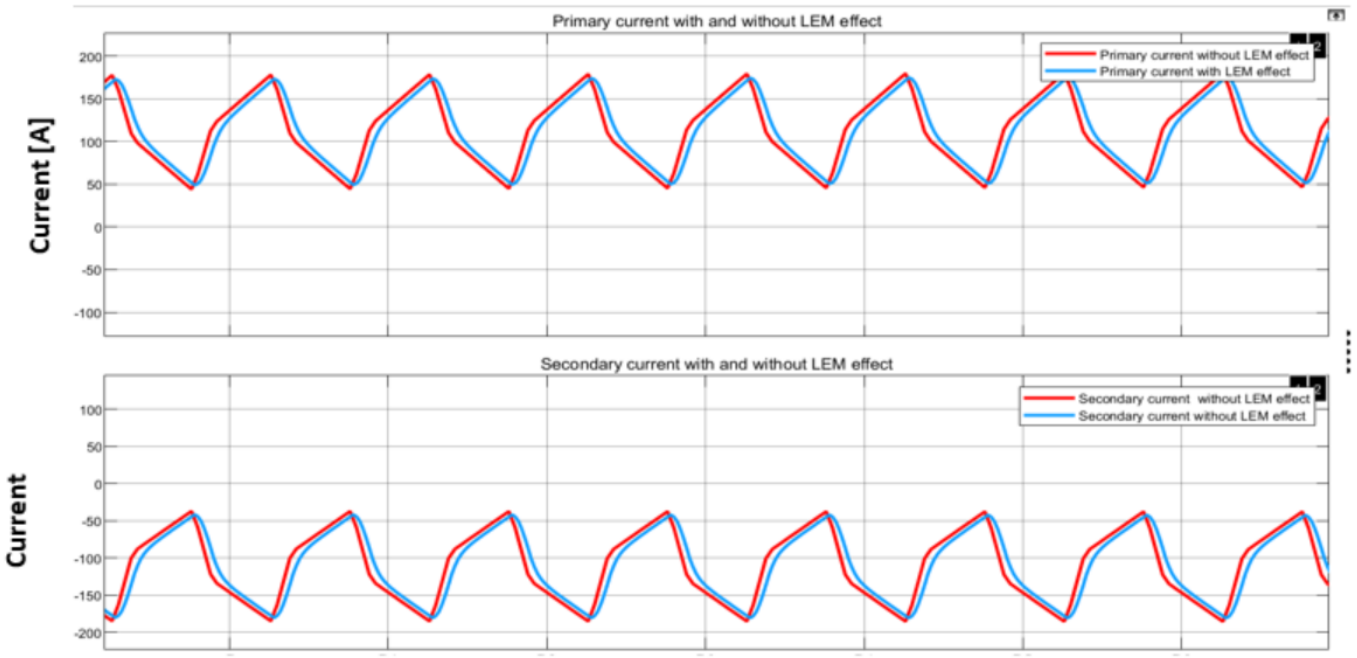


*Fig. 19. Primary and secondary current measurement with and without the LEM current sensor effect.*

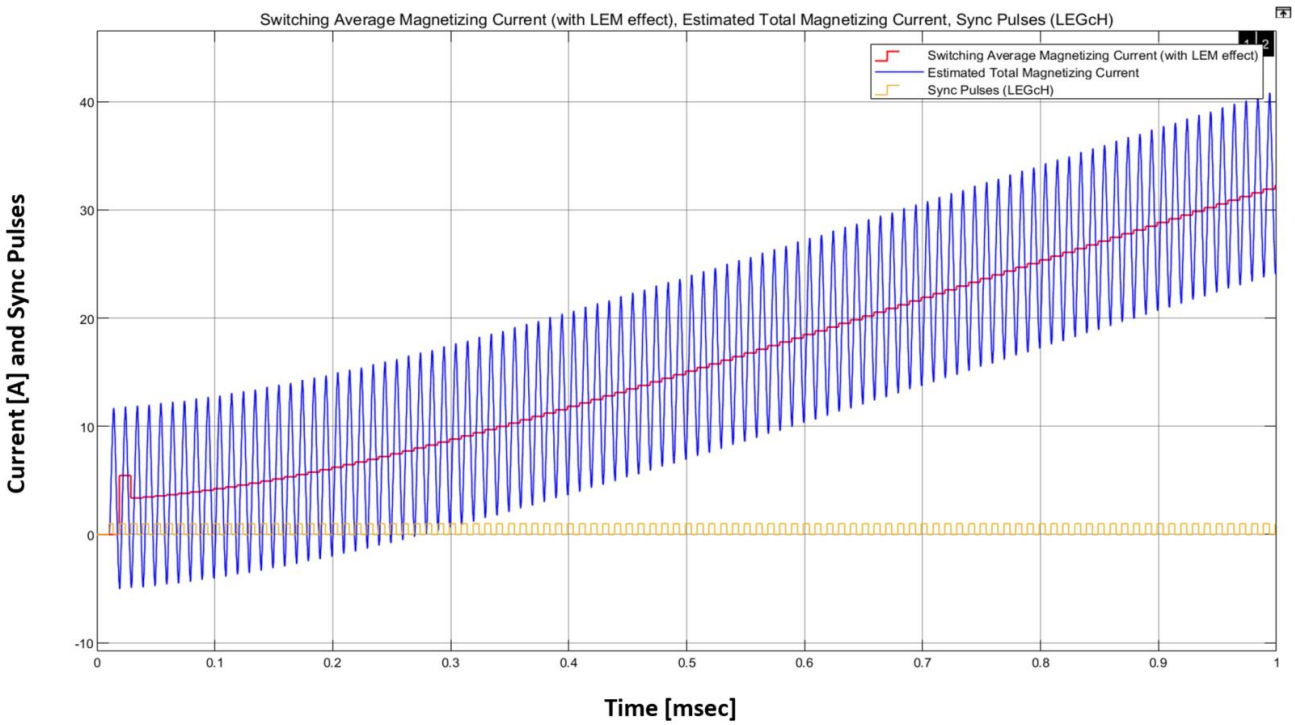


*Fig. 20. Estimated total magnetizing current (with LEM current sensor effect), estimated switching average (with LEM sensor effect), and synchronization pulses. Flux-balancing algorithm is inactive in this simulation.*

## *Wrapping Up*

As observed at the beginning of this discussion, we've taken a different angle here than in the previous parts in the series, delving into the control strategy implementation, and how this can be optimized and accelerated. This article unveiled the beneficial approach followed by the onsemi engineering team, which favors the debugging and identification of errors early-on during the simulation stage, before hardware is being produced.

Furthermore, this approach speeds up control development where hybrid controllers with ARM cores and FPGAs are combined. And finally, it facilitates the use of an FPGA by firmware engineers that are not experts in FPGA development, by automating the generation of FPGA IPs out of the simulation models created in Simulink. It's understood that the actual validation of the control algorithm will happen when the firmware can be deployed on the boards and the actual complete system is validated.

**References**

1. "Developing A 25-kW SiC-Based Fast DC Charger (Part 1): The EV Application" by Oriol Filló, Karol Rendek, Stefan Kosterec, Daniel Pruna, Dionisis Voglitsis, Rachit Kumar and Ali Husain, How2Power Today, April 2021.
2. "Developing A 25-kW SiC-Based Fast DC Charger (Part 2): Solution Overview" by Oriol Filló, Karol Rendek, Stefan Kosterec, Daniel Pruna, Dionisis Voglitsis, Rachit Kumar and Ali Husain, How2Power Today, May 2021.
3. "Developing A 25-kW SiC-Based Fast DC Charger (Part 3): PFC Stage Simulation" by Oriol Filló, Karol Rendek, Stefan Kosterec, Daniel Pruna, Dionisis Voglitsis, Rachit Kumar and Ali Husain, How2Power Today, June 2021.
4. "Developing A 25-kW SiC-Based Fast DC Charger (Part 4): Design Considerations And Simulation Of The DC-DC Stage" by Oriol Filló, Karol Rendek, Stefan Kosterec, Daniel Pruna, Dionisis Voglitsis, Rachit Kumar and Ali Husain, How2Power Today, July 2021.
5. SECO-TE0716-GEVB product page. Universal Controller Board (UCB) with Zynq-7000 SoC FPGA and ARM-based processor.
6. Model-Based Testing, page on MathWorks website.
7. "Circuit Averaging, Averaged Switch Modeling, and Simulation," pp 547-583 in *Fundamentals of Power Electronics* by Robert W. Erickson and Dragan Maksimović.
8. "On Board Charger (OBC) Three-phase PFC Converter," ON Semiconductor application note AND9957/D, June 2021.
9. "Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System" by Biao Zhao, Qiang Song, Wenhua Liu, and Yandong Sun. IEEE Transactions on Power Electronics, Vol. 29, Issue 8, Aug. 2014.

**About The Authors**

*Oriol Filló serves as a solution marketing engineer for industrial applications at ON Semiconductor. He is responsible for the marketing strategy of industrial solutions, focusing on robotics and energy infrastructure. He has developed his career in the electronics industry with a focus on power and control, and gathered experience in industrial, IoT and automotive applications.*

*Prior to joining ON Semiconductor in 2019, Oriol worked at Industrial Shields and PRAX Inductive Components in technical sales and business development roles, where among others, he propelled growth of the export business. Oriol received an engineering degree in energy engineering and a M.Sc. in industrial automation systems and industrial electronics from Universitat Politècnica de Catalunya. Oriol also holds a master in management from EADA Business School.*

*Dionisis Voglitsis is an applications engineer at ON Semiconductor. He is responsible for the development and implementation of control algorithms and control schemes for motor control and charging applications. Before joining ON Semiconductor in 2019, Dionisis worked as a researcher for various European and national research projects, while he had also joined the Advanced Technology Center of Philips. Dionisis is the author and co-author of more than 30 research and technical papers in his field, published in high-quality journals (IEEE Transactions and Journals), which have been cited in more than 200 papers. He is also a guest editor for the "Energies" MDPI journal. He holds an engineering degree in energy engineering, an M.Sc. in wireless power transfer from TU Delft, The Netherlands, and a Ph.D. in electrical engineering from Democritus University of Thrace (DUTH), Greece.*

*Karol Rendek is an applications manager at the Systems Engineering Center at ON Semiconductor. Karol joined ON Semiconductor in 2020. Previously, he spent nine years working as hardware engineer, system engineer and project manager in development of embedded systems, Class D amplifiers, rolling stock control and safety systems and industrial electric vehicle chargers. Karol has Master's degree and Ph.D. in Microelectronics from Slovak University of Technology in Bratislava. He spent three years during his Ph.D. study focusing on low frequency noise analysis of GaN HEMT transistors.*

*Stefan Kosterec is an application engineer at the Systems Engineering Center ON Semiconductor. Stefan joined ON Semiconductor in 2013. Previously, he spent eight years at Siemens PSE as ASIC/FPGA designer where he developed digital solutions targeted for various areas, among others communications, power conversion and motor control. He spent also two years at Vacuumschmelze acting as inductive components designer and also took a role of product integrity engineer at Emerson Energy Systems responsible for verification of telecom power systems. Stefan has a master's degree in Applied informatics from the Faculty of Materials Science and Technology of Slovak Technical University Trnava.*

*Rachit Kumar is a senior applications engineer at the Systems Engineering Center at ON Semiconductor. Rachit joined ON Semiconductor in 2020. Rachit has been engaged for more than ten years on embedded software development focusing on motor control algorithms. Prior to joining ON Semiconductor, Rachit worked at Nanotec Electronics doing embedded systems development for low power BLDC and stepper motor controllers. Rachit has a master's degree in mechatronics from the University of Applied Sciences, Ravensburg-Weingarten, Germany.*

For further reading on designing EV chargers, see the How2Power Design Guide, locate the Application category and select "Automotive".