Example n°6

CONTROL OF A GRID-TIED MODULAR MULTILEVEL CONVERTER (MMC)

Written by: Imperix Ltd, Rue de l'Industrie 17, 1950 Sion, Switzerland

Requirements:
- Matlab/Simulink ≥ 2015a
- Simulink Coder, Matlab Coder, Embedded Coder
- PLECS viewer ≥ 4.0.1 – free download here.
- BoomBox ACG blockset ≥ 1.1 – request trial and license here.

1 INTRODUCTION

This example shows the essential elements of a possible control implementation for a grid-tied three-phase nine-level Modular Multilevel Converter consisting of 24 submodules (Figure 1). The control is meant to be implemented using 3 BoomBox units, through the automated code generation process (ACG).

The selected control approach is inspired from [1], which is one of the simplest possible control scheme including a complete closed-loop control of all state variables. The related Simulink files contains not only the control implementation, but also a model of the plant. Hence, these files can be simultaneously used for simulation and automated code generation purposes.

The utilized parameters correspond to imperix's MMC hardware bundle (Figure 2). As such, this example is also meant to serve as a ready-made template for any control developments intended for this hardware.

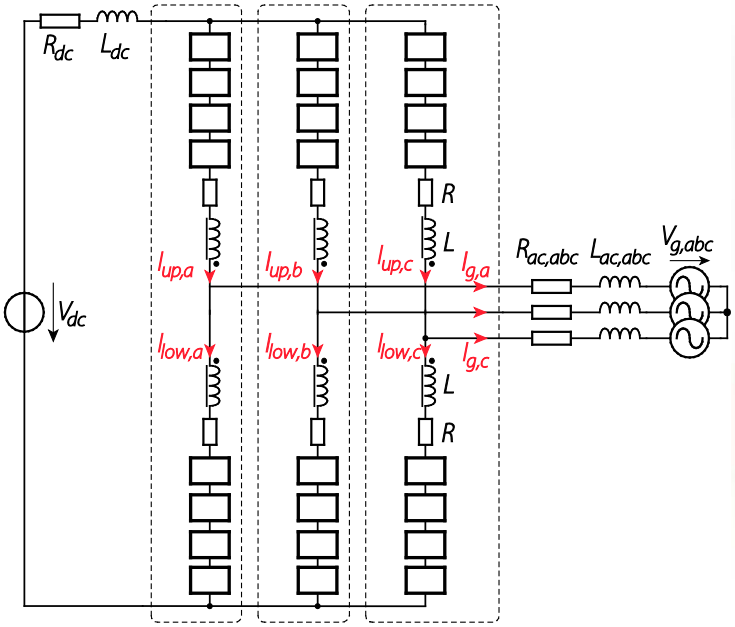

Figure 1: Electrical scheme of the considered system

Figure 2 : Reference prototyping hardware

2 GETTING STARTED WITH THE FILES

Apart from the minimum software requirements (Simulink/PLECS/ACG blockset), the complete simulation and control files are contained in:

- MMC_2015a.slx , contains both simulation and control schemes.
- init.m, contains the base hardware parameters of the plant.

The init.m file is automatically called at the beginning of the simulation or code generation process (using the Init callback of the *.slx file). As usual with imperix's Simulink files, the control part is clearly separated from the plant model, which is only used in simulation.

All Simulink sheets are commented so that their operation can be easily understood.

2.a PRINCIPLES OF OPERATION

The choice of the execution mode can be made in the Simulink config. block, located in the control section (Figure 3). Depending on the execution mode (Simulation/Code Generation), the appropriate variant for each subsystem is used, leading to the corresponding behavior:

- **Simulation**, Simulink runs the plant model as well as the variants corresponding to each input/output peripheral (modulators, analog inputs, etc.). This leads to an accurate simulation of the overall system, taking into account the exact sampling instants and PWM patterns.
- **Code Generation**, Simulink disregards the plant model and executes the appropriate C/C++ driver functions of the BoomBox(es). Real-time control code is automatically generated, compiled and downloaded into the master BoomBox.

This configuration block also controls some key parameters, such as:

- **Simulation step size**, corresponds to the time step between each iteration of the simulation solver. As such, only fixed-step solvers can be used. This step size is usually much smaller than the switching period, such that PWM-related phenomena can be highlighted by the simulation (PWM patterns, switching ripple, etc.).
- **Switching frequency**, corresponds to the carrier frequency utilized for all PWM modulators. In most cases, this also corresponds to the frequency at which the control algorithms are run (discrete control frequency).

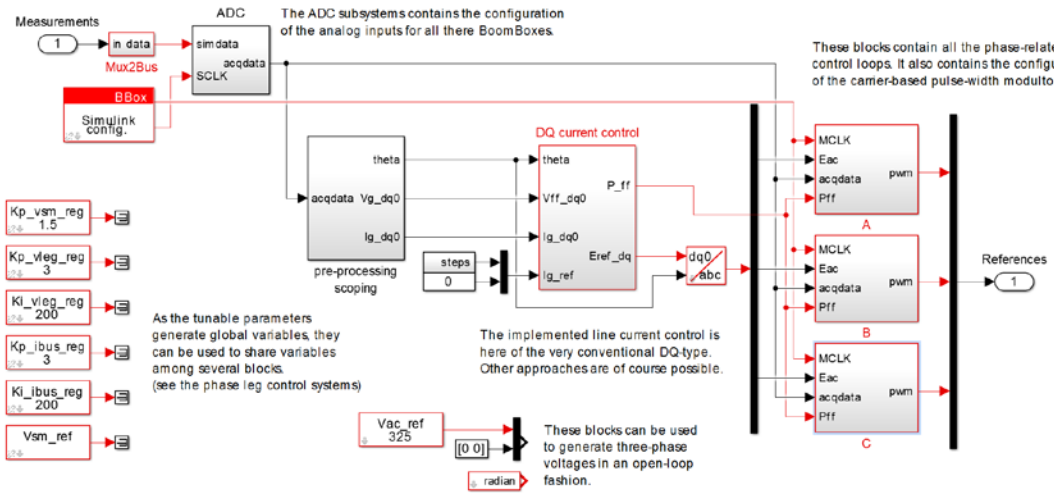

Figure 3 : Top-level scheme of the implemented control strategy.

2.b PLANT MODEL

The plant model is implemented using PLECS. It can be run using the freely-available PLECS Viewer license (available on PLEXIM website). Each submodule is implemented using a half-bridge topology, modeled with ideal switches and anti-parallel diodes.

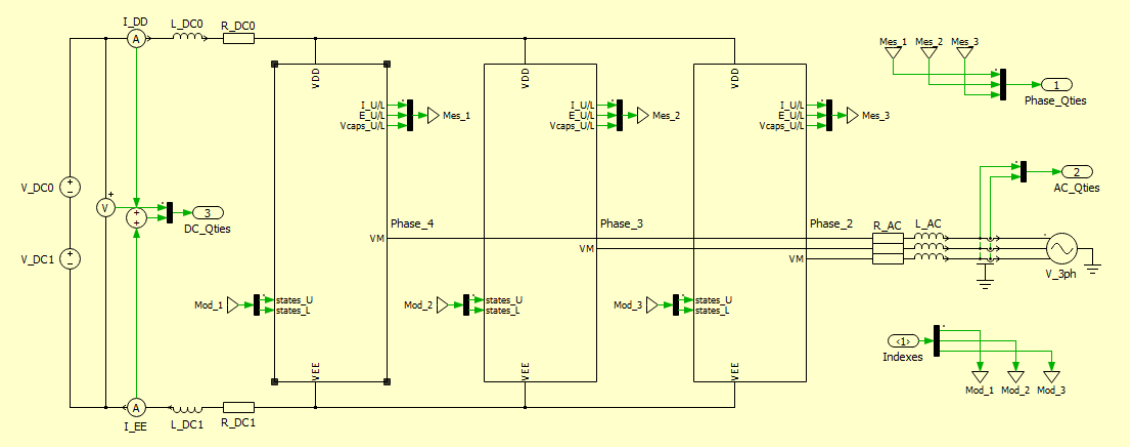

Figure 4 : Top-level schematic of the PLECS-based plant model

3 HARDWARE AND SOFTWARE CONFIGURATION

The configuration of all analog inputs for all BoomBoxes is entirely contained within the ADC subsystem (Figure 3), where all the ADC configuration blocks are conveniently grouped.

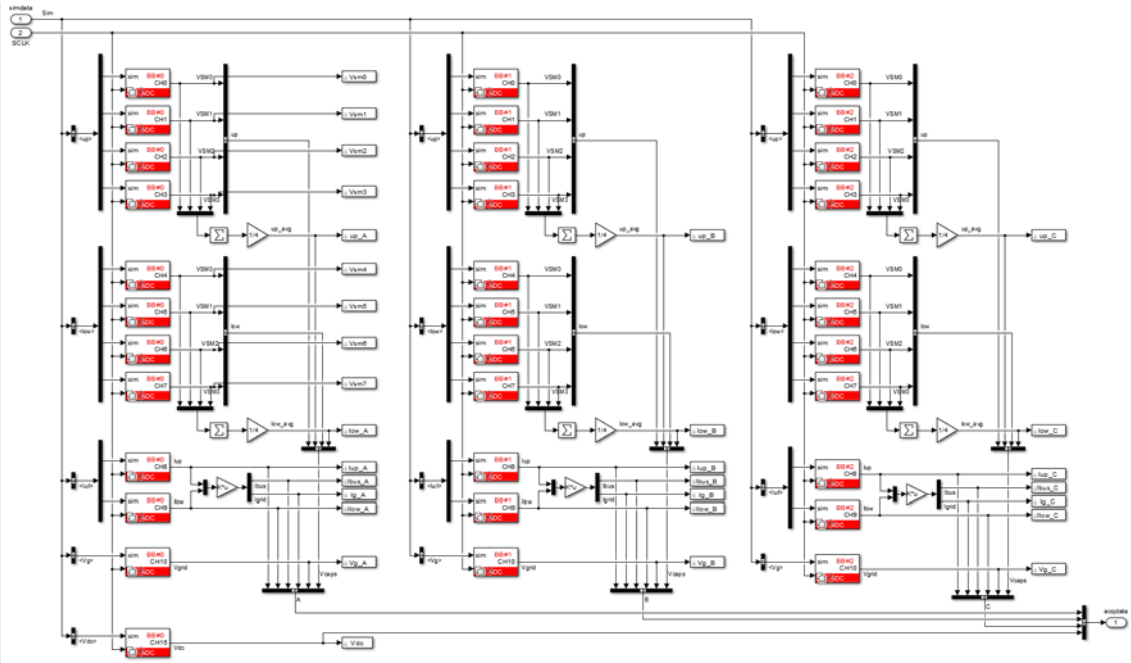

Figure 5 : Content of the ADC subsystem, including all ADC configuration blocks

Similarly, for each phase, the corresponding subsystem (Figure 5) contains the carrier-based modulators associated with each submodule, which are phase-shifted according to the desired modulation pattern. Each modulator is in this case directly fed by a local voltage controller, responsible for maintaining the appropriate charge level inside the submodule capacitors(s).

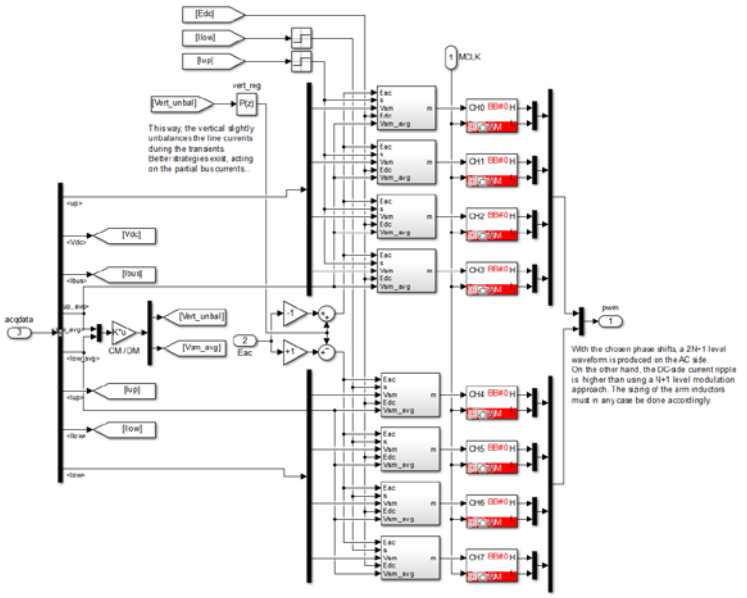

Figure 6 : Content of each phase-related subsystem

4 IMPLEMENTATION OF THE CONTROL APPLICATION

The control implementation is largely inspired from [1], which is an essential reference regarding the control of MMC with carrier-based modulation that the reader is strongly advised to consult.

Two minor differences are however present – and suggested – between [1] and this example:

- The **arm-level vertical energy balancing** is not achieved indirectly through the so-called averaging control, but instead is done explicitly by an associated controller, acting on the AC-side converter reference. This approach offers a better decoupling of the cell- and arm-level balancing mechanisms and allows to improve the cell-level dynamics.

  On the other hand, the corresponding energy exchanges remain related to the AC side of each phase leg, which means that they may lead to slight asymmetries in the grid currents during the balancing transients. Alternative approaches exist, typically altering the circulating currents in such a way that the total DC is unchanged. A recommended reference is [2].

- The **DC-side dynamics** are improved by feed-forwarding the AC-side active power into the DC-side current control. This facilitates the control of the total embedded energy, i.e. overall average capacitor voltage. More subtle approaches can be used, which are typically relevant in case of operation under unbalanced grid conditions, or single-phase systems.

4.a IMPORTANT COMMENTS

- Several controllers are sharing the same parameters (Kp, Ki), which are implemented as global variables using the **tunable parameter** block from imperix's ACG blockset. This translates into so-called Simulink global signals, which are visible throughout a Simulink model.
- Simulink busses are used to aggregate data and transmit numerous variable across Simulink sheets. Unfortunately, busses cannot be translated into C structures, unless the latter are defined manually using Simulink's so-called non-virtual busses.

- There is obviously no start-up/shut-down procedure implemented in the Simulink files. When needed, such mechanisms can be implemented using Simulink Stateflow. However, for complex projects, a control implementation using C/C++ may be preferable.
- It is strongly advised to use Simulink's sample time highlighting options to control the exact execution rate of each block. These options are conveniently available from the menu **Display >> Sample Time >> Colors**. Indeed, inappropriate execution rates are a common source of mismatch between simulation and the actual real-time control implementation.

4.b ILLUSTRATIVE SIMULATION RESULTS

Figure 7 shows some simulation results from the presented example:

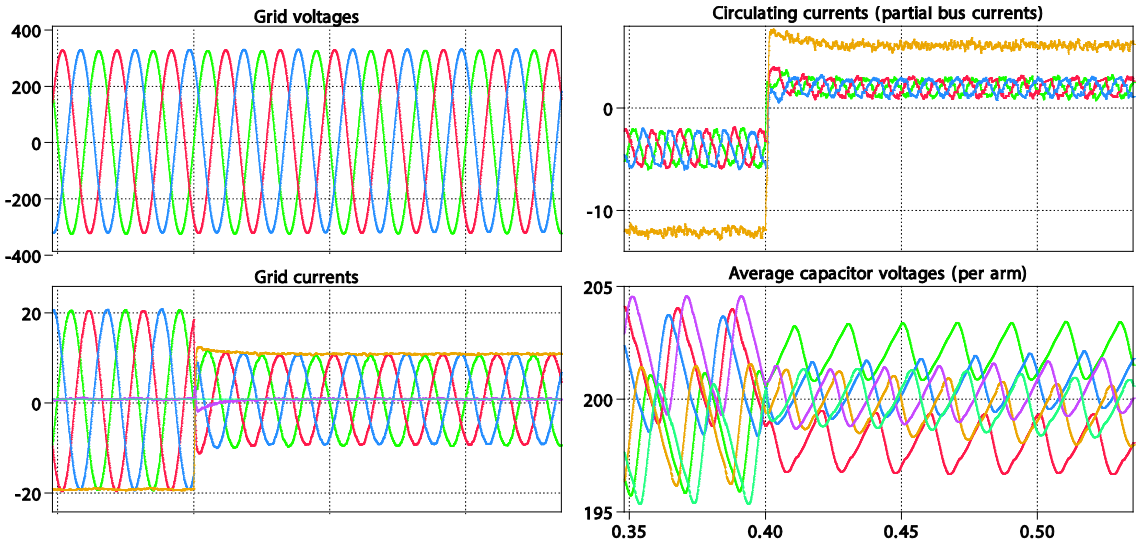

Figure 7 : Excerpt of the simulation results, showing a complete power reversal at t=0.4s.

5 REFERENCES

[1] M. Hagiwara, H. Akagi, *"Control and Experiment of Pulsewidth-Modulated Modular Multilevel Converters,"* in IEEE Transactions on Power Electronics, Vol.24, July 2009.

[2] P. Münch, D. Görges, M. Izák and S. Liu, *"Integrated current control, energy control and energy balancing of Modular Converters,"* in Proc. IECON Conference, Phoenix, 2010.